



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 © 2026 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

.....	xvii
什麼是 Managed Service for Apache Flink ?	1
決定使用 Managed Service for Apache Flink 或 Managed Service for Apache Flink Studio	1
選擇要在 Managed Service for Apache Flink 中使用的 Apache Flink APIs	3
選擇 Flink API	3
開始使用串流資料應用程式	4
運作方式	6
設定您的 Apache Flink 應用程式	6
DataStream API	6
資料表 API	7
建立 Managed Service for Apache Flink 應用程式	7
建立應用程式	8
建置 Managed Service for Apache Flink 應用程式程式碼	8
建立 Managed Service for Apache Flink 應用程式	9
使用客戶受管金鑰	10
啟動 Managed Service for Apache Flink 應用程式	10
驗證 Managed Service for Apache Flink 應用程式	11
啟用系統轉返	11
執行應用程式	13
識別應用程式和任務狀態	13
執行批次工作負載	15
應用程式資源	15
Managed Service for Apache Flink 應用程式資源	15
Apache Flink 應用程式資源	15
定價	16
運作方式	15
AWS 區域 可用性	17
定價範例	18
檢閱 DataStream API 元件	21
連接器	22
運算子	30
事件追蹤	31
資料表 API 元件	32
資料表 API 連接器	32

資料表 API 時間屬性	34
使用 Python	34
程式設計您的 Python 應用程式	35
建立您的 Python 應用程式	37
監控您的 Python 應用程式	38
使用執行期屬性	40
使用主控台管理執行期屬性	40
使用 CLI 管理執行期屬性	41
在 Managed Service for Apache Flink 應用程式中存取執行期屬性	43
使用 Apache Flink 連接器	44
Flink 2.2 的連接器	45
舊版 Flink 的連接器	46
實作容錯能力	48
在 Managed Service for Apache Flink 中設定檢查點	49
檢閱檢查點 API 範例	50
使用快照管理應用程式備份	52
管理自動建立快照	53
從包含不相容狀態資料的快照還原	53
檢閱快照 API 範例	54
針對 Apache Flink 使用就地版本升級	57
升級應用程式	57
升級至新版本	58
復原應用程式升級	64
最佳實務	64
已知問題	65
升級到 Flink 2.2	66
狀態相容性指南	74
實作應用程式擴展	79
設定應用程式平行處理和ParallelismPerKPU	79
配置 Kinesis 處理單元	80
更新應用程式的平行處理	81
使用自動擴展	82
maxParallelism 考量	84
將標籤新增至應用程式	85
建立應用程式時新增標籤	85
新增或更新現有應用程式的標籤	86

列出應用程式的標籤	86
從應用程式移除標籤	87
使用 CloudFormation	87
開始之前	87
撰寫 Lambda 函數	87
建立 Lambda 角色	89
叫用 Lambda 函數	90
檢閱延伸範例	90
使用 Apache Flink 儀表板	96
存取應用程式的 Apache Flink 儀表板	96
支援和已棄用版本	98
Amazon Managed Service for Apache Flink 2.2	105
Amazon Managed Service for Apache Flink 2.2 的新功能	105
中斷變更和棄用	107
支援 Apache Flink 2.2 功能	108
連接器可用性	110
不支援和實驗性功能	111
已知問題	112
升級體驗	112
後續步驟	112
Amazon Managed Service for Apache Flink 1.20	112
支援的功能	113
元件	114
已知問題	114
Amazon Managed Service for Apache Flink 1.19	115
支援的功能	115
Amazon Managed Service for Apache Flink 1.19.1 的變更	117
元件	118
已知問題	119
Amazon Managed Service for Apache Flink 1.18	119
使用 Apache Flink 1.15 的 Amazon Managed Service for Apache Flink 中的變更	120
元件	122
已知問題	122
Amazon Managed Service for Apache Flink 1.15	123
使用 Apache Flink 1.15 的 Amazon Managed Service for Apache Flink 中的變更	124
元件	122

已知問題	125
舊版	125
將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用	127
使用 Apache Flink 1.8.2 建置應用程式	128
使用 Apache Flink 1.6.2 建置應用程式	128
升級應用程式	129
Apache Flink 1.6.2 和 1.8.2 中的可用連接器	130
Flink 1.13.2 入門	130
Flink 1.11.1 入門	154
入門：Flink 1.8.2 - 棄用	179
入門：Flink 1.6.2 - 棄用	203
舊版範例	226
搭配 Managed Service for Apache Flink 使用 Studio 筆記本	389
使用正確的 Studio 筆記本執行期版本	390
建立 Studio 筆記本	390
執行串流資料的互動式分析	391
Flink 解譯器	392
Apache Flink 資料表環境變數	393
將 部署為具有持久狀態的應用程式	393
Scala/Python 條件	394
SQL 條件	395
IAM 許可	395
使用連接器和相依性	395
預設連接器	396
新增相依性和自訂連接器	397
使用者定義的函數	398
使用使用者定義函數的考量	399
啟用檢查點	400
設定檢查點間隔	400
設定檢查點類型	400
升級 Studio 執行期	401
將您的筆記本升級至新的 Studio 執行期	401
使用 AWS Glue	405
資料表屬性	405
Managed Service for Apache Flink 中 Studio 筆記本的範例和教學課程	407
教學課程：在 Managed Service for Apache Flink 中建立 Studio 筆記本	407

教學課程：將 Studio 筆記本部署為具有持久狀態的 Managed Service for Apache Flink 應用程式	426
檢視 Studio 筆記本中分析資料的範例查詢	429
Managed Service for Apache Flink 的 Studio 筆記本疑難排解	441
停止停滯的應用程式	441
在無網際網路存取的 VPC 中，部署為具有持久狀態的應用程式	441
部署為應用程式的大小和建置時間減少	442
取消任務	444
重新啟動 Apache Flink 解譯器	445
為 Managed Service for Apache Flink Studio 筆記本建立自訂 IAM 政策	445
AWS Glue	446
CloudWatch Logs	447
Kinesis 串流	448
Amazon MSK 叢集	450
教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API	451
檢閱應用程式元件	179
完成必要的先決條件	452
設定 帳戶	453
註冊 AWS 帳戶	131
建立具有管理存取權的使用者	132
授與程式設計存取權	455
後續步驟	456
設定 AWS CLI	456
下一步驟	458
建立應用程式	458
建立相依資源	458
設定您的本機開發環境	460
下載並檢查 Apache Flink 串流 Java 程式碼	461
將範例記錄寫入輸入串流	465
在本機執行您的應用程式	466
觀察 Kinesis 串流中的輸入和輸出資料	469
停止應用程式在本機執行	469
編譯和封裝您的應用程式程式碼	469
上傳應用程式碼 JAR 檔案	470
建立和設定 Managed Service for Apache Flink 應用程式	471
下一步驟	477

清除資源	477
刪除 Managed Service for Apache Flink 應用程式	478
刪除您的 Kinesis 資料串流	478
刪除您的 Amazon S3 物件和儲存貯體	478
刪除您的 IAM 資源	479
刪除您的 CloudWatch 資源	479
探索 Apache Flink 的其他資源	479
探索其他資源	480
教學課程：開始使用 Managed Service for Apache Flink 中的 TableAPI	481
檢閱應用程式元件	481
完成必要的先決條件	482
建立應用程式	482
建立相依資源	483
設定您的本機開發環境	484
下載並檢查 Apache Flink 串流 Java 程式碼	484
在本機執行您的應用程式	490
觀察應用程式將資料寫入 S3 儲存貯體	492
停止應用程式在本機執行	493
編譯和封裝您的應用程式程式碼	493
上傳應用程式碼 JAR 檔案	493
建立和設定 Managed Service for Apache Flink 應用程式	494
下一步驟	500
清除資源	500
刪除 Managed Service for Apache Flink 應用程式	500
刪除您的 Amazon S3 物件和儲存貯體	501
刪除您的 IAM 資源	501
刪除您的 CloudWatch 資源	502
下一步驟	502
探索其他資源	502
教學課程：開始使用 Managed Service for Apache Flink 中的 Python	503
檢閱應用程式元件	503
滿足先決條件	504
建立應用程式	505
建立相依資源	506
設定您的本機開發環境	507
下載並檢查 Apache Flink 串流 Python 程式碼	508

管理 JAR 相依性	511
將範例記錄寫入輸入串流	513
在本機執行您的應用程式	514
觀察 Kinesis 串流中的輸入和輸出資料	517
停止應用程式在本機執行	517
封裝您的應用程式程式碼	517
將應用程式套件上傳至 Amazon S3 儲存貯體	517
建立和設定 Managed Service for Apache Flink 應用程式	518
下一步驟	524
清除資源	524
刪除 Managed Service for Apache Flink 應用程式	524
刪除您的 Kinesis 資料串流	525
刪除您的 Amazon S3 物件和儲存貯體	525
刪除您的 IAM 資源	526
刪除您的 CloudWatch 資源	526
教學課程：開始使用 Managed Service for Apache Flink 中的 Scala	527
建立相依資源	527
將範例記錄寫入輸入串流	528
下載並檢查應用程式程式碼	530
編譯和上傳應用程式的程式碼	531
建立並執行應用程式（主控台）	532
建立應用程式	532
設定應用程式	533
編輯 IAM 政策	534
執行應用程式	536
停止應用程式	536
建立並執行應用程式 (CLI)	536
建立許可政策	536
建立 IAM 政策	538
建立應用程式	540
啟動應用程式	541
停止應用程式	384
新增 CloudWatch 記錄選項	385
更新環境屬性	385
更新應用程式的程式碼	386
清除 AWS 資源	544

刪除 Managed Service for Apache Flink 應用程式	544
刪除您的 Kinesis 資料串流	544
刪除您的 Amazon S3 物件和儲存貯體	545
刪除您的 IAM 資源	545
刪除您的 CloudWatch 資源	545
將 Apache Beam 與 Managed Service for Apache Flink 應用程式搭配使用	546
使用 Managed Service for Apache Flink 的 Apache Flink Runner 限制	546
使用 Managed Service for Apache Flink 的 Apache Beam 功能	547
使用 Apache Beam 建立應用程式	547
建立相依資源	547
將範例記錄寫入輸入串流	548
下載並檢查應用程式程式碼	549
編譯應用程式程式碼	550
上傳 Apache Flink 串流 Java 程式碼	550
建立並執行 Managed Service for Apache Flink 應用程式	551
清除	554
後續步驟	556
訓練研討會、實驗室和解決方案實作	557
Managed Service for Apache Flink 研討會	557
在本機開發 Apache Flink 應用程式，然後再部署到 Managed Service for Apache Flink	557
使用 Managed Service for Apache Flink Studio 進行事件偵測	558
AWS 串流資料解決方案	558
練習搭配 Apache Flink 和 Apache Kafka 使用 Clickstream 實驗室	558
使用 Application Auto Scaling 設定自訂擴展	558
檢視範例 Amazon CloudWatch 儀表板	559
使用 範本串流 AWS Amazon MSK 的資料解決方案	559
在 GitHub 上探索更多 Managed Service for Apache Flink 解決方案	559
使用 Managed Service for Apache Flink 的實用公用程式	560
快照管理員	560
基準測試	560
建立和使用 Managed Service for Apache Flink 應用程式的範例	561
Managed Service for Apache Flink 的 Java 範例	561
Managed Service for Apache Flink 的 Python 範例	564
Managed Service for Apache Flink 的 Scala 範例	566
Managed Service for Apache Flink 中的安全性	567
資料保護	567

資料加密	567
Amazon MSF 中的金鑰管理	568
Amazon MSF 中的透明加密	569
Amazon MSF 中的客戶受管金鑰	569
使用客戶受管金鑰	575
使用主控台管理 CMK	578
使用 APIs 管理 CMK	579
Managed Service for Apache Flink 的 Identity and Access Management	588
目標對象	588
使用身分驗證	589
使用政策管理存取權	590
Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用	591
身分型政策範例	600
疑難排解	602
預防跨服務混淆代理人	604
法規遵循驗證	605
FedRAMP	606
Managed Service for Apache Flink 中的彈性和災難復原	606
災難復原	606
版本控制	607
Managed Service for Apache Flink 中的基礎設施安全性	607
Managed Service for Apache Flink 的安全最佳實務	608
實作最低權限存取	608
使用 IAM 角色存取其他 Amazon 服務	608
在相依資源中實作伺服器端加密	608
使用 CloudTrail 監控 API 呼叫	608
在 Amazon Managed Service for Apache Flink 中記錄和監控	610
登入 Managed Service for Apache Flink	610
使用 CloudWatch Insights 查詢日誌	611
Managed Service for Apache Flink 中的監控	611
在 Managed Service for Apache Flink 中設定應用程式記錄	612
使用主控台設定 CloudWatch 記錄	613
使用 CLI 設定 CloudWatch 記錄	613
控制應用程式監控層級	618
套用記錄最佳實務	619
執行記錄疑難排解	619

使用 CloudWatch Logs Insights	619
使用 CloudWatch Logs Insights 分析日誌	619
執行範例查詢	620
檢閱範例查詢	621
Managed Service for Apache Flink 中的指標和維度	623
應用程式指標	624
Kinesis Data Streams 連接器指標	645
Amazon MSK 連接器指標	645
Apache Zeppelin 指標	647
檢視 CloudWatch 指標	647
設定 CloudWatch 指標報告層級	648
搭配 Amazon Managed Service for Apache Flink 使用自訂指標	649
搭配 Amazon Managed Service for Apache Flink 使用 CloudWatch 警示	653
將自訂訊息寫入 CloudWatch Logs	664
使用 Log4J 寫入 CloudWatch 日誌	664
使用 SLF4J 寫入 CloudWatch 日誌	665
使用 記錄 Managed Service for Apache Flink API 呼叫 AWS CloudTrail	666
CloudTrail 中的 Managed Service for Apache Flink 資訊	666
了解 Managed Service for Apache Flink 日誌檔案項目	667
調校效能	670
對效能問題進行故障診斷	670
了解資料路徑	670
效能疑難排解解決方案	670
使用效能最佳實務	672
適當管理擴展	672
監控外部相依性資源使用量	674
在本機執行 Apache Flink 應用程式	675
監控效能	675
使用 CloudWatch 指標監控效能	675
使用 CloudWatch 日誌和警示監控效能	675
Managed Service for Apache Flink 和 Studio 筆記本配額	676
管理 Managed Service for Apache Flink 的維護任務	678
選擇維護時段	680
識別維護執行個體	680
實現 Managed Service for Apache Flink 應用程式的生產準備	681
負載測試您的應用程式	681

定義最大平行處理	681
為所有運算子設定 UUID	682
最佳實務	683
最小化 uber JAR 的大小	683
容錯能力：檢查點和儲存點	685
不受支援的連接器版本	686
效能與平行處理層級	686
設定每個運算子的平行處理層級	687
記錄	687
編碼	687
管理憑證	688
從具有較少碎片/分割區的來源讀取	688
Studio 筆記本重新整理間隔	689
Studio 筆記本最佳效能	689
浮水印策略和閒置碎片如何影響時間範圍	689
摘要	690
範例	691
為所有運算子設定 UUID	700
將 ServiceResourceTransformer 新增至 Maven 著色外掛程式	700
Apache Flink 狀態函數	702
Apache Flink 應用程式範本	702
模組組態的位置	703
了解 Apache Flink 設定	704
Apache Flink 組態	704
狀態後端	704
檢查點	705
儲存點	706
堆積大小	706
緩衝區消脹	706
可修改的 Flink 組態屬性	707
重新啟動策略	707
檢查點和狀態後端	707
檢查點	707
RocksDB 原生指標	707
RocksDB 選項	709
進階狀態後端選項	709

完整 TaskManager 選項	709
記憶體組態	710
RPC / Akka	710
用戶端	710
進階叢集選項	710
檔案系統組態	711
進階容錯能力選項	711
記憶體組態	710
指標	711
REST 端點和用戶端的進階選項	711
進階 SSL 安全選項	711
進階排程選項	711
Flink Web UI 的進階選項	711
程式設計 Flink 組態屬性	712
管道組態	712
Python API	713
資料表 API/SQL	713
檢視設定的 Flink 屬性	717
設定 MSF 以存取 Amazon VPC 中的資源	719
Amazon VPC 概念	719
VPC 應用程式許可	720
新增存取 Amazon VPC 的許可政策	720
為 VPC 連線的 Managed Service for Apache Flink 應用程式建立網際網路和服務存取	721
相關資訊	722
使用 Managed Service for Apache Flink VPC API	722
建立應用程式	723
AddApplicationVpcConfiguration	723
DeleteApplicationVpcConfiguration	724
更新應用程式	724
範例：使用 VPC	725
Managed Service for Apache Flink 故障診斷	726
開發疑難排解	726
系統復原最佳實務	726
Hudi 組態最佳實務	727
Apache Flink 火焰圖	728
EFO 連接器 1.15.2 的登入資料提供者問題	728

應用程式使用不支援的 Kinesis 連接器	728
編譯錯誤：「無法解析專案的相依性」	731
無效的選擇：「kinesisanalyticsv2」	731
UpdateApplication 動作不會重新載入應用程式程式碼	731
S3 StreamingFileSink FileNotFoundExceptions	732
使用儲存點停止時的 FlinkKafkaConsumer 問題	733
FLINK 1.15 非同步接收器死鎖	733
在重新分片期間，Amazon Kinesis 資料串流來源處理順序不正確	743
即時向量內嵌藍圖常見問答集和疑難排解	744
執行期疑難排解	753
故障診斷工具	754
應用程式問題	754
應用程式正在重新啟動	758
輸送量太慢	760
無限制狀態成長	761
I/O 綁定運算子	762
Kinesis 資料串流的上游或來源限流	762
檢查點	763
檢查點逾時	769
Apache Beam 的檢查點失敗	770
背壓	772
資料扭曲	773
狀態扭曲	773
與不同區域中的資源整合	774
文件歷史紀錄	775
API 範例程式碼	780
AddApplicationCloudWatchLoggingOption	781
AddApplicationInput	781
AddApplicationInputProcessingConfiguration	782
AddApplicationOutput	783
AddApplicationReferenceDataSource	783
AddApplicationVpcConfiguration	784
CreateApplication	784
CreateApplicationSnapshot	786
DeleteApplication	786
DeleteApplicationCloudWatchLoggingOption	786

DeleteApplicationInputProcessingConfiguration	786
DeleteApplicationOutput	787
DeleteApplicationReferenceDataSource	787
DeleteApplicationSnapshot	787
DeleteApplicationVpcConfiguration	787
DescribeApplication	788
DescribeApplicationSnapshot	788
DiscoverInputSchema	788
ListApplications	789
ListApplicationSnapshots	789
StartApplication	789
StopApplication	790
UpdateApplication	790
API 參考	791
發行版本	792

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

什麼是 Amazon Managed Service for Apache Flink ？

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

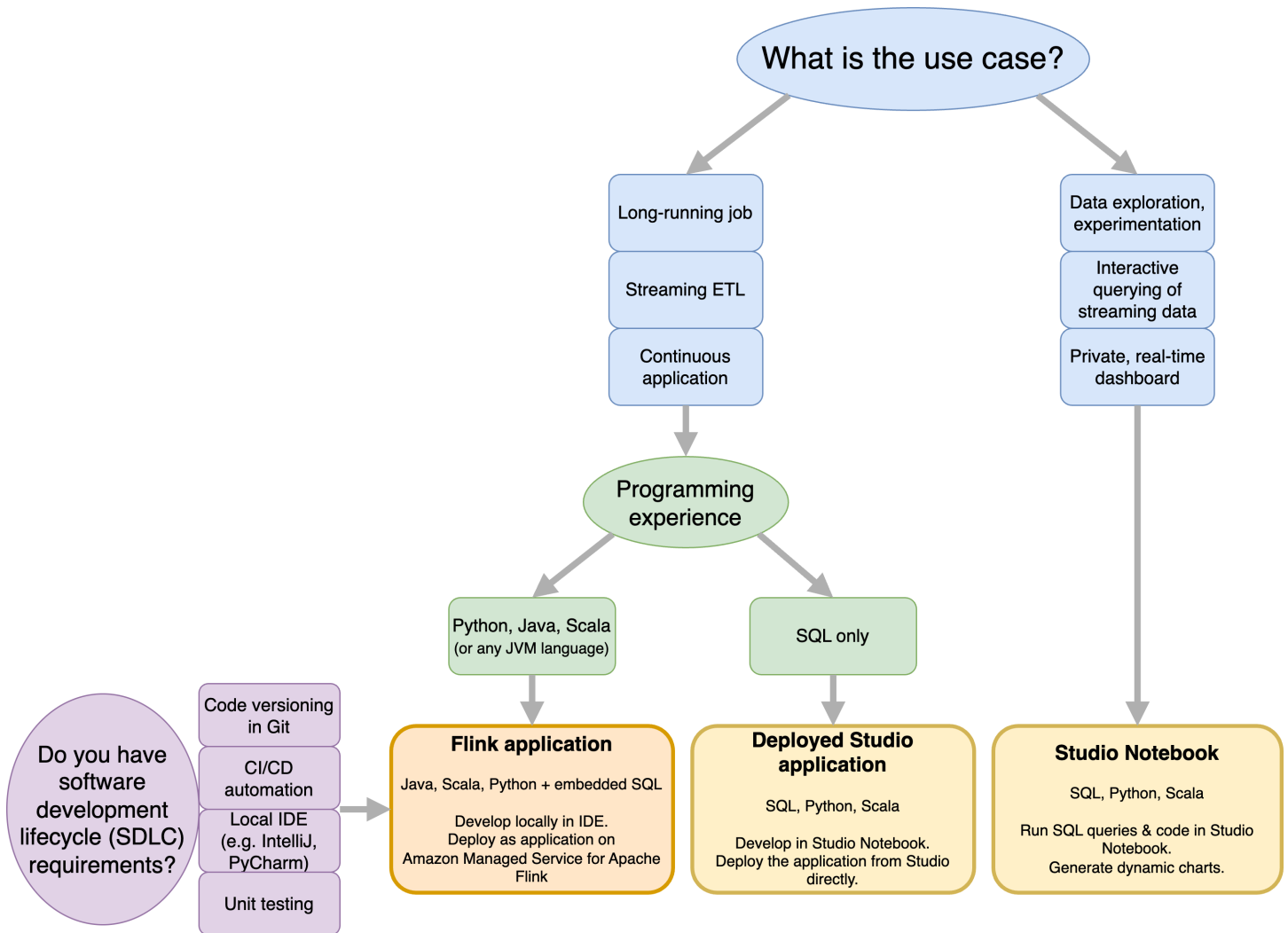
您可以使用以 Apache Flink 為基礎的開放原始碼程式庫，在 Managed Service for [Apache Flink](#) 中以您選擇的語言建置應用程式。Apache Flink 是處理資料串流的熱門框架及引擎。

Managed Service for Apache Flink 可為 Apache Flink 應用程式提供基礎設施。它處理核心功能，例如佈建運算資源、AZ 容錯移轉彈性、平行運算、自動擴展和應用程式備份（實作為檢查點和快照）。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 Managed Service for Apache Flink 或 Managed Service for Apache Flink Studio

您有兩個選項可以使用 Amazon Managed Service for Apache Flink 執行 Flink 任務。使用 [Managed Service for Apache Flink](#)，您可以使用您選擇的 IDE 和 Apache Flink Datastream 或 Table APIs，在 Java、Scala 或 Python（和內嵌 SQL）中建置 Flink 應用程式。使用 [Managed Service for Apache Flink Studio](#)，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的使用案例的方法。如果您不確定，本節將提供高階指引來協助您。



在決定是否使用 Amazon Managed Service for Apache Flink 或 Amazon Managed Service for Apache Flink Studio 之前，您應該考慮您的使用案例。

如果您計劃操作會處理串流 ETL 或持續應用程式等工作負載的長時間執行應用程式，您應該考慮使用 [Managed Service for Apache Flink](#)。這是因為您可以直接在您選擇的 IDE 中使用 Flink APIs 建立 Flink 應用程式。使用 IDE 在本機進行開發也可確保您可以利用軟體開發生命週期 (SDLC) 常見程序和工具，例如 Git 中的程式碼版本控制、CI/CD 自動化或單元測試。

如果您對隨機操作資料探索感興趣、想要以互動方式查詢串流資料，或建立私有即時儀表板，[Managed Service for Apache Flink Studio](#) 只需按幾下滑鼠就能協助您達成這些目標。熟悉 SQL 的使用者可以考慮直接從 Studio 部署長時間執行的應用程式。

 Note

您可以將 Studio 筆記本提升為長時間執行的應用程式。不過，如果您想要與 SDLC 工具整合，例如 Git 和 CI/CD 自動化上的程式碼版本控制，或是單元測試等技術，建議您使用所選的 IDE 來 Managed Service for Apache Flink。

選擇要在 Managed Service for Apache Flink 中使用的 Apache Flink APIs

您可以在 Managed Service for Apache Flink 中使用 Java、Python 和 Scala，在您選擇的 IDE 中使用 Apache Flink APIs 來建置應用程式。您可以在 [文件](#) 中找到如何使用 Flink Datastream 和資料表 API 建置應用程式的指引。您可以選擇您在中建立 Flink 應用程式的語言，以及您用來最符合應用程式和操作需求的 APIs。如果您不確定，本節提供高階指引來協助您。

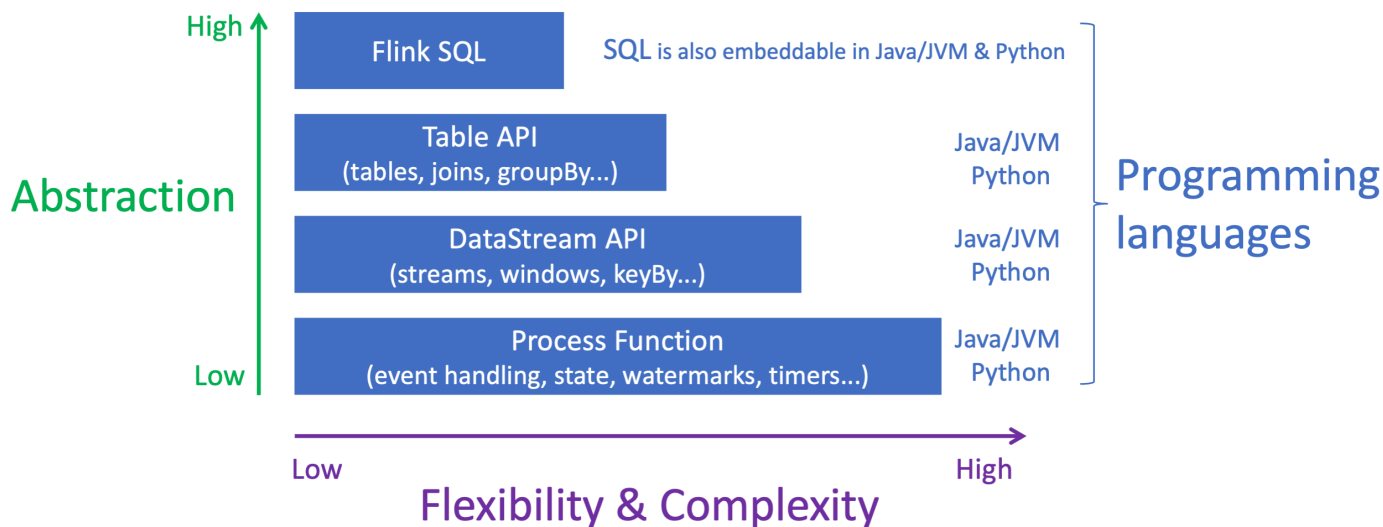
選擇 Flink API

Apache Flink APIs 具有不同層級的抽象，可能會影響您決定建置應用程式的方式。它們具有表達性和彈性，可以一起使用來建置您的應用程式。您不需要僅使用一個 Flink API。您可以在 [Apache Flink 文件中進一步了解 Flink APIs](#)。

Flink 提供四個 API 抽象層級：Flink SQL、資料表 API、DataStream API 和 Process Function，可與 DataStream API 搭配使用。這些都支援 Amazon Managed Service for Apache Flink。建議您盡可能從更高層級的抽象開始，但某些 Flink 功能僅適用於 [DataStream API](#)，您可以在其中以 Java、Python 或 Scala 建立應用程式。在以下情況下，您應該考慮使用 Datastream API：

- 您需要精細控制狀態
- 您想要利用非同步呼叫外部資料庫或端點的能力（例如用於推論）
- 您想要使用自訂計時器（例如實作自訂視窗或延遲事件處理）
- 您希望能夠在不重設狀態的情況下修改應用程式的流程

Apache Flink APIs



Note

使用 DataStream API 選擇語言：

- SQL 可以內嵌在任何 Flink 應用程式中，無論選擇何種程式設計語言。
- 如果您打算使用 DataStream API，則 Python 不支援所有連接器。
- 如果您需要低延遲/高輸送量，無論 API 為何，都應考慮 Java/Scala。
- 如果您計劃在 Process Functions API 中使用非同步 IO，則需要使用 Java。

API 的選擇也可能影響您發展應用程式邏輯的能力，而不必重設狀態。這取決於特定功能，即在運算子上設定 UID 的功能，僅適用於 Java 和 Python 的 DataStream API。如需詳細資訊，請參閱 Apache Flink 文件中的[為所有運算子設定 UUIDs](#)。

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，使用您選擇的 IDE 編寫程式碼，並使用即時串流資料對其進行測試。您也可以設定希望 Managed Service for Apache Flink 傳送結果的目的地。

建議您閱讀下列章節入門：

- [Managed Service for Apache Flink：運作方式](#)
- [開始使用 Amazon Managed Service for Apache Flink \(DataStream API\)](#)

或者，您可以先建立 Managed Service for Apache Flink Studio 筆記本，讓您以互動方式即時查詢資料串流，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 管理主控台，您就可以啟動無伺服器筆記本來查詢資料串流，並在幾秒鐘內取得結果。建議您閱讀下列章節入門：

- [使用 Studio 筆記本搭配 Managed Service for Apache Flink](#)
- [建立 Studio 筆記本](#)

Managed Service for Apache Flink：運作方式

Managed Service for Apache Flink 是一種全受管的 Amazon 服務，可讓您使用 Apache Flink 應用程式來處理串流資料。首先，您將 Apache Flink 應用程式編寫程式，然後建立 Managed Service for Apache Flink 應用程式。

設定您的 Apache Flink 應用程式

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

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

兩種 API 的功能如下：

DataStream API

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

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

使用 DataStream API 建立的應用程式會執行下列動作：

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

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

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

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

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

如需如何使用 DataStream API 的詳細資訊，請參閱 [檢閱 DataStream API 元件](#)。

資料表 API

Apache Flink 資料表 API 程式設計模型以下列元件為基礎：

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

使用資料表 API 建立的應用程式會執行下列動作：

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

使用資料表 API 的應用程式可以用 Java 或 Scala 編寫，並且可以使用 API 呼叫或 SQL 查詢來查詢資料。

如需如何使用資料表 API 的詳細資訊，請參閱 [檢閱資料表 API 元件](#)。

建立 Managed Service for Apache Flink 應用程式

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

- [使用執行期屬性](#)：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。
- [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 Managed Service for Apache Flink 中記錄和監控](#)：應用程式如何將事件記錄到 CloudWatch Logs。
- [實作應用程式擴展](#)：應用程式如何佈建運算資源。

您可以使用主控台或 AWS CLI 建立 Managed Service for Apache Flink 應用程式。若要開始建立 Managed Service for Apache Flink 應用程式，請參閱[教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#)。

建立 Managed Service for Apache Flink 應用程式

本主題包含建立 Managed Service for Apache Flink 應用程式的相關資訊。

本主題包含下列章節：

- [建置 Managed Service for Apache Flink 應用程式程式碼](#)
- [建立 Managed Service for Apache Flink 應用程式](#)
- [使用客戶受管金鑰](#)
- [啟動 Managed Service for Apache Flink 應用程式](#)
- [驗證 Managed Service for Apache Flink 應用程式](#)
- [為您的 Managed Service for Apache Flink 應用程式啟用系統復原](#)

建置 Managed Service for Apache Flink 應用程式程式碼

本節說明您用來為 Managed Service for Apache Flink 應用程式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[針對 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

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

應用程式現在可以使用任何 Scala 版本的 Java API。您必須將您選擇的 Scala 標準程式庫綁定到 Scala 應用程式。

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[將 Apache Beam 與 Managed Service for Apache Flink 應用程式搭配使用](#)。

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 Apache Flink 2.2.0，請提供下列項目：

```
mvn package -Dflink.version=2.2.0
```

如需使用舊版 Apache Flink 建置應用程式，請參閱[舊版](#)。

建立 Managed Service for Apache Flink 應用程式

建置應用程式程式碼之後，請執行下列動作來建立 Managed Service for Apache Flink (Amazon MSF) 應用程式：

- 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[教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#)課程。
- 建立 Managed Service for Apache Flink 應用程式：使用下列其中一種方法來建立 Amazon MSF 應用程式：

Note

Amazon MSF 預設會使用 加密您的應用程式 AWS 擁有的金鑰。您也可以使用 AWS KMS 客戶受管金鑰 (CMKs) 來自行建立、擁有和管理金鑰，以建立新的應用程式。如需 CMKs 的詳細資訊，請參閱[Amazon Managed Service for Apache Flink 中的金鑰管理](#)。

- 使用 AWS 主控台建立 Amazon MSF 應用程式：您可以使用 AWS 主控台建立和設定應用程式。

當您使用主控台建立應用程式時，也會為您建立應用程式的相依資源 (例如 CloudWatch Logs 串流、IAM 角色和 IAM 政策)。

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

如需如何使用 主控台建立應用程式的教學課程，請參閱教學[教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#)課程。

- 使用 CLI AWS 建立 Amazon MSF 應用程式：您可以使用 CLI AWS 建立和設定應用程式。

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

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

Note

您可以變更現有應用程式的 `RuntimeEnvironment`。若要了解作法，請參閱[針對 Apache Flink 使用就地版本升級](#)。

使用客戶受管金鑰

在 Amazon MSF 中，客戶受管金鑰 (CMKs) 是一種功能，您可以使用您在 AWS Key Management Service () 上建立、擁有和管理的金鑰來加密應用程式的資料AWS KMS。對於 Amazon MSF 應用程式，這表示受 Flink [檢查點](#)或[快照](#)約束的所有資料都會使用您為該應用程式定義的 CMK 加密。

若要搭配應用程式使用 CMK，您必須先[建立新的應用程式](#)，然後套用 CMK。如需使用 CMKs 的詳細資訊，請參閱 [Amazon Managed Service for Apache Flink 中的金鑰管理](#)。

啟動 Managed Service for Apache Flink 應用程式

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 AWS 主控台啟動 Managed Service for Apache Flink 應用程式：您可以在 AWS 主控台的應用程式頁面上選擇執行，以執行應用程式。
- 使用 AWS API 啟動 Managed Service for Apache Flink 應用程式：您可以使用 [StartApplication](#) 動作執行應用程式。

驗證 Managed Service for Apache Flink 應用程式

您可以驗證應用程式是否正常運作，方式如下：

- 使用 CloudWatch Logs：您可以使用 CloudWatch Logs 和 CloudWatch Logs Insights 來驗證您的應用程式是否在正常執行。如需將 CloudWatch Logs 與 Managed Service for Apache Flink 搭配使用的相關資訊，請參閱[在 Amazon Managed Service for Apache Flink 中記錄和監控](#)。
- 使用 CloudWatch 指標：您可以使用 CloudWatch 指標來監控應用程式的活動，或應用程式用於輸入或輸出的資源中的活動（例如 Kinesis 串流、Firehose 串流或 Amazon S3 儲存貯體）。如需 CloudWatch 指標的詳細資訊，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 Managed Service for Apache Flink 應用程式啟用系統復原

透過系統復原功能，您可以在 Amazon Managed Service for Apache Flink 上實現更高可用性的執行中 Apache Flink 應用程式。選擇此組態可讓服務在或等動作autoscaling執行到程式碼UpdateApplication或組態錯誤時，自動將應用程式還原至先前執行的版本。

Note

若要使用系統復原功能，您必須更新應用程式以選擇加入。根據預設，現有的應用程式不會自動使用系統復原。

運作方式

當您啟動應用程式操作時，例如更新或擴展動作，Amazon Managed Service for Apache Flink 會先嘗試執行該操作。如果偵測到阻止操作成功的問題，例如程式碼錯誤或許可不足，服務會自動啟動RollbackApplication操作。

轉返會嘗試將應用程式還原至先前成功執行的版本，以及相關聯的應用程式狀態。如果轉返成功，您的應用程式會使用先前的版本，以最短的停機時間繼續處理資料。如果自動轉返也失敗，Amazon Managed Service for Apache Flink 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操作。

您必須選擇加入，才能使用自動系統復原。從現在開始，您可以使用 主控台或 API 來啟用應用程式的所有操作。

UpdateApplication 動作的下列範例請求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有利之處：

- 應用程式更新：如果您使用透過主要方法初始化 Flink 任務時出現錯誤的新程式碼來更新應用程式，則自動復原允許還原先前的工作版本。系統復原有幫助的其他更新案例包括：
 - 如果您的應用程式已更新為以高於 [maxParallelism](#) 的平行處理執行。
 - 如果您的應用程式已更新為在啟動 Flink 任務期間導致失敗的 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，系統復原可讓您自動還原至先前的 Flink 版本。
- AutoScaling：當應用程式擴展，但由於快照和 Flink 任務圖表之間的運算子不相符，從儲存點遇到還原問題時。

使用操作 APIs 進行系統復原

為了提供更好的可見性，Amazon Managed Service for Apache Flink 有兩個與應用程式操作相關的 APIs，可協助您追蹤故障和相關的系統復原。

ListApplicationOperations

此 API 會依反向時間順序列出應用程式上執行的所有操作，包括 RollbackApplication、UpdateApplication Maintenance 和其他操作。ListApplicationOperations 動作的下列範例請求會列出應用程式的前 10 個應用程式操作：

```
{
```

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

的下列範例請求ListApplicationOperations有助於篩選應用程式上先前更新的清單：

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

DescribeApplicationOperation

此 API 提供 列出之特定操作的詳細資訊ListApplicationOperations，包括故障原因，如適用。DescribeApplicationOperation 動作的下列範例請求會列出特定應用程式操作的詳細資訊：

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

如需故障診斷資訊，請參閱[系統復原最佳實務](#)。

執行 Managed Service for Apache Flink 應用程式

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

執行 Managed Service for Apache Flink 應用程式時，該服務會建立 Apache Flink 作業。Apache Flink 作業是指 Managed Service 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 Apache Flink 任務的相關資訊，請參閱 Apache Flink 文件中的[任務和排程](#)。

識別應用程式和任務狀態

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：新的或已停止的應用程式處於 READY 狀態，直到您執行它為止。
 - RUNNING：已成功啟動的應用程式處於 RUNNING 狀態。
 - 暫時性應用程式狀態：處於這些狀態的應用程式通常處於轉換至其他狀態的過程中。如果應用程式已保持暫時狀態一段時間，您可以使用 [StopApplication](#) 動作停止應用程式，並將 Force 參數設定為 true。這些狀態包括下列項目：
 - STARTING：在 [StartApplication](#) 操作之後發生。應用程式正在從狀態 READY 轉換為 RUNNING 狀態。
 - STOPPING：在 [StopApplication](#) 動作之後發生。應用程式正在從狀態 RUNNING 轉換為 READY 狀態。
 - DELETING：在 [DeleteApplication](#) 動作之後發生。正在刪除應用程式。
 - UPDATING：在 [UpdateApplication](#) 動作之後發生。應用程式正在更新，並會轉換回 RUNNING 或 READY 狀態。
 - AUTOSCALING：應用程式的 [ParallelismConfiguration AutoScalingEnabled](#) 屬性設定為 true，而且服務正在增加應用程式的平行處理層級。當應用程式處於此狀態時，您唯一可以使用的有效 API 動作是 [StopApplication](#) 動作，且 Force 參數設定為 true。如需自動擴展的相關資訊，請參閱[在 Managed Service for Apache Flink 中使用自動擴展](#)。
 - FORCE_STOPPING：在呼叫 [StopApplication](#) 動作且 Force 參數設定為 true 之後發生。正在停止應用程式。應用程式正在從 STARTING、UPDATING、STOPPING 或 AUTOSCALING 狀態轉換為 READY 狀態。
 - ROLLING_BACK：在 [RollbackApplication](#) 動作之後發生。應用程式正在復原至先前的版本。應用程式正在從 UPDATING 或 AUTOSCALING 狀態轉換為 RUNNING 狀態。
 - 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 Managed Service for Apache Flink 的維護任務](#)。

您可以使用主控台或 [DescribeApplication](#) 動作來檢查應用程式的狀態。

- 作業狀態：當應用程式處於 RUNNING 狀態時，作業的狀態會描述其目前的執行期。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 CloudWatch 日誌看是否有狀態變更，以檢查作業的狀態。

執行批次工作負載

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

檢閱 Managed Service for Apache Flink 應用程式資源

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

Managed Service for Apache Flink 應用程式資源

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

一個 KPU 代表下列系統資源：

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

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

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

如需應用程式平行處理的詳細資訊，請參閱[實作應用程式擴展](#)。

Apache Flink 應用程式資源

Apache Flink 環境會使用稱為任務空位的單元為應用程式配置資源。Managed Service for Apache Flink 為應用程式配置資源時，會將一或多個 Apache Flink 任務空位指派給單一 KPU。指派給單一 KPU 的空位數量等於應用程式的 ParallelismPerKPU 設定。如需任務槽的詳細資訊，請參閱 Apache Flink 文件中的[任務排程](#)。

運算子平行處理

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。

Managed Service for Apache Flink 中的每秒帳單

Managed Service for Apache Flink 現在以一秒遞增計費。每個應用程式最低收費為 10 分鐘。每秒計費適用於新啟動或已執行的應用程式。本節說明 Managed Service for Apache Flink 如何計量您的用量並向您收費。若要進一步了解 Managed Service for Apache Flink 定價，請參閱 [Amazon Managed Service for Apache Flink 定價](#)。

運作方式

Managed Service for Apache Flink 會在支援的 中以一秒增量計費的 Kinesis 處理單元 (KPU) 期間和數量向您收費 AWS 區域。單一 KPU 包含 1vCPU 運算和 4 GB 記憶體。系統會根據用於執行應用程式的 KPUs 數目，向您收取每小時費率的費用。

例如，執行 20 分鐘和 10 秒的應用程式會收取 20 分鐘和 10 秒的費用，再乘以其使用的資源。執行 5 分鐘的應用程式將至少收取 10 分鐘的費用，再乘以其使用的資源。

Managed Service for Apache Flink 以小時為單位說明用量。例如，15 分鐘對應至 0.25 小時。

對於 Apache Flink 應用程式，您需要為每個應用程式支付單一額外 KPU，用於協調。應用程式也會針對執行中的儲存體和耐用的備份付費。執行中的應用程式儲存用於 Managed Service for Apache Flink 中的有狀態處理功能，並按 GB/月收費。持久性備份是選用的，可為應用程式提供point-in-time復原，每月每 GB 收費。

在串流模式下，Managed Service for Apache Flink 會隨著記憶體和運算的需求波動，自動擴展串流處理應用程式所需的 KPU 數量。您可以選擇使用所需的 KPU 數量來佈建應用程式。

AWS 區域 可用性

Note

目前，每秒計費不適用於下列區域：AWS GovCloud（美國東部）、AWS GovCloud（美國西部）、中國（北京）和中國（寧夏）。

每秒計費提供如下 AWS 區域：

- 美國東部 (維吉尼亞北部) – us-east-1
- 美國東部 (俄亥俄) - us-east-2
- 美國西部 (加利佛尼亞北部) – us-west-1
- 美國西部 (奧勒岡) - us-west-2
- 非洲 (開普敦) – af-south-1
- 亞太區域 (香港) – ap-east-1
- 亞太區域 (海德拉巴) - ap-south-1
- 亞太區域 (雅加達) – ap-southeast-3
- 亞太區域 (墨爾本) - ap-southeast-4
- 亞太區域 (孟買) – ap-south-1
- 亞太區域 (大阪) - ap-northeast-3
- 亞太區域 (首爾) – ap-northeast-2

- 亞太區域 (新加坡) – ap-southeast-1
- 亞太區域 (雪梨) – ap-southeast-2
- 亞太區域 (東京) – ap-northeast-1
- 加拿大 (中部) – ca-central-1
- 加拿大西部 (卡加利) - ca-west-1
- 歐洲 (法蘭克福) – eu-central-1
- 歐洲 (愛爾蘭) – eu-west-1
- 歐洲 (倫敦) – eu-west-2
- 歐洲 (米蘭) – eu-south-1
- 歐洲 (巴黎) – eu-west-3
- 歐洲 (西班牙) – eu-south-2
- 歐洲 (斯德哥爾摩) – eu-north-1
- 歐洲 (蘇黎世) – eu-central-2
- 以色列 (特拉維夫) - il-central-1
- 中東 (巴林) – me-south-1
- 中東 (阿拉伯聯合大公國) – me-central-1
- 南美洲 (聖保羅) – sa-east-1

定價範例

您可以在 Managed Service for Apache Flink 定價頁面上找到定價範例。如需詳細資訊，請參閱 [Amazon Managed Service for Apache Flink 定價](#)。以下是每個的成本用量報告圖例的進一步範例。

長時間執行、繁重的工作負載

您是大型影片串流服務，您想要根據使用者的互動建立即時影片建議。您可以在 Managed Service for Apache Flink 中使用 Apache Flink 應用程式，持續從多個 Kinesis 資料串流擷取使用者互動事件，並在輸出至下游系統之前即時處理事件。使用者互動事件會使用多個運算子進行轉換。這包括依事件類型分割資料、使用其他中繼資料擴充資料、依時間戳記排序資料，以及在交付前緩衝資料 5 分鐘。應用程式有許多需要大量運算且可平行化的轉換步驟。您的 Flink 應用程式已設定為使用 20 KPIUs 執行，以容納工作負載。您的應用程式每天使用 1 GB 的耐用應用程式備份。每月 Managed Service for Apache Flink 費用的計算方式如下：

每月費用

美國東部（維吉尼亞北部）區域的價格為每 KPU 小時 0.11 美元。Managed Service for Apache Flink 為每個 KPU 配置 50 GB 的執行中應用程式儲存體，每月收取 0.10 USD。

- 每月 KPU 費用：24 小時 * 30 天 * (20 個 KPUs + 串流應用程式的額外 1 個 KPU) * \$0.11/小時 = \$1,584.00
- 每月執行的應用程式儲存費用：30 天 * 20 KPUs * 50 GB/KPUs * \$0.10/GB-月 = \$100.00
- 每月持久性應用程式儲存費用：30 天 * 1 GB * 0.023/GB-月 = 0.03 美元
- 總費用：\$1,584.00 + \$100 + \$0.03 = \$1,684.03

帳單與成本管理主控台上當月 Managed Service for Apache Flink 的成本用量報告

Kinesis Analytics

- USD 1,684.03 - 美國東部（維吉尼亞北部）
- Amazon Kinesis Analytics CreateSnapshot
 - 每 GB 每月 \$0.023 的耐用應用程式備份
 - 每月 1 GB - 0.03 USD
- Amazon Kinesis Analytics StartApplication
 - 每 GB 每月執行應用程式儲存 0.10 美元
 - 每月 1,000 GB - 100 USD
- Apache Flink 應用程式每 Kinesis 處理單位小時 0.11 美元
 - 15,120 KPU 小時 - 1,584 USD

每天執行約 15 分鐘的批次工作負載

您可以在 Managed Service for Apache Flink 中使用 Apache Flink 應用程式，以批次模式轉換 Amazon Simple Storage Service (Amazon S3) 中的日誌資料。日誌資料會使用多個運算子進行轉換。這包括將結構描述套用至不同的日誌事件、依事件類型分割資料，以及依時間戳記排序資料。應用程式有許多轉換步驟，但沒有任何步驟具有運算密集性。此應用程式會在 30 天內，每天以每秒 2,000 筆記錄擷取資料 15 分鐘。您不會建立任何耐用的應用程式備份。每月 Managed Service for Apache Flink 費用的計算方式如下：

每月費用

美國東部（維吉尼亞北部）區域的價格為每 KPU 小時 0.11 美元。Managed Service for Apache Flink 為每個 KPU 配置 50 GB 的執行中應用程式儲存體，每月收取 0.10 USD。

- 批次工作負載：在每天 15 分鐘內，Managed Service for Apache Flink 應用程式正在處理 2,000 筆記錄/秒，這需要 2KPU。30 天/月 * 15 分鐘/天 = 450 分鐘/月
- 每月 KPU 費用：每月 450 分鐘 * (2KPU 串流應用程式的額外 1 個 KPU) * \$0.11/小時 = \$2.48
- 每月執行的應用程式儲存費用：每月 450 分鐘 * 2 KPU * 50 GB/KPU * 每月 0.10 USD = 0.11 USD
- 費用總計：\$2.48 + 0.11 = \$2.59

帳單與成本管理主控台上當月 Managed Service for Apache Flink 的成本用量報告

Kinesis Analytics

- USD 2.59 - 美國東部（維吉尼亞北部）
- Amazon Kinesis Analytics StartApplication
 - 每 GB 每月執行應用程式備份 0.10 美元
 - 每月 1.042 GB - 0.11 USD
- Apache Flink 應用程式每 Kinesis 處理單位小時 0.11 美元
 - 22.5 KPU-Hour - USD 2.48

測試應用程式，可在同一小時內停止並持續啟動，並吸引多個最低費用

您是一個大型電子商務平台，每天處理數百萬筆交易。您想要開發即時詐騙偵測。您可以在 Managed Service for Apache Flink 中使用 Apache Flink 應用程式，從 Kinesis Data Streams 擷取交易事件，並使用不同的轉換步驟即時處理事件。這包括使用滑動視窗彙總事件、依事件類型分割事件，以及針對不同的事件類型套用特定偵測規則。在開發期間，您會多次啟動和停止應用程式，以測試和偵錯行為。有時候，您的應用程式只會執行幾分鐘。當您使用 4 個 KPU 測試應用程式時，有一小時的時間，您的應用程式不會使用任何耐用的應用程式備份：

- 在上午 10：05，您會啟動應用程式，該應用程式會在上午 10：35 停止之前執行 30 分鐘。
- 在上午 10：40，您會再次啟動應用程式，該應用程式會在上午 10：45 停止之前執行 5 分鐘。
- 上午 10：50，您會再次啟動應用程式，並在上午 10：52 停止之前執行 2 分鐘。

每次應用程式啟動時，Managed Service for Apache Flink 都會收取至少 10 分鐘的使用費。您應用程式的每月 Managed Service for Apache Flink 用量計算方式如下：

- 第一次您的應用程式啟動和停止時：使用 30 分鐘

- 第二次您的應用程式啟動和停止時：使用 10 分鐘（您的應用程式執行 5 分鐘，四捨五入至最低 10 分鐘費用）
- 您的應用程式第三次啟動和停止：使用 10 分鐘（您的應用程式執行 2 分鐘，四捨五入至最低 10 分鐘費用）

您的應用程式總共需支付 50 分鐘的使用費。如果應用程式執行當月沒有其他時間，則會計算每月 Managed Service for Apache Flink 費用，如下所示：

每月費用

美國東部（維吉尼亞北部）區域的價格為每 KPU 小時 0.11 美元。Managed Service for Apache Flink 為每個 KPU 配置 50 GB 的執行中應用程式儲存體，每月收取 0.10 USD。

- 每月 KPU 費用：50 分鐘 * (4KPU + 串流應用程式的 1 個額外 KPU) * 0.11 美元/小時 = 0.46 美元（四捨五入至最接近的一元）
- 每月執行的應用程式儲存費用：50 分鐘 * 4 KPU * 50 GB/KPU * 每月 0.10 USD = 0.03 USD（四捨五入至最接近的一元）
- 費用總計：\$0.46 + 0.03 = \$0.49

帳單與成本管理主控台上當月 Managed Service for Apache Flink 的成本用量報告

Kinesis Analytics

- USD 0.49 - 美國東部（維吉尼亞北部）
- Amazon Kinesis Analytics StartApplication
 - 每 GB 每月執行應用程式儲存 0.10 美元
 - 每月 0.232 GB - 0.03 USD
 - Apache Flink 應用程式每 Kinesis 處理單位小時 0.11 美元
 - 4.167 KPU-Hour - USD 0.46

檢閱 DataStream API 元件

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

本節說明移動、轉換和追蹤資料的不同元件：

- [使用連接器透過 DataStream API 在 Managed Service for Apache Flink 中移動資料](#)：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 Managed Service for Apache Flink 中的運算子搭配 DataStream API 轉換資料](#)：這些元件可以轉換或分組應用程式中的資料元素。
- [使用 DataStream API 追蹤 Managed Service for Apache Flink 中的事件](#)：本主題說明 Managed Service for Apache Flink 如何在使用 DataStream API 時追蹤事件。

使用連接器透過 DataStream API 在 Managed Service for Apache Flink 中移動資料

在 Amazon Managed Service for Apache Flink DataStream API 中，連接器是可將資料移入和移出 Managed Service for Apache Flink 應用程式的軟體元件。連接器是靈活的整合，可讓您從檔案和目錄讀取。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：

- [新增串流資料來源](#)：從 Kinesis 資料串流、檔案或其他資料來源向應用程式提供資料。
- [使用接收器寫入資料](#)：從您的應用程式將資料傳送至 Kinesis 資料串流、Firehose 串流或其他資料目的地。
- [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。

可用的連接器

Apache Flink 架構包含用於存取各種來源之資料的連接器。如需 Apache Flink 架構中可用連接器的相關資訊，請參閱 [Apache Flink 文件](#) 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，並且想要在中東（阿拉伯聯合大公國）、亞太區域（海德拉巴）、以色列（特拉維夫）、歐洲（蘇黎世）、中東（阿拉伯聯合大公國）、亞太區域（墨爾本）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，或升級至 Flink 1.18。

Apache Flink 連接器存放在自己的開放原始碼儲存庫中。如果您要升級至 1.18 版或更新版本，則必須更新您的相依性。若要存取 Apache Flink AWS 連接器的儲存庫，請參閱 [flink-connector-aws](#)。

之前的 Kinesis 來

源 `org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer` 已停止，未來可能會隨著 Flink 版本而移除。請改用 [Kinesis 來源](#)。

`FlinkKinesisConsumer` 和 `KinesisStreamsSource` 之間沒有狀態相容性。如需詳細資訊，請參閱 Apache Flink 文件中的 [將現有任務遷移至新的 Kinesis 串流來源](#)。

以下是建議的準則：

連接器升級

Flink 版本	使用的連接器	Resolution
1.19、1.20	Kinesis 來源	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 Kinesis Data Streams 來源連接器。這必須是任何 5.0.0 版或更新版本。如需詳細資訊，請參閱 Amazon Kinesis Data Streams Connector 。
1.19、1.20	Kinesis 接收器	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 Kinesis Data Streams 接收器連接器。這必須是任何 5.0.0 版或更新版本。如需詳細資訊，請參閱 Kinesis Streams Sink 。
1.19、1.20	DynamoDB 串流來源	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 DynamoDB Streams 來源連接器。這必須是任何 5.0.0 版或更新版本。如需詳細資訊，請參閱 Amazon DynamoDB 連接器 。

Flink 版本	使用的連接器	Resolution
1.19、1.20	DynamoDB 接收器	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 DynamoDB 接收器連接器。這必須是任何 5.0.0 版或更新版本。如需詳細資訊，請參閱 Amazon DynamoDB 連接器 。
1.19、1.20	Amazon SQS 接收器	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 Amazon SQS 接收器連接器。這必須是任何 5.0.0 版或更新版本。如需詳細資訊，請參閱 Amazon SQS Sink 。
1.19、1.20	Amazon Managed Service for Prometheus Sink	升級至 Managed Service for Apache Flink 1.19 和 1.20 版時，請確定您使用的是最新的 Amazon Managed Service for Prometheus 接收器連接器。這必須是任何 1.0.0 版或更新版本。如需詳細資訊，請參閱 Prometheus Sink 。

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

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

使用 Kinesis 資料串流

KinesisStreamsSource 會從 Amazon Kinesis 資料串流提供串流資料到您的應用程式。

建立 KinesisStreamsSource

以下程式碼範例示範如何建立 KinesisStreamsSource：

```
// Configure the KinesisStreamsSource
Configuration sourceConfig = new Configuration();
sourceConfig.set(KinesisSourceConfigOptions.STREAM_INITIAL_POSITION,
    KinesisSourceConfigOptions.InitialPosition.TRIM_HORIZON); // This is optional, by
    default connector will read from LATEST

// Create a new KinesisStreamsSource to read from specified Kinesis Stream.
KinesisStreamsSource<String> kdsSource =
    KinesisStreamsSource.<String>builder()
        .setStreamArn("arn:aws:kinesis:us-east-1:123456789012:stream/test-
stream")
        .setSourceConfig(sourceConfig)
        .setDeserializationSchema(new SimpleStringSchema())

        .setKinesisShardAssigner(ShardAssignerFactory.uniformShardAssigner()) // This is
        optional, by default uniformShardAssigner will be used.
        .build();
```

如需使用 的詳細資訊 KinesisStreamsSource，請參閱 Apache Flink 文件中的 [Amazon Kinesis Data Streams Connector](#)，以及 [Github 上的公有 KinesisConnectors 範例](#)。

建立 KinesisStreamsSource 使用 EFO 取用者的

現在 KinesisStreamsSource 支援 [增強型廣發 \(EFO\)](#)。

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

如需搭配 Kinesis 取用者使用 EFO 的詳細資訊，請參閱 [FLIP-128：增強的 AWS Kinesis 取用者扇出](#)。

您可以在 Kinesis 取用者上設定下列參數來啟用 EFO 取用者：

- **READER_TYPE**：將此參數設定為 EFO，讓您的應用程式使用 EFO 取用者存取 Kinesis Data Stream 資料。

- `EFO_CONSUMER_NAME`：將此參數設定為字串值，確保在此串流的取用者中保持唯一。在相同的 Kinesis 資料串流中重複使用取用者名稱，將導致先前使用該名稱的使用者遭到終止。

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

```
sourceConfig.set(KinesisSourceConfigOptions.READER_TYPE,
    KinesisSourceConfigOptions.ReaderType.EFO);
sourceConfig.set(KinesisSourceConfigOptions.EFO_CONSUMER_NAME, "my-flink-efo-
consumer");
```

如需使用 EFO 消費者的 Managed Service for Apache Flink 應用程式範例，請參閱 [Github 上的公有 Kinesis Connectors 範例](#)。

使用 Amazon MSK

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

建立 `KafkaSource`

以下程式碼範例示範如何建立 `KafkaSource`：

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

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

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

在 Managed Service for Apache Flink 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](#) 連接器來寫入外部系統，包括 Kinesis Data Streams 和 DynamoDB 等 AWS 服務。

Apache Flink 也為檔案和通訊端提供接收器，而且您可以實作自訂接收器。在數個支援的接收器中，經常使用下列項目：

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 Managed Service for Apache Flink 中的 DataStream API](#)。

使用 Apache Kafka 和 Amazon Managed Streaming for Apache Kafka (MSK)

[Apache Flink Kafka 連接器](#) 提供廣泛的支援，可將資料發佈至 Apache Kafka 和 Amazon MSK，包括恰好一次的保證。若要了解如何寫入 Kafka，請參閱 Apache Flink 文件中的 [Kafka Connectors 範例](#)。

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 是可靠、可擴展的 Apache Flink 接收器，用於使用 [Firehose](#) 服務存放應用程式輸出。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和設定 `FlinkKinesisFirehoseProducer` 以及將資料從 Apache Flink 資料串流傳送至 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");
}
```

如需如何使用 Firehose 接收器的完整教學課程，請參閱 [the section called “Firehose 接收器”](#)。

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

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

如需詳細資訊，請參閱 Apache Flink 文件中的 [非同步 I/O](#)。

使用 Managed Service for Apache Flink 中的運算子搭配 DataStream API 轉換資料

若要在 Managed Service for Apache Flink 中轉換傳入的資料，請使用 Apache Flink 運算子。Apache Flink 運算子可將一或多個資料串流轉換為新的資料串流。新的資料串流包含來自原始資料串流的修改資料。Apache Flink 提供了超過 25 個預先建置的串流處理運算子。如需詳細資訊，請參閱 Apache Flink 文件中的 [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 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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

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](#)。

檢閱資料表 API 元件

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

本節包含下列主題：

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

資料表 API 連接器

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

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

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

資料表 API 來源

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

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

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

如需資料表來源的詳細資訊，請參閱 Apache Flink 文件中的[資料表和 SQL 連接器](#)。

資料表 API 接收器

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

下列程式碼範例示範如何將資料表資料寫入 Amazon S3 接收器：

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

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

您可以使用 format 參數來控制 Managed Service for Apache Flink 用來將輸出寫入接收器的格式。如需格式的相關資訊，請參閱 Apache Flink 文件中的[支援的連接器](#)。

使用者定義的來源和接收器

您可以使用現有的 Apache Kafka 連接器與其他 AWS 服務 (例如 Amazon MSK 和 Amazon S3) 之間相互傳送資料。若要與其他資料來源和目的地互動，您可以定義自己的來源和接收器。如需詳細資訊，請參閱 Apache Flink 文件中的[使用者定義來源和接收器](#)。

資料表 API 時間屬性

資料串流中的每個記錄都有數個時間戳記，用來定義與記錄相關的事件發生的時間：

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

當 Apache Flink Table API 根據記錄時間建立視窗時，您可以使用 `setStreamTimeCharacteristic` 方法定義其使用的時間戳記。

如需搭配資料表 API 使用時間戳記的詳細資訊，請參閱 Apache Flink 文件中的[時間屬性](#)和[及時串流處理](#)。

使用 Python 搭配 Managed Service for Apache Flink

Note

如果您在使用 Apple 晶片的新 Mac 上開發 Python Flink 應用程式，可能會遇到與 PyFlink 1.15 的 Python 相依性的一些[已知問題](#)。在這種情況下，我們建議在 Docker 中執行 Python 解譯器。如需逐步指示，請參閱[在 Apple Silicon Mac 上進行 PyFlink 1.15 開發](#)。

Apache Flink 2.2 版包含使用 Python 3.12 版建立應用程式的支援；Python 3.8 版的支援已移除。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。

在高層級上，Python 資料表 API 是 Java 資料表 API 周圍的一項包裝函式。如需有關 Python 資料表 API 的資訊，請參閱 Apache Flink 文件中的[資料表 API 教學課程](#)。

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

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

執行下列動作以使用 Python 資料表 API：

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

若要開始在 Managed Service for Apache Flink 中使用 Python 資料表 API，請參閱 [Amazon Managed Service for Apache Flink for Python 入門](#)。

讀取和寫入串流資料

若要讀取和寫入串流資料，請在資料表環境中執行 SQL 查詢。

建立資料表

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

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



```
'aws.region' = '{2}',  
'scan.stream.initpos' = '{3}',  
'sink.partitioner-field-delimiter' = ';',  
'sink.producer.collection-max-count' = '100',  
'format' = 'json',  
'json.timestamp-format.standard' = 'ISO-8601'  
) """.format(table_name, stream_name, region, stream_initpos)
```

讀取串流資料

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

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

寫入串流資料

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

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

讀取執行期屬性

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從上一個範例傳回的屬性群組讀取名為 INPUT_STREAM_KEY 的屬性：

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

建立應用程式的程式碼套件

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

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

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

Note

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

建立 Managed Service for Apache Flink Python 應用程式

指定您的程式碼檔案

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

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

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

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

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"
        }
      }
    ]
  },
}
```

監控 Managed Service for Apache Flink Python 應用程式

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

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

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

```
import logging

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

- 應用程式擲出的錯誤訊息。

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

下列範例示範從 Python 程式碼擲回例外狀況的日誌項目：

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

Note

由於效能問題，建議只在應用程式開發期間使用自訂日誌訊息。

使用 CloudWatch Insights 查詢日誌

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

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

在 Managed Service for Apache Flink 中使用執行期屬性

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期屬性](#)
- [使用 CLI 管理執行期屬性](#)
- [在 Managed Service for Apache Flink 應用程式中存取執行期屬性](#)

使用主控台管理執行期屬性

您可以使用 從 Managed Service for Apache Flink 應用程式新增、更新或移除執行期屬性 AWS 管理主控台。

Note

如果您使用的是舊版支援的 Apache Flink，而且想要將現有應用程式升級至 Apache Flink 1.19.1，則可以使用就地 Apache Flink 版本升級來執行此操作。透過就地版本升級，您可以保留跨 Apache Flink 版本的單一 ARN 的應用程式可追蹤性，包括快照、日誌、指標、標籤、Flink 組態等。您可以在 RUNNING 和 READY 狀態使用此功能。如需詳細資訊，請參閱[針對 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
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 CLI 管理執行期屬性

您可以使用 [AWS CLI](#) 新增、更新或移除執行期屬性。

本節包含針對設定應用程式執行期屬性之 API 動作的範例請求。如需如何使用 JSON 檔案作為 API 動作輸入的相關資訊，請參閱 [Managed Service for Apache Flink API 範例程式碼](#)。

Note

使用您的帳戶 ID 取代以下範例中的範例帳戶 ID (*012345678901*)。

在建立應用程式時新增執行期屬性

[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期屬性

[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。

移除執行期屬性

[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 Managed Service for Apache Flink 應用程式中存取執行期屬性

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 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

將 Apache Flink 連接器與 Managed Service for Apache Flink 搭配使用

Apache Flink 連接器是將資料移入和移出 Amazon Managed Service for Apache Flink 應用程式的軟體元件。連接器是靈活的整合，可讓您從檔案和目錄讀取。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應用程式。
- 接收器：從您的應用程式將資料傳送至 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：提供非同步存取資料來源，例如資料庫，以擴充串流。

Apache Flink 連接器存放在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、Table 或 SQL API。

Amazon Managed Service for Apache Flink 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來建置自訂接收器。如需詳細資訊，請參閱 Apache Flink 文件中的 [一般非同步基礎接收器](#)。

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

Flink 2.2 的連接器

升級至 Flink 2.2 時，您需要將連接器相依性更新為與 Flink 2.x 執行時間相容的版本。Flink 連接器獨立於 Flink 執行期發行，而且並非所有連接器都有 Flink 2.x 相容版本。下表摘要說明截至撰寫時 Amazon Managed Service for Apache Flink 中常用連接器的可用性：

Flink 2.2 的連接器

連接器	Flink 2.0+ 版本	備註
Apache Kafka	flink-connector-kafka 4.0.0-2.0	建議用於 Flink 2.2
Kinesis Data Streams（來源）	flink-connector-aws-kinesis-streams 6.0.0-2.0	建議用於 Flink 2.2
Kinesis Data Streams（接收器）	flink-connector-aws-kinesis-streams 6.0.0-2.0	建議用於 Flink 2.2
FileSystem (S3、HDFS)	與 Flink 綁定	內建於 Flink 分佈中 — 一律可用
JDBC	尚未針對 2.x 發行	沒有可用的 Flink 2.x 相容版本
OpenSearch	尚未針對 2.x 發行	沒有可用的 Flink 2.x 相容版本
Elasticsearch	尚未針對 2.x 發行	考慮遷移至 OpenSearch 連接器
Amazon Managed Service for Prometheus	尚未針對 2.x 發行	寫入時沒有 Flink 2.x 相容版本

如果您的應用程式依賴的連接器尚未有 Flink 2.2 版本，您有兩個選項：等待連接器發行相容的版本，或評估是否可以用替代版本取代它（例如，使用 JDBC 目錄或自訂接收器）。

已知問題

- 當 Kinesis 串流進行重新分片時，連接器 v5.0.0 和 v6.0.0 中引入 `KinesisStreamsSource` 搭配 EFO（增強廣播/SubscribeToShard）路徑使用的應用程式可能會失敗。這是社群中的已知問題。如需詳細資訊，請參閱 [FLINK-37648](#)。
- 如果 Flink 應用程式處於背壓狀態，則連接器 v5.0.0 和 v6.0.0 中引入 `KinesisStreamsSource` 搭配 EFO（增強廣播/SubscribeToShard）路徑使用的應用程式 `KinesisStreamsSink` 可能會遇到死結，導致一個或多個 TaskManagers 中的資料處理完全停止。需要強制停止操作和啟動應用程式操作才能復原應用程式。這是社群中已知問題的子案例：[FLINK-34071](#)。

舊版 Flink 的連接器

舊版 Flink 的連接器

連接器	Flink 1.15 版	Flink 1.18 版	Flink 1.19 版	Flink 1.20 版
Kinesis Data Stream - 來源 - DataStream 和資料表 API	flink-connector-kinesis, 1.15.4	flink-connector-kinesis, 4.3.0-1.18	flink-connector-kinesis, 5.0.0-1.19	flink-connector-kinesis, 5.0.0-1.20
Kinesis Data Stream - Sink - DataStream 和資料表 API	flink-connector-aws-kinesis-streams, 1.15.4	flink-connector-aws-kinesis-streams, 4.3.0-1.18	flink-connector-aws-kinesis-streams, 5.0.0-1.19	flink-connector-aws-kinesis-streams, 5.0.0-1.20
Kinesis Data Streams - 來源/接收器 - SQL	flink-sql-connector-kinesis, 1.15.4	flink-sql-connector-kinesis, 4.3.0-1.18	flink-sql-connector-kinesis, 5.0.0-1.19	flink-sql-connector-kinesis-streams, 5.0.0-1.20
Kafka - DataStream 和資料表 API	flink-connector-kafka, 1.15.4	flink-connector-kafka, 3.2.0-1.18	flink-connector-kafka, 3.3.0-1.19	flink-connector-kafka, 3.3.0-1.20

連接器	Flink 1.15 版	Flink 1.18 版	Flink 1.19 版	Flink 1.20 版
Kafka - SQL	flink-sql-connector-kafka , 1.15.4	flink-sql-connector-kafka , 3.2.0-1.18	flink-sql-connector-kafka , 3.3.0-1.19	flink-sql-connector-kafka , 3.3.0-1.20
Firehose - DataStream 和資料表 API	flink-connector-aws-kinesis-firehose , 1.15.4	flink-connector-aws-firehose , 4.3.0-1.18	flink-connector-aws-firehose , 5.0.0-1.19	flink-connector-aws-firehose , 5.0.0-1.20
Firehose - SQL	flink-sql-connector-aws-kinesis-firehose , 1.15.4	flink-sql-connector-aws-firehose , 4.3.0-1.18	flink-sql-connector-aws-firehose , 5.0.0-1.19	flink-sql-connector-aws-firehose , 5.0.0-1.20
DynamoDB - DataStream 和資料表 API	flink-connector-dynamodb , 3.0.0-1.15	flink-connector-dynamodb , 4.3.0-1.18	flink-connector-dynamodb , 5.0.0-1.19	flink-connector-dynamodb , 5.0.0-1.20
DynamoDB - SQL	flink-sql-connector-dynamodb , 3.0.0-1.15	flink-sql-connector-dynamodb , 4.3.0-1.18	flink-sql-connector-dynamodb , 5.0.0-1.19	flink-sql-connector-dynamodb , 5.0.0-1.20
OpenSearch - DataStream 和資料表 API	-	flink-connector-opensearch , 1.2.0-1.18	flink-connector-opensearch , 1.2.0-1.19	flink-connector-opensearch , 1.2.0-1.19
OpenSearch - SQL	-	flink-sql-connector-opensearch , 1.2.0-1.18	flink-sql-connector-opensearch , 1.2.0-1.19	flink-sql-connector-opensearch , 1.2.0-1.19

連接器	Flink 1.15 版	Flink 1.18 版	Flink 1.19 版	Flink 1.20 版
Amazon Managed Service for Prometheus DataStream	-	flink-sql-connector-opensearch , 1.2.0-1.18	flink-connector-prometheus , 1.0.0-1.19	flink-connector-prometheus , 1.0.0-1.20
Amazon SQS DataStream 和資料表 API	-	flink-sql-connector-opensearch , 1.2.0-1.18	flink-connector-sqs、5.0.0-1.19	flink-connector-sqs、5.0.0-1.20

若要進一步了解 Amazon Managed Service for Apache Flink 中的連接器，請參閱：

- [DataStream API 連接器](#)
- [資料表 API 連接器](#)

已知問題

Apache Flink 1.15 中的 Apache Kafka 連接器存在已知的開放原始碼 Apache Flink 問題。此問題已在更新版本的 Apache Flink 中解決。

如需詳細資訊，請參閱[the section called “已知問題”](#)。

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

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

如需 Apache Flink 應用程式中檢查點的詳細資訊，請參閱 Apache Flink 文件中的[檢查點](#)。

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

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

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

在 Managed Service for Apache Flink 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`：true
- `CheckpointInterval`：60000
- `MinPauseBetweenCheckpoints`：5000

如果 `ConfigurationType` 設定為 `DEFAULT`，即使使用或應用程式程式碼中的值設定為其他值 AWS Command Line Interface，也會使用上述值。

Note

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

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

檢閱檢查點 API 範例

本節包含針對設定應用程式檢查點之 API 動作的範例請求。如需如何使用 JSON 檔案作為 API 動作輸入的相關資訊，請參閱 [Managed Service for Apache Flink API 範例程式碼](#)。

設定新應用程式的檢查點

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

```
{
  "ApplicationName": "MyApplication",
  "RuntimeEnvironment": "FLINK-1_19",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "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:::amzn-s3-demo-bucket",
        "FileKey": "myflink.jar",
        "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    },
    "FlinkApplicationConfiguration": {
      "CheckpointConfiguration": {
        "CheckpointingEnabled": "false"
      }
    }
  }
}

```

設定現有應用程式的檢查點

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

```

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

```

停用現有應用程式的檢查點

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

```

{
  "ApplicationName": "MyApplication",
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "CheckpointConfigurationUpdate": {
        "CheckpointingEnabledUpdate": false,

```



```
        "CheckpointIntervalUpdate": 20000,  
        "ConfigurationTypeUpdate": "CUSTOM",  
        "MinPauseBetweenCheckpointsUpdate": 10000  
    }  
}  
}
```

使用快照管理應用程式備份

快照是 Apache Flink 儲存點的 Managed Service for Apache Flink 實作。快照是使用者或服務觸發、建立和管理的應用程式狀態備份。如需 Apache Flink 儲存點的相關資訊，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。經常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系統資源。

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

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

如需每個應用程式的快照數目限制，請參閱[Managed Service for Apache Flink 和 Studio 筆記本配額](#)。如果您的應用程式達到快照數目限制，則手動建立快照會失敗，並顯示 `LimitExceededException`。

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

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

本主題包含下列章節：

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

Note

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

Note

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

自動建立的快照具有下列特質：

- 快照由服務管理，但您可以使用 [ListApplicationSnapshots](#) 動作查看快照。自動建立的快照會根據您的快照限制計數。
- 如果您的應用程式超過快照限制，手動建立的快照將會失敗，但 Managed Service for Apache Flink 服務在應用程式更新、擴展或停止時仍會成功建立快照。您必須先使用 [DeleteApplicationSnapshot](#) 動作手動刪除快照，然後才能手動建立更多快照。

從包含不相容狀態資料的快照還原

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

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

從過時的快照還原應用程式時，您會看到下列行為：

- 新增運算子：如果新增了新的運算子，則儲存點沒有該新運算子的狀態資料。不會發生任何錯誤，並且不必設定 `AllowNonRestoredState`。
- 刪除運算子：如果現有的運算子被刪除，則儲存點會有該遺失運算子的狀態資料。除非將 `AllowNonRestoredState` 設定為 `true`，否則會發生錯誤。
- 修改運算子：如果進行了相容的變更，例如將參數類型變更為相容類型，應用程式就可以從過時的快照還原。如需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，建議將 [StartApplication](#) 動作的 `ApplicationRestoreType` 參數設定為 `SKIP_RESTORE_FROM_SNAPSHOT`，以略過從快照還原。

如需 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 動作的範例請求。如需如何使用 JSON 檔案作為 API 動作輸入的相關資訊，請參閱 [Managed Service for Apache Flink API 範例程式碼](#)。

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"  
  }  
}  
}
```

針對 Apache Flink 使用就地版本升級

透過 Apache Flink 的就地版本升級，您可以保留跨 Apache Flink 版本針對單一 ARN 的應用程式可追蹤性。這包括快照、日誌、指標、標籤、Flink 組態、資源限制增加、VPCs 等。

您可以為 Apache Flink 執行就地版本升級，將現有應用程式升級至 Amazon Managed Service for Apache Flink 中的新 Flink 版本。若要執行此任務，您可以使用 AWS CLI、AWS CloudFormation、AWS SDK 或 AWS 管理主控台。

Note

您無法將 Apache Flink 的就地版本升級與 Amazon Managed Service for Apache Flink Studio 搭配使用。

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至新的 Apache Flink 版本](#)
- [復原應用程式升級](#)
- [應用程式升級的一般最佳實務和建議](#)
- [應用程式升級的注意事項和已知問題](#)
- [升級到 Flink 2.2：完成指南](#)
- [Flink 2.2 升級的狀態相容性指南](#)

使用 Apache Flink 的就地版本升級來升級應用程式

開始之前，建議您觀看此影片：[就地版本升級](#)。

若要執行 Apache Flink 的就地版本升級，您可以使用 AWS CLI、AWS CloudFormation、AWS SDK 或 AWS 管理主控台。您可以將此功能與 READY 或 RUNNING 狀態的 Managed Service for Apache

Flink 搭配使用的任何現有應用程式搭配使用。它使用 UpdateApplication API 來新增變更 Flink 執行時間的功能。

升級之前：更新您的 Apache Flink 應用程式

當您寫入 Flink 應用程式時，您可以將它們與其相依性綁定到應用程式 JAR，並將 JAR 上傳到您的 Amazon S3 儲存貯體。從那裡，Amazon Managed Service for Apache Flink 會在您選取的新 Flink 執行時間中執行任務。您可能需要更新應用程式，才能與您要升級的 Flink 執行時間相容。Flink 版本之間可能存在導致版本升級失敗的不一致。最常見的是使用來源（輸入）或目的地（接收器、輸出）和 Scala 相依性的連接器。Managed Service for Apache Flink 中的 Flink 1.15 和更新版本與 Scala 無關，您的 JAR 必須包含您計劃使用的 Scala 版本。

更新您的應用程式

1. 閱讀 Flink 社群有關升級狀態應用程式的建議。請參閱[升級應用程式和 Flink 版本](#)。
2. 閱讀了解問題和限制的清單。請參閱[應用程式升級的注意事項和已知問題](#)。
3. 更新您的相依性，並在本機測試您的應用程式。這些相依性通常為：
 1. Flink 執行期和 API。
 2. 連接器建議用於新的 Flink 執行時間。您可以在要更新的特定執行時間的[發行版本](#)中找到這些項目。
 3. Scala – Apache Flink 從 Flink 1.15 開始和包含 Flink，與 Scala 無關。您必須包含要在應用程式 JAR 中使用的 Scala 相依性。
4. 在 zipfile 上建立新的應用程式 JAR，並將其上傳至 Amazon S3。我們建議您使用與先前 JAR/zipfile 不同的名稱。如果您需要轉返，您將使用此資訊。
5. 如果您執行具狀態應用程式，強烈建議您擷取目前應用程式的快照。如果您在升級期間或之後遇到問題，這可讓您以狀態復原。

將您的應用程式升級至新的 Apache Flink 版本

您可以使用 [UpdateApplication](#) 動作來升級 Flink 應用程式。

您可以透過多種方式呼叫 UpdateApplication API：

- 在 [上](#)使用現有的組態工作流程 AWS 管理主控台。
 - 前往 [上](#)的應用程式頁面 AWS 管理主控台。
 - 選擇設定。

- 選取您要從中開始的新執行期和快照，也稱為還原組態。使用最新的設定作為還原組態，從最新的快照啟動應用程式。指向 Amazon S3 上新升級的應用程式 JAR/zip。
- 使用 AWS CLI [update-application](#) 動作。
- Use CloudFormation (CFN)。
 - 更新 [RuntimeEnvironment](#) 欄位。先前，CloudFormation 刪除應用程式並建立新的應用程式，導致您的快照和其他應用程式歷史記錄遺失。現在會 CloudFormation 更新您的 RuntimeEnvironment，而不會刪除您的應用程式。
- 使用 AWS SDK。
 - 如需您選擇的程式設計語言，請參閱 SDK 文件。請參閱 [UpdateApplication](#)。

您可以在應用程式處於 RUNNING 狀態或應用程式處於停止READY狀態時執行升級。Amazon Managed Service for Apache Flink 會驗證原始執行時間版本與目標執行時間版本之間的相容性。當您在 RUNNING 狀態執行 [UpdateApplication](#) 時，或在READY處於 狀態時升級時，會在下一個 [StartApplication](#) 執行此相容性檢查。

升級處於 **RUNNING** 狀態的應用程式

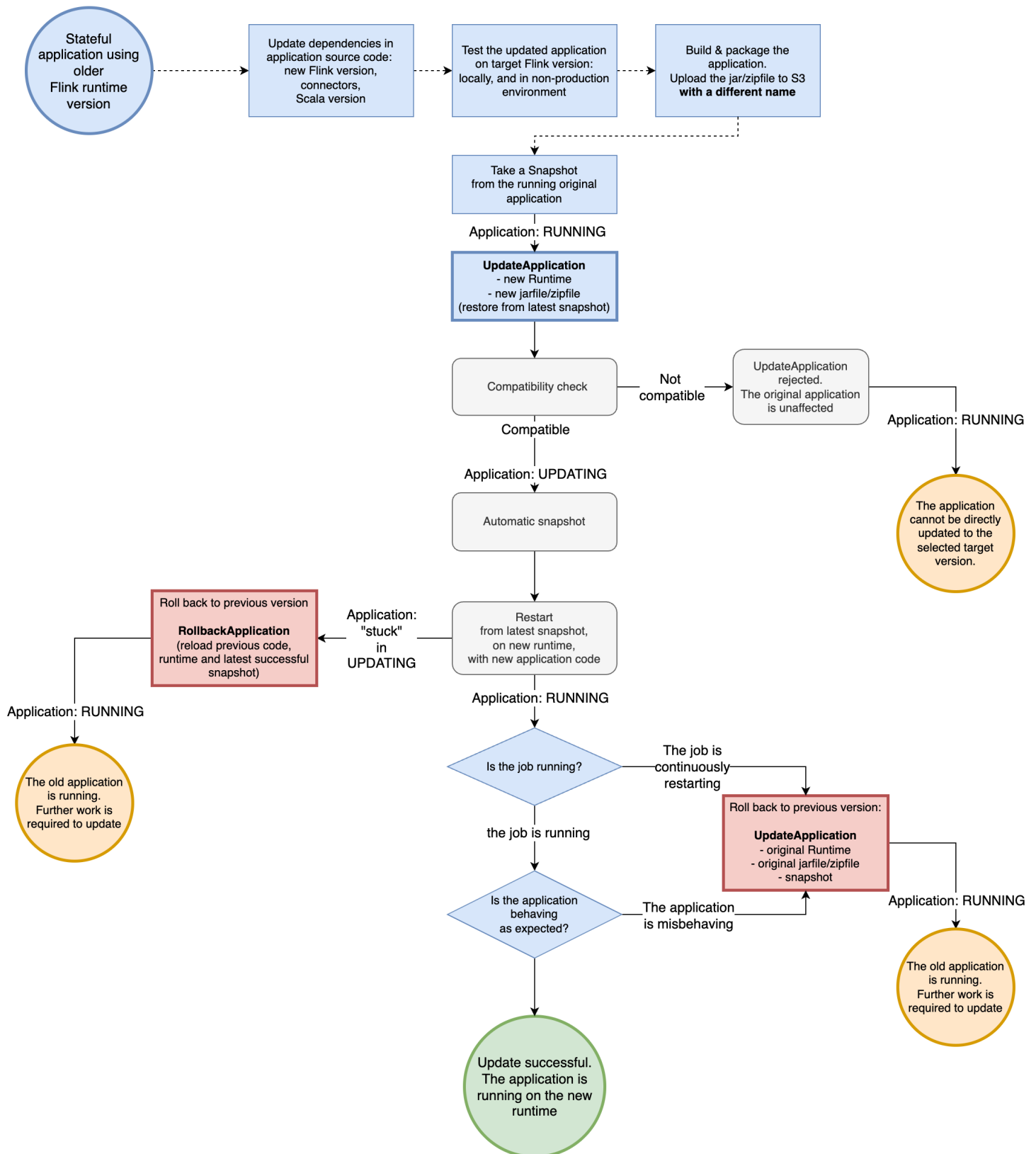
下列範例顯示使用 將RUNNING狀態為 的應用程式升級至美國東部（維吉尼亞北部）的 UpgradeTest Flink 1.18，AWS CLI 並從最新的快照啟動升級的應用程式。

```
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}
```

- 如果您啟用服務快照，並想要從最新的快照繼續應用程式，Amazon Managed Service for Apache Flink 會驗證目前RUNNING應用程式的執行時間是否與所選目標執行時間相容。
- 如果您已指定要繼續目標執行時間的快照，Amazon Managed Service for Apache Flink 會驗證目標執行時間是否與指定的快照相容。如果相容性檢查失敗，您的更新請求會遭到拒絕，而且您的應用程式在 RUNNING 狀態中保持不變。

- 如果您選擇在沒有快照的情況下啟動應用程式，Amazon Managed Service for Apache Flink 不會執行任何相容性檢查。
- 如果您升級的應用程式失敗或卡在傳輸UPDATING狀態，請遵循 [復原應用程式升級](#) 區段中的指示以返回運作狀態。

執行狀態應用程式的處理流程



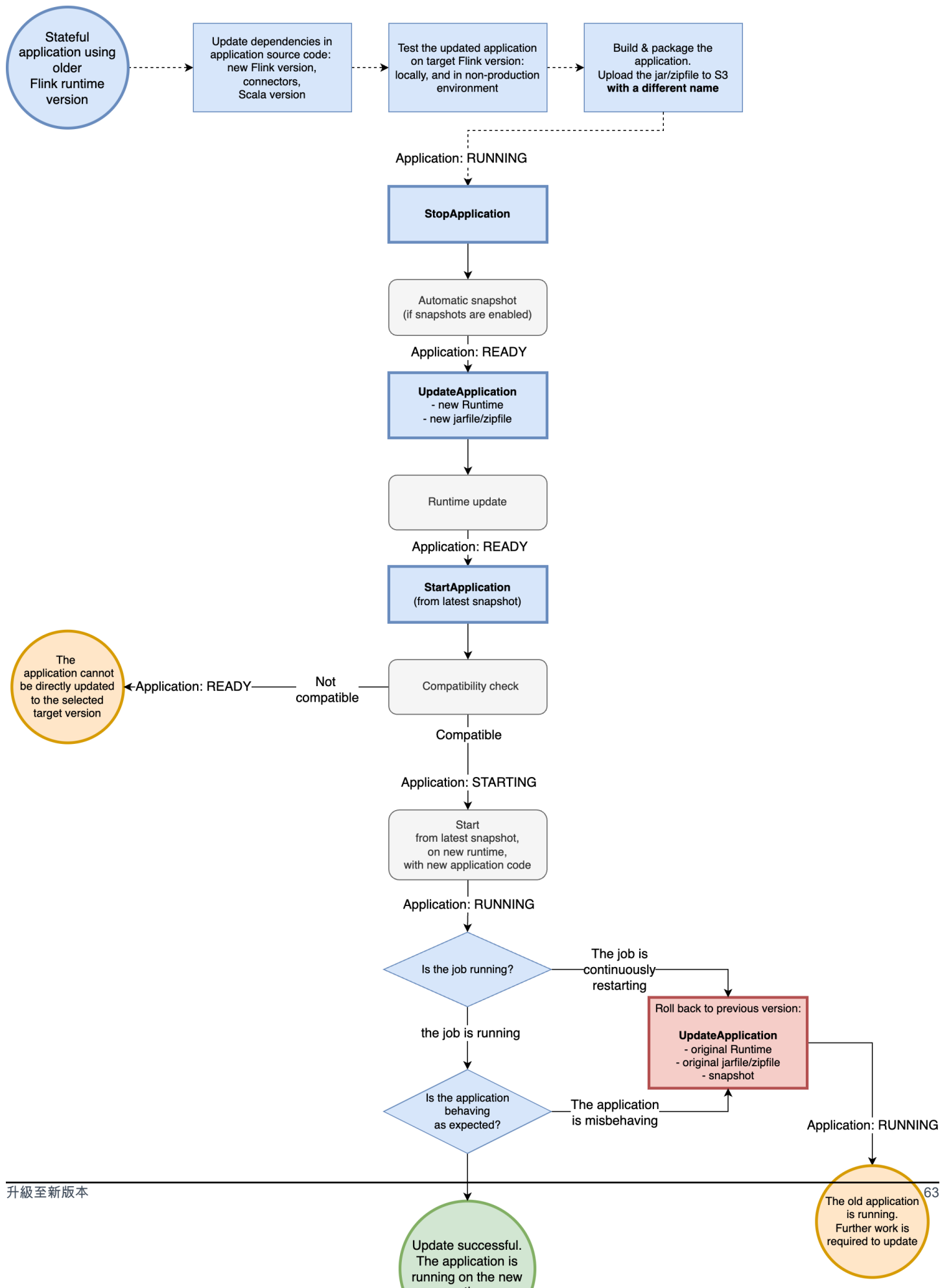
升級處於 READY 狀態的應用程式

下列範例顯示使用 將READY狀態為 的應用程式升級至美國東部（維吉尼亞北部）的 UpgradeTest Flink 1 AWS CLI.18。因為應用程式未執行，所以沒有指定的快照可啟動應用程式。您可以在發出啟動應用程式請求時指定快照。

```
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}
```

- 您可以將處於 READY 狀態的應用程式執行時間更新為任何 Flink 版本。在您啟動應用程式之前，Amazon Managed Service for Apache Flink 不會執行任何檢查。
- Amazon Managed Service for Apache Flink 只會針對您選擇啟動應用程式的快照執行相容性檢查。這些是遵循 [Flink 相容性資料表的基本相容性](#) 檢查。他們只會檢查拍攝快照的 Flink 版本，以及您要鎖定的 Flink 版本。如果所選快照的 Flink 執行時間與應用程式的新執行時間不相容，則啟動請求可能會被拒絕。

就緒狀態應用程式的處理流程



復原應用程式升級

如果您的應用程式發生問題，或在 Flink 版本之間發現應用程式程式碼不一致，您可以使用 AWS CLI、AWS CloudFormation AWS SDK 或復原 AWS 管理主控台。下列範例顯示在不同失敗案例中復原的情況。

執行時間升級成功，應用程式處於 **RUNNING** 狀態，但任務失敗並持續重新啟動

假設您嘗試將名為的狀態應用程式 TestApplication 從 Flink 1.15 升級至美國東部（維吉尼亞北部）的 Flink 1.18。不過，即使應用程式處於 RUNNING 狀態，升級的 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。為了避免進一步停機，建議您立即將應用程式復原至先前的執行版本 (Flink 1.15)，並在稍後診斷問題。

若要將應用程式復原至先前的執行版本，請使用 [Rollback-application](#) AWS CLI 命令或 [RollbackApplication](#) API 動作。此 API 動作會復原您所做的變更，進而產生最新版本。然後，它會使用最新的成功快照重新啟動您的應用程式。

強烈建議您先使用現有應用程式拍攝快照，再嘗試升級。這將有助於避免資料遺失或必須重新處理資料。

在此失敗案例中，CloudFormation 不會為您復原應用程式。您必須更新 CloudFormation 範本以指向先前的執行時間和先前的程式碼，以強制 CloudFormation 更新應用程式。否則，CloudFormation 會假設您的應用程式在轉換為 RUNNING 狀態時已更新。

復原卡在 中的應用程式 **UPDATING**

如果您的應用程式在升級嘗試後卡在 UPDATING 或 AUTOSCALING 狀態，Amazon Managed Service for Apache Flink 會提供 [轉返應用程式](#) AWS CLI 命令，或 [RollbackApplications](#) API 動作，可在卡住 UPDATING 或 AUTOSCALING 狀態之前將應用程式轉返至版本。此 API 會復原您所做的變更，導致應用程式卡在 UPDATING 或 AUTOSCALING 傳輸狀態。

應用程式升級的一般最佳實務和建議

- 在嘗試生產升級之前，在非生產環境中測試沒有狀態的新任務/執行時間。
- 考慮先使用非生產應用程式測試有狀態升級。
- 請確定您的新任務圖表具有與您用來啟動升級應用程式的快照相容的狀態。
 - 請確定存放在運算子狀態中的類型保持不變。如果類型已變更，則 Apache Flink 無法還原運算子狀態。

- 請確定您使用 `uid` 方法設定的運算 IDs 保持不變。Apache Flink 強烈建議將唯一 IDs 指派給運算子。如需詳細資訊，請參閱 Apache Flink 文件中的[指派運算子 IDs](#)。

如果您未將 IDs 指派給運算子，Flink 會自動產生 ID。在這種情況下，它們可能取決於程式結構，如果變更，可能會導致相容性問題。Flink 使用運算子 IDs 將快照中的狀態與運算子比對。變更運算子 IDs 會導致應用程式未啟動，或快照中存放的狀態遭到捨棄，而新的運算子在沒有狀態的情況下啟動。

- 請勿變更用於存放金鑰狀態的金鑰。
- 請勿修改狀態運算子的輸入類型，例如視窗或聯結。這會隱含地變更運算子的內部狀態類型，導致狀態不相容。

應用程式升級的注意事項和已知問題

代理程式重新啟動後，Kafka 遞交檢查點會重複失敗

Flink 1.15 版中的 Apache Kafka 連接器存在已知的開放原始碼 Apache Flink 問題，原因在於 Kafka 用戶端 2.8.1 中的重大開放原始碼 Kafka 用戶端錯誤。如需詳細資訊，請參閱在[代理程式重新啟動後，Kafka 遞交檢查點會重複失敗](#)，且在 [commitOffsetAsync 例外狀況之後，KafkaConsumer 無法復原與群組協調器的連線](#)。

為了避免此問題，建議您在 Amazon Managed Service for Apache Flink 中使用 Apache Flink 1.18 或更新版本。

狀態相容性的已知限制

- 如果您使用的是資料表 API，Apache Flink 不保證 Flink 版本之間的狀態相容性。如需詳細資訊，請參閱 Apache Flink 文件中的[狀態升級和演變](#)。
- 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 及更新版本，API 會拒絕您的請求。您可以升級至 1.8、1.11、1.13 和 1.15，並拍攝快照，然後升級至 1.18 和更新版本。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

- 如果您使用的是 Flink 1.11 或更早版本，並使用 `amazon-kinesis-connector-flink` 連接器進行 Enhanced-fan-out(EFO) 支援，則必須採取額外的步驟，將狀態升級到 Flink 1.13 或更新版本。這是因為連接器套件名稱的變更。如需詳細資訊，請參閱 [amazon-kinesis-connector-flink](#)。

Flink 1.11 和更早版本的 `amazon-kinesis-connector-flink` 連接器使用封裝 `software.amazon.kinesis`，而 Flink 1.13 和更新版本的 Kinesis 連接器使用 `org.apache.flink.streaming.connectors.kinesis` 使用此工具來支援遷移：[amazon-kinesis-connector-flink-state-migrator](#)。

- 如果您使用 Flink 1.13 或更早版本搭配 `FlinkKinesisProducer` 並升級至 Flink 1.15 或更新版本，若要有狀態升級，您必須繼續 `FlinkKinesisProducer` 在 Flink 1.15 或更新版本中使用，而非較新的。`KinesisStreamsSink` 不過，如果您的接收器上已有自訂 `uid` 設定，您應該可以切換到 `KinesisStreamsSink` 因為 `FlinkKinesisProducer` 不會保持狀態。Flink 會將其視為相同的運算子，因為 `uid` 已設定自訂。

以 Scala 撰寫的 Flink 應用程式

- 從 Flink 1.15 開始，Apache Flink 在執行時間中不包含 Scala。升級至 Flink 1.15 或更新版本時，您必須在程式碼 JAR/zip 中包含要使用的 Scala 版本和其他 Scala 相依性。如需詳細資訊，請參閱 [Amazon Managed Service for Apache Flink 1.15.2 版](#)。
- 如果您的應用程式使用 Scala，而且您要從 Flink 1.11 或更早版本 (Scala 2.11) 升級至 Flink 1.13 (Scala 2.12)，請確定您的程式碼使用 Scala 2.12。否則，您的 Flink 1.13 應用程式可能無法在 Flink 1.13 執行時間中找到 Scala 2.11 類別。

降級 Flink 應用程式時應考量的事項

- 可以降級 Flink 應用程式，但僅限於應用程式之前使用舊版 Flink 執行的情況。對於狀態升級，Managed Service for Apache Flink 需要使用相符或更舊版本的快照進行降級。
- 如果您要將執行時間從 Flink 1.13 或更新版本更新為 Flink 1.11 或更新版本，而且如果您的應用程式使用 `HashMap` 狀態後端，您的應用程式將會持續失敗。

升級到 Flink 2.2：完成指南

本指南提供 step-by-step 說明。這是主要版本升級，其中包含需要仔細規劃和測試的重大變更。

主要版本升級是單向

升級操作可以使用狀態保留將您的應用程式從 Flink 1.x 移至 2.2，但您無法從 2.2 移至具有 2.2 狀態的 1.x。如果您的應用程式在升級後變得運作狀態不佳，請使用轉返 API，從最新的快照返回具有原始 1.x 狀態的 1.x 版本。

先決條件

開始升級之前：

- 檢閱 [中斷變更和棄用](#)
- 檢閱 [Flink 2.2 升級的狀態相容性指南](#)
- 確保您擁有用於測試的非生產環境
- 記錄您目前的應用程式組態和相依性

了解遷移路徑

您的升級體驗取決於應用程式與 Flink 2.2 的相容性。了解這些路徑可協助您適當地準備並設定逼真的期望。

路徑 1：相容的二進位和應用程式狀態

預期事項：

- 叫用升級操作
- 使用應用程式狀態轉換完成遷移至 2.2：RUNNING → UPDATING → RUNNING
- 保留所有應用程式狀態，而不會遺失資料或進行重新處理
- 與次要版本遷移相同的體驗

最適合：無狀態應用程式或使用相容序列化的應用程式 (Avro、相容的 Protobuf 結構描述、不含集合 POJOs)

路徑 2：二進位不相容

預期事項：

- 叫用升級操作

- 操作失敗，並透過操作 API 和日誌顯示二進位不相容
- 啟用自動轉返：應用程式會在幾分鐘內自動轉返，無需您的介入
- 停用自動轉返：應用程式在未處理資料的情況下仍處於執行中狀態；您可以手動轉返至較舊的版本
- 修正二進位檔案後，請使用 [UpdateApplication API](#) 來獲得類似路徑 1 的體驗

最適合：使用 Flink 任務啟動期間偵測到之已移除 APIs 的應用程式

路徑 3：不相容的應用程式狀態

預期事項：

- 叫用升級操作
- 遷移一開始似乎成功
- 當狀態還原失敗時，應用程式會在幾秒鐘內進入重新啟動迴圈
- 透過顯示持續重新啟動的 CloudWatch 指標偵測失敗
- 手動叫用轉返操作
- 啟動復原後幾分鐘內返回生產環境
- [狀態遷移](#) 檢閱您的應用程式

最適合：狀態序列化不相容的應用程式（具有集合、特定 Kryo 序列化狀態的 POJOs）

Note

強烈建議您建立生產應用程式的複本，並在複本上測試下列每個升級階段，然後再遵循生產應用程式的相同步驟。

階段 1：準備

更新應用程式程式碼

更新您的應用程式程式碼，使其與 Flink 2.2 相容：

- 在 或 中將 Flink 相依性更新至 2.2.0 版 pom.xml build.gradle
- 將連接器相依性更新至 Flink 2.2 相容版本（請參閱 [連接器可用性](#)）
- 移除已棄用的 API 用量：

- 以 DataStream API 或資料表 API/SQL 取代 DataSet API DataStream
- 使用 FLIP-27 來源和 FLIP-143 接收器 APIs 取代舊版 SourceFunction/SinkFunction
- 以 Java API 取代 Scala API 用量
- 更新至 Java 17

上傳更新的應用程式程式碼

- 使用 Flink 2.2 相依性建置您的應用程式 JAR
- 使用與目前 JAR 不同的檔案名稱上傳到 Amazon S3 (例如 my-app-flink-2.2.jar)
- 請注意要在升級步驟中使用的 S3 儲存貯體和金鑰

階段 2：啟用自動轉返

自動轉返可讓 Amazon Managed Service for Apache Flink 在升級失敗時自動還原至先前的版本。

檢查自動轉返狀態

AWS 管理主控台：

1. 導覽至您的應用程式
2. 選擇 Configuration (組態)。
3. 在應用程式設定下，確認系統復原已啟用

AWS CLI:

```
aws kinesisanalyticsv2 describe-application \  
  --application-name MyApplication \  
  --query  
'ApplicationDetail.ApplicationConfigurationDescription.ApplicationSystemRollbackConfigurationD
```

啟用自動轉返 (如果未啟用)

```
aws kinesisanalyticsv2 update-application \  
  --application-name MyApplication \  
  --current-application-version-id <version-id> \  
  --application-configuration-update '{  
    "ApplicationSystemRollbackConfigurationUpdate": {
```

```
        "RollbackEnabledUpdate": true
    }
}'
```

階段 3：拍攝快照（選用）

如果您的應用程式已啟用自動快照，您可以略過此步驟，否則請在升級之前擷取應用程式的快照以儲存應用程式的狀態。

從執行中的應用程式擷取快照

AWS 管理主控台：

1. 導覽至您的應用程式
2. 選擇快照
3. 選擇建立快照
4. 輸入快照名稱（例如 pre-flink-2.2-upgrade）
5. 選擇 Create (建立)

AWS CLI:

```
aws kinesisanalyticstv2 create-application-snapshot \
  --application-name MyApplication \
  --snapshot-name pre-flink-2.2-upgrade
```

驗證快照建立

```
aws kinesisanalyticstv2 describe-application-snapshot \
  --application-name MyApplication \
  --snapshot-name pre-flink-2.2-upgrade
```

等到 SnapshotStatus 變成 READY再繼續。

階段 4：升級應用程式

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

您可以透過多種方式呼叫 UpdateApplication API：

- 使用 AWS 管理主控台。

- 前往 上的應用程式頁面 AWS 管理主控台。
- 選擇設定。
- 選取您要從中開始的新執行期和快照，也稱為還原組態。使用最新的設定作為還原組態，從最新的快照啟動應用程式。指向 Amazon S3 上新升級的應用程式 JAR/zip。
- 使用 AWS CLI [update-application](#) 動作。
- 使用 CloudFormation。
 - 更新 RuntimeEnvironment 欄位。先前，CloudFormation 已刪除應用程式並建立新的應用程式，導致您的快照和其他應用程式歷史記錄遺失。現在會 CloudFormation 更新您的 RuntimeEnvironment 就地，不會刪除您的應用程式。
- 使用 AWS SDK。
 - 如需您選擇的程式設計語言，請參閱 SDK 文件。請參閱 [UpdateApplication](#)。

您可以在應用程式處於 RUNNING 狀態或應用程式處於停止READY狀態時執行升級。Amazon Managed Service for Apache Flink 會驗證原始執行時間版本與目標執行時間版本之間的相容性。此相容性檢查會在處於 UpdateApplication RUNNING 狀態時執行，或在READY處於 狀態時升級StartApplication時於下一個執行。

從 RUNNING 狀態升級

```
aws kinesisanalyticsv2 update-application \  
  --application-name MyApplication \  
  --current-application-version-id <version-id> \  
  --runtime-environment-update FLINK-2_2 \  
  --application-configuration-update '{  
    "ApplicationCodeConfigurationUpdate": {  
      "CodeContentUpdate": {  
        "S3ContentLocationUpdate": {  
          "FileKeyUpdate": "my-app-flink-2.2.jar"  
        }  
      }  
    }  
  }'
```

從 READY 狀態升級

```
aws kinesisanalyticsv2 update-application \  
  --application-name MyApplication \  
  --current-application-version-id <version-id> \  
  --runtime-environment-update FLINK-2_2
```

```
--runtime-environment-update FLINK-2_2 \  
--application-configuration-update '{  
  "ApplicationCodeConfigurationUpdate": {  
    "CodeContentUpdate": {  
      "S3ContentLocationUpdate": {  
        "FileKeyUpdate": "my-app-flink-2.2.jar"  
      }  
    }  
  }  
}'
```

階段 5：監控升級

相容性檢查

- 使用 操作 API 檢查升級的狀態。如果任務啟動有二進位不相容或問題，升級操作將使用 日誌失敗。
- 如果升級操作成功，但應用程式卡在重新啟動迴圈中，這表示狀態與新的 Flink 版本不相容，或更新的程式碼發生問題。[Flink 2.2 升級的狀態相容性指南](#) 檢閱如何識別狀態不相容問題。

監控應用程式運作狀態

應用程式狀態：

- 應用程式狀態應轉換：RUNNING→ UPDATING → RUNNING
- 檢查應用程式的執行時間。如果是 2.2，表示升級操作成功。
- 如果您的應用程式在 中，RUNNING但仍處於較舊的執行時間，則自動轉返會開始。操作 API 會將操作顯示為 FAILED。檢查日誌以尋找失敗的例外狀況。

此外，請在 CloudWatch 中監控這些指標：

重新啟動指標：

- numRestarts：監控意外重新啟動 — 如果 numRestarts 為零且 uptime 或 runningTime 正在增加，則升級成功。

檢查點指標：

- lastCheckpointDuration：應該類似於預先升級的值
- numberOfFailedCheckpoints：應保持在 0

階段 6：驗證應用程式行為

在 Flink 2.2 上執行應用程式之後：

功能驗證

- 確認正在從來源讀取資料
- 確認資料正在寫入目的地
- 驗證業務邏輯產生預期結果
- 比較輸出與升級前基準

效能驗證

- 監控延遲指標 end-to-end處理時間)
- 監控輸送量指標 (每秒記錄數)
- 監控檢查點持續時間和大小
- 監控記憶體和 CPU 使用率

執行 24 小時以上

允許應用程式在生產環境中執行至少 24 小時，以確保：

- 沒有記憶體洩漏
- 穩定檢查點行為
- 沒有意外重新啟動
- 一致的輸送量

階段 7：轉返程序

如果升級失敗或應用程式正在執行但運作狀態不佳，請復原至先前的版本。

自動轉返

如果啟用自動轉返，且在啟動期間升級失敗，Amazon Managed Service for Apache Flink 會自動還原至先前的版本。

手動轉返

如果應用程式正在執行但運作狀態不佳，請使用 RollbackApplication API：

AWS 管理主控台：

1. 導覽至您的應用程式
2. 選擇動作 → 復原
3. 確認轉返

AWS CLI:

```
aws kinesisanalyticsv2 rollback-application \  
  --application-name MyApplication \  
  --current-application-version-id <version-id>
```

復原期間會發生的情況：

- 應用程式停止
- 執行時間會還原至先前的 Flink 版本
- 應用程式程式碼會還原至先前的 JAR
- 應用程式會從升級前取得的最後一個成功快照重新啟動

Important

- 您無法還原 Flink 1.x 上的 Flink 2.2 快照
- 轉返使用升級之前拍攝的快照
- 升級之前一律拍攝快照（階段 3）

後續步驟

有關升級期間的問題，請參閱 [Managed Service for Apache Flink 故障診斷](#) 或聯絡 AWS Support。

Flink 2.2 升級的狀態相容性指南

從 Flink 1.x 升級到 Flink 2.2 時，狀態相容性問題可能會使您的應用程式無法從快照還原。本指南可協助您識別潛在的相容性問題，並提供遷移策略。

了解狀態相容性變更

Amazon Managed Service for Apache Flink 2.2 引進數個會影響狀態相容性的序列化變更。以下是主要項目：

- **Kryo 版本升級：**Apache Flink 2.2 會將綁定的 Kryo 序列化程式從第 2 版升級到第 5 版。由於 Kryo v5 使用與 Kryo v2 不同的二進位編碼格式，因此無法在 Flink 2.2 中還原透過 Kryo 在 Flink 1.x 儲存點中序列化的任何運算子狀態。
- **Java 集合序列化：**在 Flink 1.x 中，POJOs 內的 Java 集合（例如 ArrayList、HashMap 和 HashSet）使用 Kryo 序列化。Flink 2.2 推出與來自 1.x 的 Kryo 序列化狀態不相容的集合特定最佳化序列化程式。在 1.x 中使用 Java 集合搭配 POJO 或 Kryo 序列化程式的應用程式無法在 Flink 2.2 中還原此狀態。如需資料類型和序列化的詳細資訊，請參閱 Flink [文件](#)。
- **Kinesis 連接器相容性：**低於 5.0 的 Kinesis Data Streams (KDS) 連接器版本會維持與 Flink 2.2 Kinesis 連接器 6.0 版不相容的狀態。您必須先遷移至連接器 5.0 版或更新版本，才能升級。

序列化相容性參考

檢閱應用程式中的所有狀態宣告，並將序列化類型與下表比對。如果有任何狀態類型不相容，請先參閱[狀態遷移](#)一節，再繼續升級。

序列化相容性參考

序列化類型	相容？	詳細資訊
Avro (SpecificRecord、GenericRecord)	是	使用自己的二進位格式，獨立於 Kryo。確保您使用的是 Flink 的原生 Avro 類型資訊，而不是註冊為 Kryo 序列化程式的 Avro。
Protobuf	是	使用自己的二進位編碼，獨立於 Kryo。驗證結構描述變更遵循回溯相容的演變規則。
沒有集合 POJOs	是	由 Flink 的 POJO 序列化程式處理 — 但僅限於類別符合所有 POJO 條件時：公有類別、公有無參數建構函數、所有公開

序列化類型	相容？	詳細資訊
		或可透過 getter/setters 存取的欄位，以及所有欄位類型本身可透過 Flink 序列化。違反上述任何一項的 POJO 無提示地落回 Kryo，並變得不相容。
自訂 TypeSerializers	是	只有在您的序列化程式未在內部委派給 Kryo 時才相容。
SQL 和資料表 API 狀態	是（具有警告）	使用 Flink 的內部序列化程式。不過，Apache Flink 不保證資料表 API 應用程式的主要版本之間的狀態相容性。首先在非生產環境中進行測試。
具有 Java 集合的 POJOs (HashMap、ArrayList、HashSet)	否	在 Flink 1.x 中，POJOs 內的集合是透過 Kryo v2 序列化。Flink 2.2 推出專用集合序列化程式，其二進位格式與 Kryo v2 格式不相容。
Scala 案例類別	否	在 Flink 1.x 中透過 Kryo 序列化。Kryo v2 至 v5 升級會變更二進位格式。
Java 記錄	否	通常回到 Flink 1.x 中的 Kryo 序列化。使用 <code>進行測試來驗證 disableGenericType</code> <code>s()</code> 。
第三方程式庫類型	否	沒有已註冊自訂序列化程式的類型會回到 Kryo。Kryo v2 到 v5 二進位格式變更會中斷相容性。

序列化類型	相容？	詳細資訊
使用 Kryo 備用的任何類型	否	如果 Flink 無法處理具有內建或已註冊序列化程式的類型，則會回到 Kryo。來自 1.x 的所有 Kryo 序列化狀態與 2.2 不相容。

診斷方法

您可以在 [UpdateApplication API](#) 操作之後查看應用程式日誌或檢查日誌，以主動識別狀態相容性問題。

識別應用程式中的 Kryo 備用

您可以在日誌中使用下列 regex 模式來識別應用程式中的 Kryo 備用：

```
Class class (?<className>[^\s]+) cannot be used as a POJO type
```

範例日誌：

```
Class class org.apache.flink.streaming.connectors.kinesis.model.SequenceNumber
cannot be used as a POJO type because not all fields are valid POJO fields,
and must be processed as GenericType. Please read the Flink documentation on
"Data Types & Serialization" for details of the effect on performance and
schema evolution.
```

如果使用 UpdateApplication API 升級失敗，下列例外狀況可能表示您遇到序列化程式型狀態不相容：

IndexOutOfBoundsException

```
Caused by: java.lang.IndexOutOfBoundsException: Index 116 out of bounds for length 1
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Unknown Source)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Unknown Source)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Unknown Source)
    at java.base/java.util.Objects.checkIndex(Unknown Source)
    at java.base/java.util.ArrayList.get(Unknown Source)
    at
    com.esotericsoftware.kryo.util.MapReferenceResolver.getReadObject(MapReferenceResolver.java:77)
```

```
at com.esotericsoftware.kryo.Kryo.readReferenceOrNull(Kryo.java:923)
... 23 more
```

StateMigrationException (POJOSerializer)

Caused by: org.apache.flink.util.StateMigrationException: The new state serializer (org.apache.flink.api.java.typeutils.runtime.PojoSerializer@8bf85b5d) must not be incompatible with the old state serializer (org.apache.flink.api.java.typeutils.runtime.PojoSerializer@3282ee3).

升級前檢查清單

- 檢閱應用程式中的所有狀態宣告
- 使用集合 (HashMap、ArrayList、HashSet) 檢查 POJOs
- 驗證每種狀態類型的序列化方法
- 在此複本上使用 UpdateApplication API 建立生產複本應用程式並測試狀態相容性
- 如果狀態不相容，請從 選取策略 [狀態遷移](#)
- 在生產 Flink 應用程式組態中啟用自動轉返

狀態遷移

重建完成狀態

最適合可以從來源資料重建狀態的應用程式。

如果您的應用程式可以從來源資料重建狀態：

1. 停止 Flink 1.x 應用程式
2. 使用更新的程式碼升級至 Flink 2.x
3. 從 開始 SKIP_RESTORE_FROM_SNAPSHOT
4. 允許應用程式重建狀態

```
aws kinesisanalyticsv2 start-application \  
  --application-name MyApplication \  
  --run-configuration '{  
    "ApplicationRestoreConfiguration": {  
      "ApplicationRestoreType": "SKIP_RESTORE_FROM_SNAPSHOT"
```

```
}  
'
```

最佳實務

1. 一律將 Avro 或 Protobuf 用於複雜狀態 — 這些可提供結構描述演變，且與 Kryo 無關
2. 避免 POJOs 中的集合 — 請 MapState 改用 Flink 的原生 ListState 和。
3. 在本機測試狀態還原 - 在生產升級之前，使用實際快照進行測試
4. 經常拍攝快照 - 特別是在主要版本升級之前
5. 啟用自動轉返 — 將您的 MSF 應用程式設定為在失敗時自動轉返
6. 記錄您的狀態類型 — 維護所有狀態類型及其序列化方法的文件
7. 監控檢查點大小 — 不斷增長的檢查點大小可能表示序列化問題

後續步驟

規劃升級：請參閱[升級到 Flink 2.2：完成指南](#)。

如需遷移期間的問題，請參閱 [Managed Service for Apache Flink 故障診斷](#) 或聯絡 AWS Support。

在 Managed Service for Apache Flink 中實作應用程式擴展

您可以為 Amazon Managed Service for Apache Flink 設定任務的平行執行和資源配置，以實作擴展。如需有關 Apache Flink 如何排程任務平行執行個體的資訊，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ParallelismPerKPU](#)
- [配置 Kinesis 處理單元](#)
- [更新應用程式的平行處理](#)
- [在 Managed Service for Apache Flink 中使用自動擴展](#)
- [maxParallelism 考量](#)

設定應用程式平行處理和ParallelismPerKPU

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**：使用此屬性設定依應用程式每 Kinesis 處理單元 (KPU) 可排程的平行任務數目。預設值為 1，最大值為 8。對於具有封鎖作業 (例如 I/O) 的應用程式，較高的 ParallelismPerKPU 值會導致 KPU 資源的完整使用率。

Note

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

如需為特定運算子設定任務平行處理的資訊，請參閱 Apache Flink 文件中的[設定平行處理：運算子](#)。

配置 Kinesis 處理單元

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

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

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

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

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

Note

額外的 KPU 需要為了協同運作目的付費。如需詳細資訊，請參閱 [Managed Service for Apache Flink 定價](#)。

更新應用程式的平行處理

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

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

```
{
  "ApplicationName": "string",
  "RuntimeEnvironment": "FLINK-1_18",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "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 Managed Service for Apache Flink 中使用自動擴展

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

- 如果 CloudWatch 指標最大值containerCPUUtilization大於 75% 或更高 15 分鐘，您的應用程式會縱向擴展（增加平行處理）。這表示當 15 個連續資料點的 1 分鐘期間等於或超過 75% 時，就會啟動ScaleUp動作。ScaleUp 動作會將CurrentParallelism您應用程式的 加倍。ParallelismPerKPU 不會修改。因此，配置KPUs 數量也會加倍。
- 當 CPU 使用率維持在 10% 以下達 6小時時，應用程式會縮減規模 (減少平行處理層級)。這表示當有 360 個連續資料點且 1 分鐘期間小於 10% 時，就會啟動ScaleDown動作。ScaleDown 動作會減半（四捨五入）應用程式的平行處理。ParallelismPerKPU 不會修改，配置KPUs 數量也會減半（四捨五入）。

Note

最多可以參考containerCPUUtilization超過 1 分鐘的期間，以尋找與用於擴展動作之資料點的關聯性，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，您唯一可以使用的有效 API 動作是 [StopApplication](#)，且 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。AWS 您的帳戶會針對 Managed Service for Apache Flink 佈建的 KPIs 收費，這是應用程式parallelism和parallelismPerKPU設定的函數。活動尖峰會增加您的 Managed Service for Apache Flink 成本。

如需定價相關的資訊，請參閱 [Amazon Managed Service for Apache Flink 定價](#)。

請留意下列與應用程式擴展相關的資訊：

- 預設會啟用自動擴展。
- 擴展不適用於 Studio 筆記本。不過，如果您將 Studio 筆記本部署為具有持久狀態的應用程式，則擴展將套用到已部署的應用程式。
- 應用程式的預設限制為 64 個 KPI。如需詳細資訊，請參閱[Managed Service for Apache Flink 和 Studio 筆記本配額](#)。
- 當自動擴展更新應用程式的平行處理層級時，應用程式會經歷停機。若要避免停機，請執行下列動作：
 - 停用自動擴展
 - 使用 [UpdateApplication](#) 動作設定應用程式的 parallelism 和 parallelismPerKPI。如需設定應用程式平行處理設定的詳細資訊，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 “Managed Service for Apache Flink 中的指標和維度”](#)。

實作自訂自動擴展

如果您想要對自動擴展進行更精細的控制，或使用 以外的觸發指標 `containerCPUUtilization`，您可以使用此範例：

- [AutoScaling](#)

此範例說明如何使用與 Apache Flink 應用程式不同的 CloudWatch 指標來擴展 Managed Service for Apache Flink 應用程式，包括使用 Amazon MSK 和 Amazon Kinesis Data Streams 做為來源或接收器的指標。

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

實作排定的自動擴展

如果您的工作負載隨著時間遵循可預測的設定檔，您可能偏好先行擴展 Apache Flink 應用程式。這會在排程時間擴展您的應用程式，而不是根據指標被動擴展。若要在一天中的固定時間設定擴展和縮減，您可以使用此範例：

- [ScheduledScaling](#)

maxParallelism 考量

Flink 任務可以擴展的最大平行處理，受限於任務 `maxParallelism` 所有運算子的最小值。例如，如果您有一個只有來源和接收器的簡單任務，而來源的 為 `maxParallelism 16`，而目的地的 為 8，則應用程式無法擴展到超過 8 的平行處理。

若要了解如何計算 運算子 `maxParallelism` 的預設值，以及如何覆寫預設值，請參閱 Apache Flink 文件中的 [設定最大平行處理](#)。

作為基本規則，請注意，如果您未 `maxParallelism` 為任何運算子定義，並且您以小於或等於 128 `maxParallelism` 的平行處理啟動應用程式，則所有運算子的 都將是 128。

Note

任務的最大平行處理是擴展應用程式保留 狀態的平行處理上限。

如果您修改 `maxParallelism` 現有應用程式，則應用程式將無法從先前使用舊 拍攝的快照重新啟動 `maxParallelism`。您只能在沒有快照的情況下重新啟動應用程式。

如果您計劃將應用程式擴展到大於 128 的平行處理，您必須在maxParallelism應用程式中明確設定。

- 自動擴展邏輯可防止將 Flink 任務擴展到超過任務最大平行處理的平行處理。
- 如果您使用自訂自動擴展或排程擴展，請將它們設定為不超過任務的最大平行處理。
- 如果您手動擴展應用程式超過最大平行處理，則應用程式無法啟動。

將標籤新增至 Managed Service for Apache Flink 應用程式

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 [TagResource](#) 動作將標籤新增至應用程式。您無法使用 [UpdateApplication](#) 動作為應用程式新增標籤。

若要更新現有的標籤，請使用與現有標籤相同的索引鍵新增標籤。

`TagResource` 動作的下列請求範例會新增標籤或更新現有標籤：

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

列出應用程式的標籤

若要列出現有標籤，請使用 [ListTagsForResource](#) 動作。

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

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

從應用程式移除標籤

若要從應用程式移除標籤，請使用 [UntagResource](#) 動作。

UntagResource 動作的下列請求範例可移除應用程式的標籤：

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

搭配 Managed Service for Apache Flink 使用 CloudFormation

下列練習示範如何在相同堆疊中使用 Lambda 函數啟動 CloudFormation 使用 建立的 Flink 應用程式。

開始之前

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

撰寫 Lambda 函數

在建立或更新 Flink 應用程式後，若要啟動它，可以使用 kinesisanalyticsv2 [start-application](#) API。建立 Flink 應用程式後，CloudFormation 事件會觸發呼叫。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 {}:
{}'.format(application_name, run_configuration))

        # this call doesn't wait for an application to transfer to 'RUNNING'
state.

        client_kda.start_application(ApplicationName=application_name,
RunConfiguration=run_configuration)

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

```

在上述程式碼中，Lambda 會處理傳入 CloudFormation 事件、篩選除 Create 和 之外的所有內容 Update、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政策，但您可能想要將其縮小為使用自訂政策。

```

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

下列範例是先前步驟的稍微擴充版本，並透過[範本參數](#)完成額外的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:
                              - kinesisanalyticsv2.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('kinesisanalyticsv2',
region_name=event['ResourceProperties']['Region'])

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

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

        return

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

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

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

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

        logger.info('Started Application: {}'.format(application_name))
        cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

```

```

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

```

您也可以調整 Lambda 的角色以及應用程式本身的角色。

在建立上面的堆疊之前，不要忘記指定參數。

parameters.json

```

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

```

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。

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

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

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

如需詳細資訊，請參閱下列內容：

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

使用 Apache Flink 儀表板搭配 Managed Service for Apache Flink

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

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

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

Note

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

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

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

Note

如果您在三分鐘內未存取產生的 URL，則該 URL 將不再有效。

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

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

支援和取代的 Apache Flink 版本

本主題包含 Managed Service for Apache Flink 中 Apache Flink 支援版本的相關資訊。本主題也列出每個版本中支援的 Apache Flink 功能。

Note

如果您使用的是已棄用的 Apache Flink 版本，建議您使用 Managed Service for Apache Flink 中的 [針對 Apache Flink 使用就地版本升級](#) 功能，將應用程式升級至最新的支援 Flink 版本。

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
2.2.0	支援	支援	Amazon Managed Service for Apache Flink 2.2	這是具有重大變更的主要版本。在繼續 中斷變更和棄用 之前，請參閱。
1.20.0	支援	支援	Amazon Managed Service for Apache Flink 1.20	
1.19.1	支援	支援	Amazon Managed Service for Apache Flink 1.19	
1.18.1	支援	不支援	Amazon Managed Service for	

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
			Apache Flink 1.18	
1.15.2	支援	不支援	Amazon Managed Service for Apache Flink 1.15	
1.13.1	棄用	不支援	入門：Flink 1.13.2	Amazon Managed Service for Apache Flink 中對此版本的支援將於 2025 年 10 月 16 日結束。

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
1.11.1	棄用	不支援	Managed Service for Apache Flink 的早期版本資訊 (自 2025 年 2 月起不支援此版本)	我們計劃結束對 Amazon Managed Service for Apache Flink 中 Apache Flink 1.6、1.8 和 1.11 版的支援。 從 2025 年 7 月 14 日起，我們將使用這些版本的應用程式置於 READY 狀態。 從 2025 年 7 月 28 日起，您將無法使用這些版本啟動應用程式。 我們建議您現在立即使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，將應用程式升級至 Flink 1.20 版。如需詳細資訊，請參閱針對 Apache

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
				Flink 使用就地版本升級。

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
1.8.2	棄用	不支援	Managed Service for Apache Flink 的早期版本資訊 (自 2025 年 2 月起不支援此版本)	我們計劃結束對 Amazon Managed Service for Apache Flink 中 Apache Flink 1.6、1.8 和 1.11 版的支援。 從 2025 年 7 月 14 日起，我們將使用這些版本的應用程式置於 READY 狀態。 從 2025 年 7 月 28 日起，您將無法使用這些版本啟動應用程式。 我們建議您現在立即使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，將應用程式升級至 Flink 1.20 版。如需詳細資訊，請參閱針對 Apache

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
				Flink 使用就地版本升級。

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
1.6.2	棄用	不支援	Managed Service for Apache Flink 的早期版本資訊 (自 2025 年 2 月起不支援此版本)	我們計劃結束對 Amazon Managed Service for Apache Flink 中 Apache Flink 1.6、1.8 和 1.11 版的支援。 從 2025 年 7 月 14 日起，我們將使用這些版本的應用程式置於 READY 狀態。 從 2025 年 7 月 28 日起，您將無法使用這些版本啟動應用程式。 我們建議您現在立即使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，將應用程式升級至 Flink 1.20 版。如需詳細資訊，請參閱針對 Apache

Apache Flink 版本	狀態 - Amazon Managed Service for Apache Flink	狀態 - Apache Flink 社群	連結	注意
				Flink 使用就地版本升級。

主題

- [Amazon Managed Service for Apache Flink 2.2](#)
- [Amazon Managed Service for Apache Flink 1.20](#)
- [Amazon Managed Service for Apache Flink 1.19](#)
- [Amazon Managed Service for Apache Flink 1.18](#)
- [Amazon Managed Service for Apache Flink 1.15](#)
- [Managed Service for Apache Flink 的早期版本資訊](#)

Amazon Managed Service for Apache Flink 2.2

Amazon Managed Service for Apache Flink 現在支援 Apache Flink 2.2 版。這是服務的第一個主要版本升級。此頁面涵蓋 Flink 2.2 中介紹的功能，以及從 Flink 1.x 升級的重要考量。

Note

Flink 2.2 引入了需要仔細規劃的重大變更。從 1.x 升級[Flink 2.2 升級的狀態相容性指南](#)之前，請先檢閱以下和 的重大變更和棄用的完整清單。

Amazon Managed Service for Apache Flink 2.2 的新功能

Amazon Managed Service for Apache Flink 2.2 引入可能在升級時破壞現有應用程式的行為變更。與下一節的 Flink API 變更一起仔細檢閱這些變更。

程式設計組態處理

- 當客戶嘗試透過 `env.getConfig().set()` 或類似 APIs.2 現在會報告例外狀況。請參閱 [程式設計 Flink 組態屬性](#)。

- 客戶仍然可以透過支援票證請求變更特定組態（請參閱[可修改的設定](#)）

指標移除

- 指標 `fullRestarts` 已在 Flink 2.2 中移除。請改用 `numRestarts` 指標。
- 已在 Flink 連接器 `bytesRequestedPerFetch` 6.0.0 版中移除 KDS AWS 連接器的指標（僅限與 Flink 2.2 相容的連接器版本）。
- `uptime` 和 `downtime` 指標在 Flink 2.2 中都標示為已棄用，很快就會移除。`uptime` 將取代為新的指標 `runningTime`。`downtime` 以一或多個 `restartingTime`、`cancellingTime` 和 `failingTime`。
- 如需支援指標的完整清單，請參閱[指標和維度頁面](#)。

封鎖的非憑證 IMDS 呼叫

- 這些允許的端點由 AWS SDK 的 `DefaultCredentialsProvider (/latest/meta-data/iam/security-credentials/)` 和 `DefaultAwsRegionProviderChain (/latest/dynamic/instance-identity/document)` 使用，以自動設定應用程式的登入資料和區域。
- 使用依賴非憑證 IMDS 呼叫（例如 `EC2MetadataUtils.getInstanceId()`、`EC2MetadataUtils.getLocalHostName()`、`EC2MetadataUtils.getInstanceType()` 或 `EC2MetadataUtils.getAvailabilityZone()`）的 AWS 開發套件函數的應用程式，在嘗試這些呼叫時會收到 HTTP 4xx 錯誤。
- 如果您的應用程式將 IMDS 用於執行個體中繼資料或允許路徑以外的其他資訊，請重構程式碼以改用環境變數或應用程式組態。

唯讀根檔案系統

- 為了提高安全性，`/tmp` 以外的任何相依性都是預設 flink 工作目錄，將導致：`java.io.FileNotFoundException: /{path}/{filename} (Read-only file system)`。
- 檔案系統相依性可以直接來自您的程式碼，也可以間接來自您的相依性中包含的程式庫。覆寫程式碼/`tmp`/中的直接檔案系統相依性。對於來自程式庫的間接檔案系統相依性，請使用程式庫組態覆寫將檔案系統操作重新導向至 `/tmp/`。

中斷變更和棄用

以下是 Managed Service for Apache Flink 2.2 中引進的重大變更和棄用摘要。如需介紹這些重大變更的 [Apache Flink 2.0 完整版本備註](#)，請參閱 [Apache Flink 2.0 版本備註](#)。

Flink API 和語言移除

DataSet API 已移除

- 用於批次處理的舊版 DataSet API 已在 Flink 2.0+ 中完全移除。所有批次處理現在必須使用統一的 DataStream API。
- 使用 DataSet API 的應用程式必須先遷移至 DataStream API，才能升級。請參閱 [DataSet 至 DataStream 轉換的 Apache Flink 遷移指南](#)

Java 11 和 Python 3.8 已移除

- Java 11 支援完全移除；Java 17 是預設和建議的執行時間。
- Python 3.8 支援已移除；Python 3.12 現在為預設值。

已移除舊版連接器類別

- 舊版 SourceFunction 和 SinkFunction 界面已由新的統一來源 (FLIP-27) 和接收器 (FLIP-143) APIs 取代，可更好地支援受限/未受限的雙重性、改善檢查點協調，以及更簡潔的程式設計模型。
- 對於 Kinesis Data Streams，請使用 KinesisStreamsSink 來自的 KinesisStreamsSource 和 `flink-connector-aws-kinesis-streams:6.0.0-2.0`。

已移除 Scala API

- Flink Scala API 已移除。Flink 的 Java API 現在是 JVM 型應用程式的單一支援 API。
- 如果您的應用程式是以 Scala 撰寫，您仍然可以使用來自 Scala 程式碼的 Flink 的 Java API，主要變更是 Scala 特定的包裝函式和隱含轉換不再可用。如需更新 Scala 應用程式的詳細資訊，請參閱 [升級應用程式和 Flink 版本](#)。

狀態相容性考量事項

- 從 2.24 版升級至 5.6 版的 Kryo 序列化程式可能會導致狀態相容性問題。
- 具有集合 (HashMap、ArrayList、HashSet) POJOs 可能有狀態相容性問題。

- Avro 和 Protobuf 序列化不受影響。
- [Flink 2.2 升級的狀態相容性指南](#) 如需分類應用程式風險層級的詳細評估，請參閱。

支援 Apache Flink 2.2 功能

執行期和語言支援

功能	Description	文件
Java 17 執行期	Java 17 現在是預設和建議的執行時間；Java 11 支援已移除。	Java 相容性
Python 3.12 支援	Python 3.12 現在支援；Python 3.8 支援已移除。	PyFlink 文件

狀態管理和效能

功能	Description	文件
RocksDB 8.10.0	透過 RocksDB 升級改善 I/O 效能。	狀態後端
序列化改進	Map、List、Set；Kryo 專用序列化程式從 2.24 升級到 5.6。	類型序列化

SQL 和資料表 API 功能

功能	Description	文件
VARIANT 資料類型	原生支援半結構化資料 (JSON)，無需重複字串剖析。	資料類型
Delta 加入	僅維護每個金鑰的最新版本，以減少串流聯結的狀態需求；	聯結

功能	Description	文件
	需要客戶管理的基礎設施（例如 Apache Fluss）。	
StreamingMultiJoinOperator	以單一運算子執行多方聯結，消除中繼具體化。	FLIP-516
ProcessTableFunction (PTF)	使用每個金鑰狀態和計時器，在 SQL 中直接啟用具狀態的事件驅動邏輯。	使用者定義的函數
ML_PREDICT 函數	直接從 SQL 在串流/批次資料表上呼叫已註冊的 ML 模型。要求客戶綁定 ModelProvider 實作（例如 flink-model-openai）。ModelProvider 程式庫並非由 Managed Service for Apache Flink 運送。	ML 預測
模型 DDL	使用 CREATE MODEL 陳述式將 ML 模型定義為一級目錄物件。	CREATE 陳述式
向量搜尋	Flink SQL API 支援搜尋向量資料庫。目前沒有可用的開放原始碼 VectorSearchTableSource 實作；客戶必須提供自己的實作。	Flink SQL

DataStream API 功能

功能	Description	文件
FLIP-27 來源 API	取代舊版 SourceFunction 的新統一來源界面。	來源

功能	Description	文件
FLIP-143 接收器 API	取代舊版 SinkFunction 的新統一接收器界面。	接收器
非同步 Python DataStream	使用 AsyncFunction 在 Python DataStream API 中非封鎖 I/O 操作。	非同步 I/O

連接器可用性

升級至 Flink 2.2 時，您也需要將連接器相依性更新為與 Flink 2.2 執行時間相容的版本。Flink 連接器獨立於 Flink 執行期發行，而且並非所有連接器都具有與 Flink 2.2 相容的版本。下表摘要說明 Amazon Managed Service for Apache Flink 中常用連接器的可用性：

Flink 2.2 的連接器可用性

連接器	Flink 1.20 版本	Flink 2.0+ 版本	備註
Apache Kafka	flink-connector-kafka 3.4.0-1.20	flink-connector-kafka 4.0.0-2.0	建議用於 Flink 2.2
Kinesis Data Streams (來源)	flink-connector-kinesis 5.0.0-1.20	flink-connector-aws-kinesis-streams 6.0.0-2.0	建議用於 Flink 2.2
Kinesis Data Streams (接收器)	flink-connector-aws-kinesis-streams 5.1.0-1.20	flink-connector-aws-kinesis-streams 6.0.0-2.0	建議用於 Flink 2.2
Amazon Data Firehose	flink-connector-aws-kinesis-firehose 5.1.0-1.20	flink-connector-aws-kinesis-firehose 6.0.0-2.0	與 Flink 2.0 相容
Amazon DynamoDB	flink-connector-dynamodb 5.1.0-1.20	flink-connector-dynamodb 6.0.0-2.0	與 Flink 2.0 相容
Amazon SQS	flink-connector-sqs 5.1.0-1.20	flink-connector-sqs 6.0.0-2.0	與 Flink 2.0 相容

連接器	Flink 1.20 版本	Flink 2.0+ 版本	備註
FileSystem (S3、HDFS)	與 Flink 綁定	與 Flink 綁定	內建於 Flink 分佈中 — 一律可用
JDBC	flink-connector-jdbc 3.3.0-1.20	尚未針對 2.x 發行	沒有可用的 Flink 2.x 相容版本
OpenSearch	flink-connector-opensearch 1.2.0-1.19	尚未針對 2.x 發行	沒有可用的 Flink 2.x 相容版本
Elasticsearch	僅限舊版連接器	尚未針對 2.x 發行	考慮遷移至 OpenSearch 連接器
Amazon Managed Service for Prometheus	flink-connector-prometheus 1.0.0-1.20	尚未針對 2.x 發行	沒有可用的 Flink 2.x 相容版本

- 如果您的應用程式依賴的連接器尚未有 Flink 2.x 版本，您有兩個選項：等待連接器發行相容的版本，或評估您是否可以使用替代版本取代它（例如，使用 JDBC 目錄或自訂接收器）。
- 更新連接器版本時，請注意成品名稱變更 — 某些連接器在主要版本之間重新命名（例如，在某些中繼版本 `flink-connector-aws-kinesis-firehose` 和 `flink-connector-aws-firehose` 中，Firehose 連接器從 `flink-connector-aws-kinesis-firehose` 變更為 `flink-connector-aws-firehose`）。
- 請務必檢查 [Amazon Managed Service for Apache Flink 連接器文件](#)，了解目標執行時間中支援的確切成品名稱和版本。

不支援和實驗性功能

Amazon Managed Service for Apache Flink 2.2 不支援下列功能：

- 具體化資料表：持續維護、可查詢的資料表快照。
- 自訂遙測變更：自訂指標報告程式和遙測組態。
- ForSt 狀態後端：分解狀態儲存（在開放原始碼中進行實驗）。
- Java 21：開放原始碼中的實驗性支援，在 Managed Service for Apache Flink 中不支援。

已知問題

Amazon Managed Service for Apache Flink Studio

Amazon Managed Service for Apache Flink 中的 Flink 2.2 不支援 Studio 應用程式。如需詳細資訊，請參閱[建立 Studio 筆記本](#)。

Kinesis 連接器 EFO

- 當 Kinesis 串流進行重新分片時，連接器 v5.0.0 和 v6.0.0 中引入KinesisStreamsSource搭配 EFO (增強廣播/SubscribeToShard) 路徑使用的應用程式可能會失敗。這是社群中的已知問題。如需詳細資訊，請參閱 [FLINK-37648](#)。
- 如果 Flink 應用程式處於背壓狀態，則連接器 v5.0.0 和 v6.0.0 中引入KinesisStreamsSource的搭配 EFO (增強廣播/SubscribeToShard) 路徑使用的應用程式KinesisStreamsSink可能會遇到死結，導致一個或多個 TaskManagers 中的資料處理完全停止。需要強制停止操作和啟動應用程式操作才能復原應用程式。這是社群中已知問題的子案例。如需詳細資訊，請參閱 [FLINK-34071](#)。

升級體驗

Amazon Managed Service for Apache Flink 支援就地版本升級，保留您的應用程式組態、日誌、指標、標籤，以及如果狀態和二進位檔相容，則為您的應用程式狀態。如需逐步說明，請參閱[升級到 Flink 2.2：完成指南](#)。

如需在升級期間評估狀態相容性風險及處理不相容狀態的指引，請參閱[Flink 2.2 升級的狀態相容性指南](#)。

後續步驟

- 第一次使用 Flink 2.2？如需詳細的 Apache Flink 2.2 文件，請參閱[Apache Flink 2.2 文件](#)。
- 規劃升級？請參閱[升級到 Flink 2.2：完成指南](#)
- 狀態相容性問題？請參閱[Flink 2.2 升級的狀態相容性指南](#)

如有疑問或問題，請參閱[Managed Service for Apache Flink 故障診斷](#)或聯絡 AWS Support。

Amazon Managed Service for Apache Flink 1.20

Managed Service for Apache Flink 現在支援 Apache Flink 1.20.0 版。本節將介紹 Managed Service for Apache Flink 1.20.0 支援的主要新功能和變更。Apache Flink 1.20 預期是最後一個 1.x 版本和

Flink 長期支援 (LTS) 版本。如需詳細資訊，請參閱 [FLIP-458：Apache Flink 1.x Line 最終版本的長期支援](#)。

Note

如果您使用舊版支援的 Apache Flink，並想要將現有應用程式升級至 Apache Flink 1.20.0，則可以使用就地 Apache Flink 版本升級來執行此操作。如需詳細資訊，請參閱[針對 Apache Flink 使用就地版本升級](#)。透過就地版本升級，您可以針對跨 Apache Flink 版本的單一 ARN 保留應用程式可追蹤性，包括快照、日誌、指標、標籤、Flink 組態等。

支援的功能

Apache Flink 1.20.0 在 SQL APIs、DataStream APIs 和 Flink 儀表板中引入了改進功能。

支援的功能和相關文件

支援的功能	Description	Apache Flink 文件參考
新增 DISTRIBUTED BY 子句	許多 SQL 引擎會公開 Partitioning、Bucketing 或的概念 Clustering。Flink 1.20 將 Bucketing 的概念引入 Flink。	FLIP-376：新增 DISTRIBUTED BY 子句
DataStream API：支援完整分割區處理	Flink 1.20 引入了透過 FullPartitionWindow API 在非金鑰串流上彙總的內建支援。	FLIP-380：支援非金鑰 DataStream 上的完整分割區處理
在 Flink Dashboard 上顯示資料扭曲分數	Flink 1.20 儀表板現在會顯示資料扭曲侵權。Flink 任務圖表 UI 上的每個運算子都會顯示額外的資料扭曲分數。	FLIP-418：在 Flink Dashboard 上顯示資料扭曲分數

如需 Apache Flink 1.20.0 版本文件，請參閱 [Apache Flink 文件 1.20.0](#) 版。如需 Flink 1.20 版本備註，請參閱[版本備註 - Flink 1.20](#)

元件

Flink 1.20 元件

元件	版本
Java	11 (建議使用)
Python	3.11
Kinesis Data Analytics Flink 執行期 (aws-kinesisanalytics-runtime)	1.2.0
連接器	如需可用連接器的資訊，請參閱 Apache Flink 連接器 。
Apache Beam (僅限於 Beam 應用程式)	沒有適用於 Flink 1.20 的相容 Apache Flink Runner。如需詳細資訊，請參閱 Flink 版本相容性 。

已知問題

Apache 光束

Apache Beam 中目前沒有適用於 Flink 1.20 的相容 Apache Flink Runner。如需詳細資訊，請參閱 [Flink 版本相容性](#)。

Amazon Managed Service for Apache Flink Studio

Amazon Managed Service for Apache Flink Studio 使用 Apache Zeppelin 筆記本提供單一介面開發體驗，用於開發、偵錯程式碼和執行 Apache Flink 串流處理應用程式。Zeppelin 的 Flink 解譯器需要升級，才能支援 Flink 1.20。此工作是與 Zeppelin 社群一起排程。我們會在工作完成時更新這些備註。您可以繼續使用 Flink 1.15 搭配 Amazon Managed Service for Apache Flink Studio。如需詳細資訊，請參閱[建立 Studio 筆記本](#)。

反向移植錯誤修正

Amazon Managed Service for Apache Flink 後端連接埠修正來自 Flink 社群的重大問題。以下是我們已向後移植的錯誤修正清單：

反向移植錯誤修正

Apache Flink JIRA 連結	Description
FLINK-35886	此修正解決了當子任務受到背壓/封鎖時，導致浮水印閒置逾時計算不正確的問題。

Amazon Managed Service for Apache Flink 1.19

Managed Service for Apache Flink 現在支援 Apache Flink 1.19.1 版。本節將介紹 Managed Service for Apache Flink 1.19.1 支援的主要新功能和變更。

Note

如果您使用的是舊版支援的 Apache Flink，而且想要將現有應用程式升級至 Apache Flink 1.19.1，則可以使用就地 Apache Flink 版本升級來執行此操作。如需詳細資訊，請參閱[針對 Apache Flink 使用就地版本升級](#)。透過就地版本升級，您可以針對跨 Apache Flink 版本的單一 ARN 保留應用程式可追蹤性，包括快照、日誌、指標、標籤、Flink 組態等。

支援的功能

Apache Flink 1.19.1 引進 SQL API 的改進，例如具名參數、自訂來源平行處理，以及各種 Flink 運算子的不同狀態 TTLs。

支援的功能和相關文件

支援的功能	Description	Apache Flink 文件參考
SQL API：支援使用 SQL 提示設定不同的狀態 TTLs	使用者現在可以在串流定期聯結和群組彙總上設定狀態 TTL。	FLIP-373：使用 SQL Hint 設定不同的狀態 TTLs
SQL API：支援函數和呼叫程序的具名參數	使用者現在可以在函數中使用具名參數，而不是依賴參數的順序。	FLIP-378：支援函數和呼叫程序的具名參數
SQL API：設定 SQL 來源的平行處理	使用者現在可以指定 SQL 來源的平行處理。	FLIP-367：支援設定資料表/SQL 來源的平行處理

支援的功能	Description	Apache Flink 文件參考
SQL API：支援工作階段視窗 TVF	使用者現在可以使用工作階段視窗的資料表值函數。	FLINK-24024：支援工作階段視窗 TVF
SQL API：Window TVF 彙總支援 Changelog 輸入	使用者現在可以在 changelog 輸入上執行視窗彙總。	FLINK-20281：視窗彙總支援 changelog 串流輸入
支援 Python 3.11	Flink 現在支援 Python 3.11，比 Python 3.10 快 10-60%。如需詳細資訊，請參閱 Python 3.11 中的最新消息。	FLINK-33030：新增 python 3.11 支援
提供 TwoPhaseCommitting 接收器的指標	使用者可以檢視兩個階段遞交目的地中遞交者狀態的統計資料。	FLIP-371：提供在 TwoPhaseCommittingSink 中建立發射器的初始化內容
用於任務重新啟動和檢查點的追蹤報告程式	使用者現在可以監控檢查點持續時間和周轉趨勢的追蹤。在 Amazon Managed Service for Apache Flink 中，我們預設啟用 Slf4j 追蹤報告程式，讓使用者可以透過應用程式 CloudWatch Logs 監控檢查點和任務追蹤。	FLIP-384：介紹 TraceReporter，並使用它來建立檢查點和復原追蹤

Note

您可以透過提交[支援案例](#)來選擇加入下列功能：

選擇加入功能和相關文件

選擇加入功能	Description	Apache Flink 文件參考
當來源正在處理待處理項目時，支援使用較大的檢查點間隔	這是選擇加入功能，因為使用者必須針對其特定任務需求調整組態。	FLIP-309：支援在來源處理待處理項目時使用較大的檢查點間隔
將 System.out 和 System.err 重新導向至 Java 日誌	這是選擇加入功能。在 Amazon Managed Service for Apache Flink 上，預設行為是忽略 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。	FLIP-390：支援系統輸出和錯誤重新導向至 LOG 或捨棄

如需 Apache Flink 1.19.1 版本文件，請參閱 [Apache Flink 文件 1.19.1 版](#)。

Amazon Managed Service for Apache Flink 1.19.1 的變更

預設啟用記錄追蹤報告程式

Apache Flink 1.19.1 引進檢查點和復原追蹤，讓使用者能夠更好地偵錯檢查點和任務復原問題。在 Amazon Managed Service for Apache Flink 中，這些追蹤會登入 CloudWatch 日誌串流，允許使用者細分任務初始化所花費的時間，並記錄檢查點的歷史大小。

預設重新啟動策略現在為指數延遲

在 Apache Flink 1.19.1 中，指數延遲重新啟動策略有大幅改善。在 Flink 1.19.1 之後的 Amazon Managed Service for Apache Flink 中，Flink 任務預設使用指數延遲重新啟動策略。這表示使用者任務將從暫時性錯誤中更快速復原，但如果任務重新啟動持續存在，則不會讓外部系統超載。

反向移植錯誤修正

Amazon Managed Service for Apache Flink 後端連接埠修正來自 Flink 社群的重大問題。這表示執行時間與 Apache Flink 1.19.1 版不同。以下是我們已向後移植的錯誤修正清單：

反向移植錯誤修正

Apache Flink JIRA 連結	Description
FLINK-35531	此修正解決了 1.17.0 中引入的效能迴歸，這會導致 HDFS 的寫入速度變慢。
FLINK-35157	此修正解決了當具有浮水印對齊的來源遇到已完成子任務時，Flink 任務停滯的問題。
FLINK-34252	此修正解決了產生浮水印時導致 IDLE 浮水印狀態錯誤的問題。
FLINK-34252	此修正透過減少系統呼叫來解決浮水印產生期間的效能迴歸。
FLINK-33936	此修正解決了資料表 API 上的微批次彙總期間重複記錄的問題。
FLINK-35498	此修正解決了在資料表 API UDFs 中定義具名參數時，引數名稱衝突的問題。
FLINK-33192	此修正解決了由於不正確的計時器清除，視窗運算子中狀態記憶體流失的問題。
FLINK-35069	此修正解決了 Flink 任務在視窗結束時卡住觸發計時器的問題。
FLINK-35832	此修正解決了 IFNULL 傳回不正確結果的問題。
FLINK-35886	此修正可解決將背壓任務視為閒置時的問題。

元件

元件	版本
Java	11 (建議使用)
Python	3.11

元件	版本
Kinesis Data Analytics Flink 執行期 (aws-kinesisanalytics-runtime)	1.2.0
連接器	如需可用連接器的資訊，請參閱 Apache Flink 連接器 。
Apache Beam (僅限於 Beam 應用程式)	從 2.61.0 版開始。如需詳細資訊，請參閱 Flink 版本相容性 。

已知問題

Amazon Managed Service for Apache Flink Studio

Studio 使用 Apache Zeppelin 筆記本提供單一介面開發體驗，用於開發、偵錯程式碼和執行 Apache Flink 串流處理應用程式。Zeppelin 的 Flink 解譯器需要升級，才能支援 Flink 1.19。此工作是與 Zeppelin 社群排程的，我們將在完成時更新這些備註。您可以繼續使用 Flink 1.15 搭配 Amazon Managed Service for Apache Flink Studio。如需詳細資訊，請參閱[建立 Studio 筆記本](#)。

Amazon Managed Service for Apache Flink 1.18

Managed Service for Apache Flink 現在支援 Apache Flink 1.18.1 版。了解 Managed Service for Apache Flink 支援 Apache Flink 1.18.1 推出的重要新功能和變更。

Note

如果您使用的是舊版支援的 Apache Flink，而且想要將現有應用程式升級至 Apache Flink 1.18.1，則可以使用就地 Apache Flink 版本升級來執行此操作。透過就地版本升級，您可以保留跨 Apache Flink 版本的單一 ARN 的應用程式可追蹤性，包括快照、日誌、指標、標籤、Flink 組態等。您可以在 RUNNING 和 READY 狀態使用此功能。如需詳細資訊，請參閱[針對 Apache Flink 使用就地版本升級](#)。

Apache Flink 文件參考支援的功能

支援的功能	Description	Apache Flink 文件參考
Opensearch 連接器	此連接器包含提供at-least-once保證的接收器。	github : Opensearch Connector
Amazon DynamoDB 連接器	此連接器包含提供at-least-once保證的接收器。	Amazon DynamoDB Sink
MongoDB 連接器	此連接器包含提供at-least-once保證的來源和接收器。	MongoDB 連接器
使用 Flink 規劃器解耦 Hive	您可以直接使用 Hive 方言，無需額外的 JAR 交換。	FLINK-26603 : 使用 Flink 規劃器解耦 Hive
根據預設，在 RocksDBWriteBatchWrapper 中停用 WAL	這可提供更快的復原時間。	FLINK-32326 : 預設在 RocksDBWriteBatchWrapper 中停用 WAL
改善啟用浮水印對齊時的浮水印彙總效能	改善啟用浮水印對齊時的浮水印彙總效能，並新增相關基準。	FLINK-32524 : 浮水印彙總效能
讓浮水印對齊準備好供生產使用	消除大型任務超載 JobManager 的風險	FLINK-32548 : 準備好浮水印對齊
非同步接收器的可設定 RateLimitingStrategy	RateLimitingStrategy 可讓您設定要擴展的內容、擴展的時間，以及擴展的數量。	FLIP-242 : 引入適用於非同步接收器的可設定 RateLimitingStrategy
大量擷取資料表和資料欄統計資料	改善查詢效能。	FLIP-247 : 大量擷取指定分割區的資料表和資料欄統計資料

如需 Apache Flink 1.18.1 版本文件，請參閱 [Apache Flink 1.18.1 版本公告](#)。

Amazon Managed Service for Apache Flink with Apache Flink 1.18 的變更

Akka 取代為 Pekko

Apache Flink 1.18 中的 Apache Flink 將 Akka 取代為 Pekko。Managed Service for Apache Flink 1.18.1 及更新版本完全支援此變更。由於此變更，您不需要修改應用程式。如需詳細資訊，請參閱 [FLINK-32468 : Replace Akka by Pekko](#)。

在執行緒模式中支援 PyFlink 執行期執行

此 Apache Flink 變更為 Pyflink 執行期架構 程序模式引進新的執行模式。程序模式現在可以在相同的執行緒中執行 Python 使用者定義的函數，而不是個別的程序。

反向移植錯誤修正

Amazon Managed Service for Apache Flink 後端連接埠修正來自 Flink 社群的重大問題。這表示執行時間與 Apache Flink 1.18.1 版不同。以下是我們已向後移植的錯誤修正清單：

反向移植錯誤修正

Apache Flink JIRA 連結	Description
FLINK-33863	此修正可解決壓縮快照的狀態還原失敗的問題。
FLINK-34063	此修正可解決啟用快照壓縮時來源運算子遺失分割的問題。Apache Flink 為所有檢查點和儲存點提供選用的壓縮（預設：關閉）。Apache Flink 已識別 Flink 1.18.1 中的錯誤，其中在啟用快照壓縮時，無法正確還原運算子狀態。這可能會導致資料遺失或無法從檢查點還原。
FLINK-35069	此修正解決了 Flink 任務在視窗結束時卡住觸發計時器的問題。
FLINK-35097	此修正以原始格式解決資料表 API 檔案系統連接器中重複記錄的問題。
FLINK-34379	此修正解決了啟用動態資料表篩選時 OutOfMemoryError 的問題。
FLINK-28693	此修正解決了如果浮水印具有 columnBy 表達式，資料表 API 無法產生圖形的問題。
FLINK-35217	此修正可解決特定 Flink 任務失敗模式期間檢查點損毀的問題。

元件

元件	版本
Java	11 (建議使用)
Scala	自 1.15 版以來，Flink 與 Scala 無關。作為參考，MSF Flink 1.18 已針對 Scala 3.3 (LTS) 驗證。
Managed Service for Apache Flink 執行期 (aws-kinesisanalytics-runtime)	1.2.0
AWS Kinesis 連接器 (flink-connector-kinesis) 【來源】	4.2.0-1.18
AWS Kinesis 連接器 (flink-connector-kinesis) 【Sink】	4.2.0-1.18
Apache Beam (僅限於 Beam 應用程式)	從 2.57.0 版開始。如需詳細資訊，請參閱 Flink 版本相容性 。

已知問題

Amazon Managed Service for Apache Flink Studio

Studio 使用 Apache Zeppelin 筆記本提供單一介面開發體驗，用於開發、偵錯程式碼和執行 Apache Flink 串流處理應用程式。Zeppelin 的 Flink 解譯器需要升級，才能支援 Flink 1.18。此工作是與 Zeppelin 社群一起排程的，我們將在完成時更新這些備註。您可以繼續使用 Flink 1.15 搭配 Amazon Managed Service for Apache Flink Studio。如需詳細資訊，請參閱[建立 Studio 筆記本](#)。

當子任務受到背壓時，浮水印閒置不正確

當子任務受到背壓時，浮水印產生中存在已知問題，該問題已從 Flink 1.19 及更新版本修正。當 Flink 任務圖表受到背壓時，這可能會在延遲記錄的數量中顯示為峰值。我們建議您升級至最新的 Flink 版本，以插入此修正。如需詳細資訊，請參閱當[子任務受到背壓/封鎖時，浮水印閒置逾時會計不正確](#)。

Amazon Managed Service for Apache Flink 1.15

Managed Service for Apache Flink 支援 Apache 1.15.2 中的下列新功能：

功能	Description	Apache FLIP 參考
非同步接收器	用於建置非同步目的地的 AWS 貢獻架構，可讓開發人員建置自訂 AWS 連接器，前次工作不到一半。如需詳細資訊，請參閱 通用非同步基本接收器 。	FLIP-171：非同步接收器 。
Kinesis Data Firehose 接收器	AWS 已使用非同步架構貢獻新的 Amazon Kinesis Firehose Sink。	Amazon Kinesis Data Firehose 接收器 。
使用儲存點停止	「使用儲存點停止」可確保乾淨利落的停止操作，最重要的是為依賴它們的客戶提供了僅支援一次的語義。	FLIP-34：使用儲存點終止/暫停作業 。
Scala 解耦	使用者現在可以利用任何 Scala 版本的 Java API，包括 Scala 3。客戶需要將所選擇的 Scala 標準程式庫綁定在他們的 Scala 應用程式中。	FLIP-28：將移除 flink-table 的 Scala 相依性作為長期目標 。
Scala	請參閱上面的 Scala 解耦	FLIP-28：將移除 flink-table 的 Scala 相依性作為長期目標 。
統一的連接器指標	Flink 針對作業、任務和運算子擁有 已定義的標準指標 。Managed Service for Apache Flink 將繼續支援接收器和來源指標，並在 1.15 版中為可用性指標同時引入了 numRestarts 與 fullRestarts。	FLIP-33：將連接器指標標準化 和 FLIP-179：公開標準化的運算子指標 。

功能	Description	Apache FLIP 參考
檢查點已完成的任務	此功能在 Flink 1.15 中預設為啟用，即使作業圖表的某些部分已完成處理所有資料 (如果包含綁定的 (批次) 來源，可能會發生此情況)，仍可以繼續執行檢查點。	FLIP-147：在任務完成後支援檢查點。

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

Studio 筆記本

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

EFO 連接器

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

Scala 解耦

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

Kinesis Data Firehose 接收器

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

Kafka 連接器

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

元件

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

已知問題

代理程式重新啟動後，Kafka 遞交檢查點會重複失敗

Flink 1.15 版中的 Apache Kafka 連接器存在已知的開放原始碼 Apache Flink 問題，因為 Kafka 用戶端 2.8.1 中的重大開放原始碼 Kafka 用戶端錯誤。如需詳細資訊，請參閱 [Kafka Commit on checkpointing 在代理程式重新啟動後重複失敗](#)，且在 [commitOffsetAsync 例外狀況後，KafkaConsumer 無法復原與群組協調器的連線](#)。

為了避免此問題，建議您在 Amazon Managed Service for Apache Flink 中使用 Apache Flink 1.18 或更新版本。

Managed Service for Apache Flink 的早期版本資訊

Note

Apache Flink 社群已支援 Apache Flink 1.6、1.8 和 1.11 版超過三年。我們已在 2024 年 6 月和 2024 年 10 月發出此變更的通知，現在將在 Amazon Managed Service for Apache Flink 中結束對這些版本的支援。

- 2025 年 7 月 14 日，我們將停止您的應用程式，並將其置於就緒狀態。屆時，您將能夠重新啟動應用程式，並根據服務限制繼續正常使用應用程式。

- 從 2025 年 7 月 28 日起，我們將停用啟動應用程式的功能。您目前無法啟動或操作 Flink 1.6 版應用程式。

建議您使用 Apache Flink 1.6、1.8 或 1.11 版，立即將任何現有的應用程式升級至 Apache Flink 1.20 版。這是最新支援的 Flink 版本。您可以使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能來升級應用程式。如需詳細資訊，請參閱[針對 Apache Flink 使用就地版本升級](#)。

如果您有其他問題或疑慮，您可以聯絡 [AWS 支援](#)。

Note

Apache Flink 1.13 版已超過三年未被 Apache Flink 社群支援。我們現在計劃在 2025 年 10 月 16 日終止 Amazon Managed Service for Apache Flink 對此版本的支援。在此日期之後，您將無法再使用 Amazon Managed Service for Apache Flink 1.13 版建立、啟動或執行應用程式。您可以使用 Managed Service for Apache Flink 中的就地版本升級功能，以狀態升級應用程式。如需詳細資訊，請參閱[針對 Apache Flink 使用就地版本升級](#)。

Managed Service for Apache Flink 支援 1.15.2 版，但 Apache Flink 社群不再支援。

本主題包含下列章節：

- [將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用](#)
- [使用 Apache Flink 1.8.2 建置應用程式](#)
- [使用 Apache Flink 1.6.2 建置應用程式](#)
- [升級應用程式](#)
- [Apache Flink 1.6.2 和 1.8.2 中的可用連接器](#)
- [入門：Flink 1.13.2](#)
- [入門：Flink 1.11.1 - 棄用](#)
- [入門：Flink 1.8.2 - 棄用](#)
- [入門：Flink 1.6.2 - 棄用](#)
- [Managed Service for Apache Flink 的舊版（舊版）範例](#)

將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用

1.11 版之前的 Apache Flink 中不包含 Apache Flink Kinesis 串流連接器。若要讓應用程式能夠將 Apache Flink Kinesis 連接器與先前版本的 Apache Flink 搭配使用，必須下載、編譯並安裝該應用程式所使用的 Apache Flink 版本。此連接器用於取用作為應用程式來源的 Kinesis 串流中的資料，或將資料寫入作為應用程式輸出的 Kinesis 串流。

Note

確保正在使用 [KPL 0.14.0 版本](#) 或更高版本建置連接器。

若要下載並安裝 Apache Flink 1.8.2 版來源程式碼，請執行下列動作：

1. 確保已安裝 [Apache Maven](#)，並且 JAVA_HOME 環境變數指向 JDK 而不是 JRE。您可以使用以下命令來測試 Apache Maven 安裝：

```
mvn -version
```

2. 下載 Apache Flink 版本 1.8.2 來源程式碼：

```
wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
```

3. 解壓縮 Apache Flink 來源程式碼：

```
tar -xvf flink-1.8.2-src.tgz
```

4. 切換到 Apache Flink 來源程式碼目錄：

```
cd flink-1.8.2
```

5. 編譯並安裝 Apache Flink：

```
mvn clean install -Pinclude-kinesis -DskipTests
```

Note

如果您在 Microsoft 視窗中編譯 Flink，則需要添加 `-Drat.skip=true` 參數。

使用 Apache 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
用於 Flink 執行期 (aws-kinesisanalytics-runtime) 的 Managed Service for Apache Flink	1.0.1
Managed Service for Apache Flink Flink 連接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1

若要編譯使用 Apache Flink 1.8.2 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[建立應用程式](#)。

使用 Apache 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
用於 Flink 執行期 (aws-kinesisanalytics-runtime) 的 Managed Service for Apache Flink	1.0.1
Managed Service for Apache Flink Flink 連接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1
Apache Beam	不支援用於 Apache Flink 1.6.2。

Note

使用 Managed Service for Apache Flink 執行期 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 [建立應用程式](#)。

升級應用程式

若要升級 Amazon Managed Service for Apache Flink 應用程式的 Apache Flink 版本，請使用 AWS CLI、AWS SDK CloudFormation 或來使用就地 Apache Flink 版本升級功能 AWS 管理主控台。如需詳細資訊，請參閱 [針對 Apache Flink 使用就地版本升級](#)。

您可以搭配 Amazon Managed Service for Apache Flink 處於 READY 或 RUNNING 狀態的任何現有應用程式使用此功能。

Apache Flink 1.6.2 和 1.8.2 中的可用連接器

Apache Flink 架構包含用於存取各種來源之資料的連接器。

- 如需 Apache Flink 1.6.2 架構中可用連接器的相關資訊，請參閱 [Apache Flink 文件 \(1.6.2\)](#) 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 \(1.8.2\)](#)。

入門：Flink 1.13.2

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

主題

- [Managed Service for Apache Flink 應用程式的元件](#)
- [完成練習的先決條件](#)
- [步驟 1：設定 AWS 帳戶並建立管理員使用者](#)
- [下一步驟](#)
- [步驟 2：設定 AWS Command Line Interface \(AWS CLI\)](#)
- [步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)
- [步驟 4：清除 AWS 資源](#)
- [步驟 5：後續步驟](#)

Managed Service for Apache Flink 應用程式的元件

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

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[新增串流資料來源](#)。

- **運算子**：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱[運算子](#)。
- **接收器**：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[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 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時，AWS 帳戶，AWS 帳戶根使用者會建立。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 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者[AWS 管理主控台](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄做為身分來源的教學課程，請參閱 AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

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 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 • AWS SDKs請參閱 AWS SDKs和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface

哪個使用者需要程式設計存取權？	到	根據
		《使用者指南》中的設定 AWS CLI 要使用 AWS IAM Identity Center的。 AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中 將臨時登入資料與 AWS 資源搭配使用 的指示。
IAM	(不建議使用) 使用長期登入資料來簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 - AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的使用長期憑證進行身分驗證。 - 對於 AWS APIs，請參閱《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

下一步驟

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

下一步驟

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

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

在此步驟中，您會下載並設定 AWS CLI 以搭配 Managed Service for Apache Flink 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 AWS CLI 安裝，您可能需要升級以取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[安裝 AWS Command Line Interface](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本教學課程中的練習需要下列 AWS CLI 版本 或更新版本：

```
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 AWS CLI config 檔案中為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[具名描述檔](#)。

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

如需可用 AWS 區域的清單，請參閱 [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 帳戶和 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

[步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)

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

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

本節包含下列步驟：

- [建立兩個 Amazon Kinesis 資料串流](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查 Apache Flink 串流 Java 程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [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 命令。

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

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

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
import datetime
import json
import random
import boto3
STREAM_NAME = "ExampleInputStream"
def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

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

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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[使用執行期屬性](#)。

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，您可以指定程式碼內容類型 (JAR 或 ZIP)。

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

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

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

上傳應用程式的程式碼

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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-*<username>* 儲存貯體，並選擇上傳。
8. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

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

Note

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(AWS CLI\)](#)

建立並執行應用程式（主控台）

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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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**)。

JSON

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

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

5. 在監控下，確保監控指標層級設為應用程式。
6. 針對 CloudWatch 記錄，選取啟用核取方塊。
7. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

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，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，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

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

JSON

```
{
  "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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

Note

若要存取其他 Amazon 服務，您可以使用適用於 Java 的 AWS SDK。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 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的[建立 IAM 角色 \(主控台\)](#)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。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 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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 [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 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱 [the section called “在 Managed Service for Apache Flink 中設定應用程式記錄”](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

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

2. 使用前述請求執行 [UpdateApplication](#) 動作以更新環境屬性：

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

更新應用程式的程式碼

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，您可以使用 [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 儲存貯體中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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"
        }
      }
    }
  }
}
```

下一步驟

[步驟 4：清除 AWS 資源](#)

步驟 4：清除 AWS 資源

本節包含清除入門教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Kinesis Data Streams 面板中，選擇 ExampleInputStream。
3. 在 ExampleInputStream 頁面，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

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。

- 適用於 [Amazon Kinesis 的 AWS 串流資料解決方案](#)：適用於 Amazon Kinesis 的 AWS 串流資料解決方案會自動設定所需的 AWS 服務，以輕鬆擷取、儲存、處理和交付串流資料。該解決方案提供了多種解決串流資料使用案例的選項。Managed Service for Apache Flink 選項提供端對端串流 ETL 範例，展示真實世界的應用程式如何根據模擬的紐約計程車資料執行分析作業。解決方案會設定所有必要 AWS 的資源，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](#)：適用於 AWS Amazon MSK 的串流資料解決方案提供 AWS CloudFormation 範本，其中資料流經生產者、串流儲存體、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。
- [Amazon Managed Service for Apache Flink 研討會](#)：在本研討會中，您將建置 end-to-end 串流架構，以近乎即時的方式擷取、分析和視覺化串流資料。您著手改善紐約市一家計程車公司的運營。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 Flink Managed Service for Apache Flink 中使用 Flink 1.15.2。

入門：Flink 1.11.1 - 棄用

Note

Apache Flink 社群已超過三年不支援 Apache Flink 1.6、1.8 和 1.11 版。我們計劃於 2024 年 11 月 5 日在 Amazon Managed Service for Apache Flink 中棄用這些版本。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。此時您可以繼續執行現有的應用程式。您可以使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，以狀態升級應用程式。如需詳細資訊，請參閱 [針對 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 Managed Service for Apache Flink 中的 DataStream API](#)教學課程版本。

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

主題

- [Managed Service for Apache Flink 應用程式的元件](#)
- [完成練習的先決條件](#)
- [步驟 1：設定 AWS 帳戶並建立管理員使用者](#)

- [步驟 2：設定 AWS Command Line Interface \(AWS CLI\)](#)
- [步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)
- [步驟 4：清除 AWS 資源](#)
- [步驟 5：後續步驟](#)

Managed Service for Apache Flink 應用程式的元件

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

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[新增串流資料來源](#)。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱[運算子](#)。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[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 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。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 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者[AWS 管理主控台](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

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 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface

哪個使用者需要程式設計存取權？	到	根據
		《使用者指南》中的登入以進行 AWS 本機開發。 • AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的設定 AWS CLI 要使用 AWS IAM Identity Center的。 • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDK 和工具參考指南中的 SDKs IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中將臨時登入資料與 AWS 資源搭配使用的指示。

哪個使用者需要程式設計存取權？	到	根據
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 使用 IAM 使用者憑證進行身分驗證。 - AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的 使用長期憑證進行身分驗證。 - 對於 AWS APIs，請參閱《IAM 使用者指南》中的 管理 IAM 使用者的存取金鑰。

下一步驟

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

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

在此步驟中，您會下載並設定 AWS CLI 以搭配 Managed Service for Apache Flink 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 AWS CLI 安裝，您可能需要升級以取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本教學課程中的練習需要以下 AWS CLI 版本 或更新版本：

```
aws-cli/1.16.63
```

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 AWS CLI config 檔案中為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[具名描述檔](#)。

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

如需可用 AWS 區域的清單，請參閱 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 帳戶和 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

[步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)

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

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

本節包含下列步驟：

- [建立兩個 Amazon Kinesis 資料串流](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查 Apache Flink 串流 Java 程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [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 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
```

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

- 使用下列內容建立名為 `stock.py` 的檔案：

```
import datetime  
import json  
import random  
import boto3  
  
STREAM_NAME = "ExampleInputStream"  
  
def get_data():  
    return {  
        "EVENT_TIME": datetime.datetime.now().isoformat(),  
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),  
        "PRICE": round(random.random() * 100, 2),  
    }  
  
def generate(stream_name, kinesis_client):  
    while True:  
        data = get_data()  
        print(data)
```

```
kinesis_client.put_record(  
    StreamName=stream_name, Data=json.dumps(data),  
    PartitionKey="partitionkey"  
)  
  
if __name__ == "__main__":  
    generate(STREAM_NAME, boto3.client("kinesis"))
```

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

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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[使用執行期屬性](#)。

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，您可以指定程式碼內容類型 (JAR 或 ZIP)。

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

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

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

上傳應用程式的程式碼

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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-*<username>* 儲存貯體，並選擇上傳。
8. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

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

Note

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立並執行應用程式 \(主控台\)](#)
- [建立並執行應用程式 \(AWS CLI\)](#)

建立並執行應用程式 (主控台)

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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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**)。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

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

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

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，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，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 來建立和執行 Managed Service for Apache Flink 應用程式。Managed Service for Apache Flink 會使用 `kinesisanalyticsv2` AWS CLI 命令來建立並與 Managed Service for Apache Flink 應用程式互動。

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。

JSON

```
{
  "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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

Note

若要存取其他 Amazon 服務，您可以使用適用於 Java 的 AWS SDK。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策略，[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策略)。
- d. 選擇 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的[建立 IAM 角色 \(主控台\)](#)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。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 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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 [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 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱 [the section called “在 Managed Service for Apache Flink 中設定應用程式記錄”](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

```
{
  "ApplicationName": "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來更新應用程式程式碼時，您可以使用 [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 儲存貯體中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：清除 AWS 資源](#)****步驟 4：清除 AWS 資源**

本節包含清除入門教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)

- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除 Rour IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Kinesis Data Streams 面板中，選擇 ExampleInputStream。
3. 在 ExampleInputStream 頁面，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除 Rour 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 政策。
5. 選擇政策動作，然後選擇刪除。

6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

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。

- 適用於 [Amazon Kinesis 的 AWS 串流資料解決方案](#)：適用於 Amazon Kinesis 的 AWS 串流資料解決方案會自動設定所需的 AWS 服務，以輕鬆擷取、儲存、處理和交付串流資料。該解決方案提供了多種解決串流資料使用案例的選項。Managed Service for Apache Flink 選項提供端對端串流 ETL 範例，展示真實世界的應用程式如何根據模擬的紐約計程車資料執行分析作業。解決方案會設定所有必要 AWS 的資源，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](#)：適用於 Amazon MSK 的串流資料解決方案提供 AWS CloudFormation 範本，其中資料流經生產者、串流儲存體、消費者和目的地。AWS
- [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。
- [Amazon Managed Service for Apache Flink 研討會](#)：在本研討會中，您將建置 end-to-end 串流架構，以近乎即時的方式擷取、分析和視覺化串流資料。您著手改善紐約市一家計程車公司的運營。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 Flink Managed Service for Apache Flink 中使用 Flink 1.15.2。

- [Apache Flink 程式碼範例](#)：一個 GitHub 儲存庫，其中有各種 Apache Flink 應用程式範例。

入門：Flink 1.8.2 - 棄用

Note

Apache Flink 社群已超過三年不支援 Apache Flink 1.6、1.8 和 1.11 版。我們計劃於 2024 年 11 月 5 日在 Amazon Managed Service for Apache Flink 中棄用這些版本。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。此時您可以繼續執行現有的應用程式。您可以使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，以狀態升級應用程式。如需詳細資訊，請參閱 [針對 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 Managed Service for Apache Flink 中的 DataStream API](#)教學課程版本。

主題

- [Managed Service for Apache Flink 應用程式的元件](#)
- [完成練習的先決條件](#)
- [步驟 1：設定 AWS 帳戶並建立管理員使用者](#)
- [步驟 2：設定 AWS Command Line Interface \(AWS CLI\)](#)
- [步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)
- [步驟 4：清除 AWS 資源](#)

Managed Service for Apache Flink 應用程式的元件

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

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[新增串流資料來源](#)。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱[運算子](#)。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[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[將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用](#)。
- 我們建議您使用開發環境 (如 [Eclipse Java Neon](#) 或 [IntelliJ Idea](#)) 來開發和編譯您的應用程式。
- [Git 用戶端](#)。如果您尚未安裝 Git 用戶端，請先安裝。
- [Apache Maven 編譯器外掛程式](#)。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 [步驟 1：設定 AWS 帳戶並建立管理員使用者](#)。

步驟 1：設定 AWS 帳戶並建立管理員使用者

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。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 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者[AWS 管理主控台](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄做為身分來源的教學課程，請參閱 AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

- 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

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

- 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 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 - AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。

哪個使用者需要程式設計存取權？	到	根據
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 設定 AWS CLI 要使用 AWS IAM Identity Center 的。 - AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的 IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中 將臨時登入資料與 AWS 資源搭配使用 的指示。
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 使用 IAM 使用者憑證進行身分驗證。 - AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的 使用長期憑證進行身分驗證。 - 對於 AWS APIs，請參閱《IAM 使用者指南》中的 管理 IAM 使用者的存取金鑰。

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

在此步驟中，您會下載並設定 AWS CLI 以搭配 Managed Service for Apache Flink 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 AWS CLI 安裝，您可能需要升級以取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[安裝 AWS Command Line Interface](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本教學課程中的練習需要以下 AWS CLI 版本 或更新版本：

```
aws-cli/1.16.63
```

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 AWS CLI config 檔案中為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[具名描述檔](#)。

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

如需可用區域的清單，請參閱 Amazon Web Services 一般參考 中的 [區域與端點](#)。

 Note

本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用不同的 AWS 區域，請將本教學課程程式碼和命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 帳戶和 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

[步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)

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

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

本節包含下列步驟：

- [建立兩個 Amazon Kinesis 資料串流](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查 Apache Flink 串流 Java 程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [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 命令。

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

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

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

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

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

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

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

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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[使用執行期屬性](#)。

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 “將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用”](#)。

編譯應用程式的程式碼

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

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

```
mvn package -Dflink.version=1.8.2
```

- 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

Note

提供的來源程式碼依賴於 Java 1.8 中的程式庫。確保您專案的 Java 版本是 1.8。

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

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

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

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

上傳應用程式的程式碼

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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-**<username>** 儲存貯體，並選擇上傳。
8. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

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

Note

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(AWS CLI\)](#)

建立並執行應用程式（主控台）

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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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**)。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入以下應用程式屬性和數值：

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

5. 在監控下，確保監控指標層級設為應用程式。
6. 針對 CloudWatch 記錄，選取啟用核取方塊。
7. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

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

建立並執行應用程式 (AWS CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

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

JSON

```
{
  "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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

Note

若要存取其他 Amazon 服務，您可以使用適用於 Java 的 AWS SDK。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策略，[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策略)。
- d. 選擇 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的[建立 IAM 角色 \(主控台\)](#)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。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 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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 [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 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱 [the section called “在 Managed Service for Apache Flink 中設定應用程式記錄”](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

```
{
  "ApplicationName": "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來更新應用程式程式碼時，您可以使用 [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：清除 AWS 資源](#)****步驟 4：清除 AWS 資源**

本節包含清除入門教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)

- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Kinesis Data Streams 面板中，選擇 ExampleInputStream。
3. 在 ExampleInputStream 頁面，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。

5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

入門：Flink 1.6.2 - 棄用

Note

Apache Flink 社群超過三年不支援 Apache Flink 1.6、1.8 和 1.11 版。我們計劃於 2024 年 11 月 5 日在 Amazon Managed Service for Apache Flink 中棄用這些版本。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。此時您可以繼續執行現有的應用程式。您可以使用 Amazon Managed Service for Apache Flink 中的就地版本升級功能，以狀態升級應用程式。如需詳細資訊，請參閱 [針對 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 Managed Service for Apache Flink 中的 DataStream API](#)教學課程版本。

主題

- [Managed Service for Apache Flink 應用程式的元件](#)
- [完成練習的先決條件](#)
- [步驟 1：設定 AWS 帳戶並建立管理員使用者](#)
- [步驟 2：設定 AWS Command Line Interface \(AWS CLI\)](#)
- [步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)
- [步驟 4：清除 AWS 資源](#)

Managed Service for Apache Flink 應用程式的元件

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

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[新增串流資料來源](#)。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱[運算子](#)。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[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 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。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 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者[AWS 管理主控台](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄做為身分來源的教學課程，請參閱 AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

- 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

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

- 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 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 - AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。

哪個使用者需要程式設計存取權？	到	根據
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 設定 AWS CLI 要使用 AWS IAM Identity Center 的。 • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的 IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中 將臨時登入資料與 AWS 資源搭配使用 的指示。
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 使用 IAM 使用者憑證進行身分驗證。 • AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的 使用長期憑證進行身分驗證。 • 對於 AWS APIs，請參閱《IAM 使用者指南》中的 管理 IAM 使用者的存取金鑰。

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

在此步驟中，您會下載並設定 AWS CLI 以搭配 Managed Service for Apache Flink 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 AWS CLI 安裝，您可能需要升級以取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[安裝 AWS Command Line Interface](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本教學課程中的練習需要以下 AWS CLI 版本 或更新版本：

```
aws-cli/1.16.63
```

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 AWS CLI config 檔案中為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[具名描述檔](#)。

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

如需可用 AWS 區域的清單，請參閱 [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 帳戶和 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

[步驟 3：建立並執行 Managed Service for Apache Flink 應用程式](#)

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

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

本節包含下列步驟：

- [建立兩個 Amazon Kinesis 資料串流](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查 Apache Flink 串流 Java 程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)

建立兩個 Amazon Kinesis 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 命令。

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

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

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
import datetime  
import json  
import random  
import boto3
```

```
STREAM_NAME = "ExampleInputStream"

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

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

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

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

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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[使用執行期屬性](#)。

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，您可以指定程式碼內容類型 (JAR 或 ZIP)。

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

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

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

上傳應用程式的程式碼

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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-*<username>* 儲存貯體，並選擇上傳。
8. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 在設定許可步驟中，保留原有設定。選擇下一步。

10. 在設定屬性步驟中，保留原有設定。選擇上傳。

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

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

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

 Note

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(AWS CLI\)](#)

建立並執行應用程式（主控台）

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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
 5. 選擇 建立應用程式。

Note

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

- 政策：**kinesis-analytics-service-MyApplication-us-west-2**
- 角色：**kinesisanalytics-MyApplication-us-west-2**

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 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**)。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **java-getting-started-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入以下應用程式屬性和數值：

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

5. 在監控下，確保監控指標層級設為應用程式。
6. 針對 CloudWatch 記錄，選取啟用核取方塊。
7. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

執行應用程式

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

建立並執行應用程式 (AWS CLI)

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

建立許可政策

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

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

JSON

```
{
  "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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

Note

若要存取其他 Amazon 服務，您可以使用適用於 Java 的 AWS SDK。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 **AKReadSourceStreamWriteSinkStream** 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的[建立 IAM 角色 \(主控台\)](#)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 [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 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱 [the section called “在 Managed Service for Apache Flink 中設定應用程式記錄”](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

```
{"ApplicationName": "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來更新應用程式程式碼時，您可以使用 [UpdateApplication](#) AWS CLI 動作。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體中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：清除 AWS 資源

本節包含清除入門教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Kinesis Data Streams 面板中，選擇 ExampleInputStream。

3. 在 ExampleInputStream 頁面，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

Managed Service for Apache Flink 的舊版（舊版）範例

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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逐步指示，可協助您建立 Managed Service for Apache Flink 應用程式並測試結果。

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 Managed Service for Apache Flink 中的 DataStream API](#)

Note

這些範例假設您使用美國西部 (奧勒岡) 區域 (us-west-2)。如果您使其他區域，請相應地更新應用程式的程式碼、命令和 IAM 角色。

主題

- [DataStream API 範例](#)
- [Python 範例](#)
- [Scala 範例](#)

DataStream API 範例

下列範例示範如何使用 Apache Flink DataStream API 建立應用程式。

主題

- [範例：輪轉時段](#)
- [範例：滑動視窗](#)
- [範例：寫入 Amazon S3 儲存貯體](#)
- [教學課程：使用 Managed Service for Apache Flink 應用程式，將資料從 MSK 叢集中的一個主題複寫到 VPC 中的另一個主題](#)
- [範例：搭配 Kinesis 資料串流使用 EFO 取用者](#)
- [範例：寫入 Firehose](#)
- [範例：從不同帳戶中的 Kinesis 串流讀取](#)
- [教學課程：搭配 Amazon MSK 使用自訂信任存放區](#)

範例：輪轉時段

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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 Managed Service for Apache Flink 中的 DataStream API](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

 Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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
```

```
Flink 1.13 onward

        .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //
        .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 Managed Service for Apache Flink 中的 DataStream API](#)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 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-**<username>** 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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**)。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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. 針對 CloudWatch 記錄，選取啟用核取方塊。
6. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`

- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。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，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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：滑動視窗

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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 Managed Service for Apache Flink 中的 DataStream API 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

 Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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 Managed Service for Apache Flink 中的 DataStream API](#)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 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-**<username>** 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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`)。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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. 針對 CloudWatch 記錄，選取啟用核取方塊。
6. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：**/aws/kinesis-analytics/MyApplication**
- 日誌串流：**kinesis-analytics-log-stream**

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，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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在滑動視窗教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

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 Managed Service for Apache Flink 中的 DataStream API](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [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 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [驗證應用程式輸出](#)
- [選用：自訂來源和接收器](#)
- [清除 AWS 資源](#)

建立相依資源

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 資源尚不存在，則應用程式會建立這些資源：

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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 StreamingFileSink 的詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [StreamingFileSink](#)。

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 Managed Service for Apache Flink 中的 DataStream API](#)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 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-*<username>* 儲存貯體，導航到 code 資料夾，然後選擇上傳。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 aws-kinesis-analytics-java-apps-1.0.jar 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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. 針對 CloudWatch 記錄，選取啟用核取方塊。
6. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。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，選擇執行。保持選取不使用快照執行選項，然後確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。

驗證應用程式輸出

在 Amazon S3 主控台中開啟 S3 儲存貯體中的 data 資料夾。

幾分鐘後，將顯示包含來自應用程式之彙總資料的物件。

Note

彙總在 Flink 中預設為啟用。可使用下列命令將其停用：

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

選用：自訂來源和接收器

在本節中，您將自訂來源和接收器物件的設定。

Note

在變更以下各節中描述的程式碼區段之後，請執行下列動作以重新載入應用程式的程式碼：

- 重複[the section called “編譯應用程式程式碼”](#)一節中的步驟，以編譯更新的應用程式程式碼。

- 重複[the section called “上傳 Apache Flink 串流 Java 程式碼”](#)一節中的步驟，以上傳更新的應用程式程式碼。
- 在主控台的應用程式頁面，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儲存貯體指派者，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。[UpdateApplication](#) 動作的下列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](#) 動作擷取目前的應用程式版本。

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](#)。

清除 AWS 資源

本節包含清除您在 Amazon S3 教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

教學課程：使用 Managed Service for Apache Flink 應用程式，將資料從 MSK 叢集中的一個主題複寫到 VPC 中的另一個主題

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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 Managed Service for Apache Flink 中的 DataStream API 練習](#)。

本教學課程包含下列章節：

- [建立 Amazon VPC 和 Amazon MSK 叢集](#)
- [建立應用程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)

- [建立應用程式](#)
- [設定應用程式](#)
- [執行應用程式](#)
- [測試應用程式。](#)

建立 Amazon VPC 和 Amazon MSK 叢集

若要建立範例 VPC 和 Amazon MSK 叢集以從 Managed Service for Apache Flink 應用程式存取，請按照 [Amazon MSK 使用入門](#) 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。您可以使用以下命令取得啟動伺服器的清單 (使用 *ClusterArn* 取代 MSK 叢集的 ARN)：

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

- 遵循教學課程中的步驟時，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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。如需詳細資訊，請參閱[安裝 Git](#)。
2. 使用以下指令複製遠端儲存庫：

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

3. 應用程式的程式碼位於 `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java` 檔案中。您可以檢查程式碼，以熟悉 Managed Service for Apache Flink 應用程式程式碼的結構。
4. 使用命令列 Maven 工具或偏好的開發環境來建立 JAR 檔案。若要使用命令列 Maven 工具編譯 JAR 檔案，請輸入下列命令：

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

如果建置成功，會建立下列檔案：

```
target/KafkaGettingStartedJob-1.0.jar
```

Note

提供的來源程式碼依賴於 Java 11 中的程式庫。如果您使用的是開發環境，

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#)一節建立的 Amazon S3 儲存貯體。

Note

如果您從入門教學課程中刪除了 Amazon S3 儲存貯體，請再次執行下列 [the section called “上傳應用程式碼 JAR 檔案”](#) 步驟。

1. 在 Amazon S3 主控台中，選擇 `ka-app-code-<username>` 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 `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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink>：// 開啟 Amazon MSF 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 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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

當您使用主控台 (例如 CloudWatch Logs 或 Amazon VPC) 指定應用程式資源時，主控台會修改您的應用程式執行角色，以授與存取這些資源的許可。

4. 在屬性下，選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSource	主題	AWS KafkaTutorialTopic
KafkaSource	bootstrap.servers	#####
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

Note

預設憑證的 ssl.truststore.password 是「changeit」；如果您使用預設憑證，不需要變更此值。

- 再次選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSink	主題	AWS KafkaTutorialTopic Destination
KafkaSink	bootstrap.servers	#####
KafkaSink	security.protocol	SSL

群組 ID	金鑰	值
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

應用程式的程式碼會讀取上述應用程式屬性，以設定用來與 VPC 和 Amazon MSK 叢集互動的來源和接收器。如需如何使用屬性的詳細資訊，請參閱[使用執行期屬性](#)。

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. 針對 CloudWatch 記錄，選取啟用核取方塊。
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，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。

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 [Managed Service for Apache Flink 故障診斷](#) 主題中的 [無法存取 VPC 中的資源](#) 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 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，使用 [增強型廣發 \(EFO\)](#) 取用者從 Kinesis 資料串流讀取。如果 Kinesis 取用者使用 EFO，Kinesis Data Streams 服務會提供專屬頻寬，而不是讓取用者與其他從串流讀取的取用者共用串流的固定頻寬。

如需將 EFO 用於 Kinesis 取用者的詳細資訊，請參閱 [FLIP-128：將增強型扇出用於 Kinesis 取用者](#)。

您在此範例中建立的應用程式使用 AWS Kinesis 連接器 (flink-connector-kinesis) 1.15.3。

 Note

若要設定此練習的必要先決條件，請先完成 [教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)

- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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 Managed Service for Apache Flink 中的 DataStream API](#)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 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 `ka-app-code-<username>` 儲存貯體，並選擇上傳。

2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 `https://console.aws.amazon.com/flink` 開啟 Amazon MSF 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 IAM 政策

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。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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	EFO

群組 ID	金鑰	值
ConsumerConfigProperties	flink.stream.efo.consumername	basic-efo-flink-app
ConsumerConfigProperties	INPUT_STREAM	ExampleInputStream
ConsumerConfigProperties	flink.inputstream.initpos	LATEST
ConsumerConfigProperties	AWS_REGION	us-west-2

6. 在屬性下，選擇建立群組。
7. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	OUTPUT_STREAM	ExampleOutputStream
ProducerConfigProperties	AWS_REGION	us-west-2
ProducerConfigProperties	AggregationEnabled	false

8. 在監控下，確保監控指標層級設為應用程式。
9. 針對 CloudWatch 記錄，選取啟用核取方塊。
10. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication

- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。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 主控台上查看 Managed Service for Apache Flink 指標，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扇出標籤中查看 Kinesis Data Streams 主控台，以取得取用者名稱 (basic-efo-flink-app)。

清除 AWS 資源

本節包含清除 efo Window 教學課程中建立 AWS 的資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 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 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：寫入 Firehose

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，其具有 Kinesis 資料串流做為來源，以及 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 Managed Service for Apache Flink 中的 DataStream API 練習](#)。

本節包含下列步驟：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查 Apache Flink 串流 Java 程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 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 () 的 Firehose 串流 ExampleDeliveryStream。
- 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。
- 《[Amazon Data Firehose 開發人員指南](#)》中的 [建立 Amazon Kinesis Data Firehose 交付串流](#)。為您的 Firehose 串流命名 **ExampleDeliveryStream**。當您建立 Firehose 串流時，也請建立 Firehose 串流的 S3 目的地和 IAM 角色。

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

 Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

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

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

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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));
```

- 應用程式使用 Firehose 接收器將資料寫入 Firehose 串流。下列程式碼片段會建立 Firehose 接收器：

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {  
    Properties sinkProperties = new Properties();  
    sinkProperties.setProperty(AWS_REGION, region);  
  
    return KinesisFirehoseSink.<String>builder()  
        .setFirehoseClientProperties(sinkProperties)  
        .setSerializationSchema(new SimpleStringSchema())  
        .setDeliveryStreamName(outputDeliveryStreamName)  
        .build();  
}
```

編譯應用程式程式碼

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 Managed Service for Apache Flink 中的 DataStream API](#)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 “將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用”](#)。
3. 使用下列命令編譯應用程式：

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

Note

提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (target/aws-kinesis-analytics-java-apps-1.0.jar)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

上傳應用程式的程式碼

1. 前往 <https://console.aws.amazon.com/s3/> 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在主控台中，選擇 ka-app-code-**<username>** 儲存貯體，並選擇上傳。
3. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 java-getting-started-1.0.jar 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

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

Note

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(AWS CLI\)](#)

建立並執行應用程式（主控台）

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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 IAM 政策

編輯 IAM 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`) 的所有執行個體。

JSON

```
{
  "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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **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. 針對 CloudWatch 記錄，選取啟用核取方塊。
6. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：**/aws/kinesis-analytics/MyApplication**
- 日誌串流：**kinesis-analytics-log-stream**

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，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，選擇設定。更新應用程式設定，然後選擇更新。

Note

若要在主控台上更新應用程式的程式碼，您必須變更 JAR 的物件名稱、使用不同的 S3 儲存貯體，或使用[the section called “更新應用程式的程式碼”](#)章節中所述的 AWS CLI。如果檔案名稱或儲存貯體未變更，當您在設定頁面上選擇更新時，不會重新載入應用程式的程式碼。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 來建立和執行 Managed Service for Apache Flink 應用程式。

建立許可政策

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 read 動作的許可，而另一條則是授與目的地串流上 write 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。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。

JSON

```
{
  "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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

Note

若要存取其他 Amazon 服務，您可以使用適用於 Java 的 AWS SDK。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 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

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。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的[建立 IAM 角色 \(主控台\)](#)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 create_request.json 的檔案。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 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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 [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 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱 [the section called “在 Managed Service for Apache Flink 中設定應用程式記錄”](#)。

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來更新應用程式程式碼時，您可以使用 [UpdateApplication](#) AWS CLI 動作。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體中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

本節包含清除入門教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Firehose 串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，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 物件和儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：<https://console.aws.amazon.com/s3/>。
2. 選擇 ka-app-code-**<username>** 儲存貯體。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。
4. 如果您已為 Firehose 串流的目的地建立 Amazon S3 儲存貯體，請一併刪除該儲存貯體。

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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 如果您已為 Firehose 串流建立新的政策，請一併刪除該政策。
7. 在導覽列中，選擇角色。
8. 選擇 kinesis-analytics-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 串流建立新的角色，也請刪除該角色。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：從不同帳戶中的 Kinesis 串流讀取

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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：

- [先決條件](#)
- [設定](#)
- [建立來源 Kinesis 串流](#)
- [建立和更新 IAM 角色和政策](#)
- [更新 Python 指令碼](#)
- [更新 Java 應用程式](#)
- [建置、上傳和執行應用程式](#)

先決條件

- 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 Managed Service for Apache Flink 中的 DataStream API](#)教學課程後再繼續。
- 您需要兩個 AWS 帳戶才能完成此教學課程：一個用於來源串流，另一個用於應用程式和接收器串流。使用您用於應用程式和接收器串流入門教學 AWS 課程的帳戶。對來源串流使用不同的 AWS 帳戶。

設定

您將使用具名設定檔來存取您的兩個 AWS 帳戶。修改您的 AWS 登入資料和組態檔案，以包含兩個設定檔，其中包含兩個帳戶的 區域和連線資訊。

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`。其中包含了如何使用設定檔存取兩個不同 AWS 帳戶的範例。

如需搭配 使用具名設定檔的詳細資訊 AWS CLI，請參閱 AWS Command Line Interface 文件中的 [具名設定檔](#)。

建立來源 Kinesis 串流

在本節中，您將在來源帳戶中建立 Kinesis 串流。

輸入下列命令建立 Kinesis 串流，供應用程式用來輸入。請注意，`--profile` 參數會指定要使用的帳戶設定檔。

```
$ aws kinesis create-stream \  
--stream-name SourceAccountExampleInputStream \  
--shard-count 1 \  
--profile ka-source-stream-account-profile
```

建立和更新 IAM 角色和政策

若要允許跨 AWS 帳戶存取物件，您可以在來源帳戶中建立 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](#)

接收器帳戶角色和政策

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`)。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AssumeRoleInSourceAccount",  
      "Effect": "Allow",  
      "Action": "sts:AssumeRole",
```

```

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-2:123456789012:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [

```

```

        "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
}
]
}

```

2. 開啟 kinesis-analytics-MyApplication-us-west-2 角色，記下其 Amazon Resource Name (ARN)。您會在下一節中用到它。角色 ARN 如下所示：

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-west-2
```

來源帳戶角色和政策

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。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetRecords",
        "kinesis:GetShardIterator",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:us-west-2:111122223333: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*)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/service-role/kinesis-analytics-MyApplication-us-west-2"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

更新 Python 指令碼

在本節中，您將更新產生範例資料的 Python 指令碼，以使用來源帳戶設定檔。

使用下列反白顯示的變更更新 stock.py 指令碼。

```

import json
import boto3
import random
import datetime
import os

os.environ['AWS_PROFILE'] = 'ka-source-stream-account-profile'
os.environ['AWS_DEFAULT_REGION'] = 'us-west-2'

kinesis = boto3.client('kinesis')
def getReferrer():
    data = {}
    now = datetime.datetime.now()
    str_now = now.isoformat()
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data

while True:
    data = json.dumps(getReferrer())
    print(data)
    kinesis.put_record(
        StreamName="SourceAccountExampleInputStream",
        Data=data,
        PartitionKey="partitionkey")

```

更新 Java 應用程式

在本節中，您將更新 Java 應用程式的程式碼，以便在從來源串流讀取時擔任來源帳戶角色。

對 `BasicStreamingJob.java` 檔案進行下列變更。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、上傳和執行應用程式

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 使用自訂信任存放區

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

目前資料來源 API

如果您使用的是目前的資料來源 APIs，您的應用程式可以利用[此處](#)所述的 Amazon MSK Config Providers 公用程式。這可讓您的 `KafkaSource` 函數存取 Amazon S3 中雙向 TLS 的金鑰存放區和信任存放區。

```
...
// 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 open 方法以載入自訂信任存放區。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 Kafka 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 Managed Service for Apache Flink 中的 DataStream API 練習](#)。

以下教學課程示範如何安全地連線 (傳輸中加密) 到使用由自訂、私有甚至自託管憑證授權機構 (CA) 發行的伺服器憑證之 Kafka 叢集。

若要透過 TLS 將任何 Kafka 用戶端安全地連線至 Kafka 叢集，Kafka 用戶端（例如範例 Flink 應用程式）必須信任 Kafka 叢集伺服器憑證（從發行 CA 到根層級 CA）所呈現的完整信任鏈。作為自訂信任存放區的範例，我們將使用啟用相互 TLS (MTLS) 身分驗證的 Amazon MSK 叢集。這表示 MSK 叢集節點使用由 AWS Certificate Manager Private Certificate Authority (ACM Private CA) 發行的伺服器憑證，該憑證是您帳戶和區域的私有憑證，因此不受執行 Flink 應用程式之 Java 虛擬機器 (JVM) 的預設信任存放區所信任。

Note

- 金鑰存放區用於存放私有金鑰，以及應用程式應同時呈現給伺服器或用戶端進行驗證的身分憑證。
- 信任存放區用於存放來自認證授權機構 (CA) 的憑證，以驗證伺服器在 SSL 連線中呈現的憑證。

您也可以使用本教學課程中的技術，在 Managed Service for Apache Flink 應用程式與其他 Apache Kafka 來源之間進行互動，例如：

- 在 AWS ([Amazon EC2](#) 或 [Amazon EKS](#)) 中託管的自訂 Apache Kafka 叢集
- 在 中託管的 [Confluent Kafka](#) 叢集 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](#)
- [建立自訂信任存放區並將其套用至您的叢集](#)
- [建立應用程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立應用程式](#)
- [設定應用程式](#)
- [執行應用程式](#)
- [測試應用程式。](#)

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。您可以使用以下命令取得啟動伺服器的清單 (使用 `ClusterArn` 取代 MSK 叢集的 ARN)：

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

- 遵循本教學課程中的步驟和先決條件教學課程時，請務必在程式碼、命令和主控台項目中使用您選取的 AWS 區域。

建立自訂信任存放區並將其套用至您的叢集

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。如需詳細資訊，請參閱[安裝 Git](#)。
2. 使用以下指令複製遠端儲存庫：

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

3. 應用程式的程式碼位於 amazon-kinesis-data-analytics-java-examples/CustomKeystore 檔案中。您可以檢查程式碼，以熟悉 Managed Service for Apache Flink 程式碼的結構。
4. 使用命令列 Maven 工具或偏好的開發環境來建立 JAR 檔案。若要使用命令列 Maven 工具編譯 JAR 檔案，請輸入下列命令：

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

如果建置成功，會建立下列檔案：

```
target/flink-app-1.0-SNAPSHOT.jar
```

Note

提供的來源程式碼依賴於 Java 11 中的程式庫。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[教學課程：開始使用 Managed Service for Apache Flink 中的 DataStream API](#)一節建立的 Amazon S3 儲存貯體。

Note

如果您從入門教學課程中刪除了 Amazon S3 儲存貯體，請再次執行下列 [the section called “上傳應用程式碼 JAR 檔案”](#) 步驟。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-*<username>* 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。

- 對於 Amazon S3 物件的路徑，請輸入 **flink-app-1.0-SNAPSHOT.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。

 Note

當您使用主控台 (例如日誌或 VPC) 指定應用程式資源時，主控台會修改您的應用程式執行角色，以授與存取這些資源的許可。

4. 在屬性下，選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSource	主題	AWS KafkaTutorialTopic
KafkaSource	bootstrap.servers	#####
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

 Note

預設憑證的 ssl.truststore.password 是「changeit」；如果您使用預設憑證，不需要變更此值。

再次選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSink	主題	AWS KafkaTutorialTopic Destination

群組 ID	金鑰	值
KafkaSink	bootstrap.servers	#####
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

應用程式的程式碼會讀取上述應用程式屬性，以設定用來與 VPC 和 Amazon MSK 叢集互動的來源和接收器。如需如何使用屬性的詳細資訊，請參閱[使用執行期屬性](#)。

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. 針對 CloudWatch 記錄，選取啟用核取方塊。
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，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。

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 [Managed Service for Apache Flink 故障診斷](#) 主題中的 [無法存取 VPC 中的資源](#) 一節。

Python 範例

下列範例示範如何搭配使用 Python 與 Apache Flink 資料表 API 建立應用程式。

主題

- [範例：在 Python 中建立輪轉視窗](#)
- [範例：在 Python 中建立滑動視窗](#)
- [範例：在 Python 中將串流資料傳送至 Amazon S3](#)

範例：在 Python 中建立輪轉視窗

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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：開始使用 Managed Service for Apache Flink 中的 Python 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳 Apache Flink 串流 Python 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

Note

本節中的 Python 指令碼會使用 AWS CLI。您必須 AWS CLI 將設定為使用您的帳戶登入資料和預設區域。若要設定您的 AWS CLI，請輸入下列項目：

```
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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 使用偏好的壓縮應用程式來壓縮 tumbling-windows.py 和 flask-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-**<username>** 儲存貯體，並選擇上傳。
3. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 myapp.zip 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。

3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱[指定您的程式碼檔案](#)。
9. 輸入下列資料：

群組 ID	金鑰	值
kinesis.analytics.flink.run.options	python	tumbling-windows.py

群組 ID	金鑰	值
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis-1.15.2.jar

10. 在監控下，確保監控指標層級設為應用程式。
11. 針對 CloudWatch 記錄，選取啟用核取方塊。
12. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 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**)。

JSON

```
{
  "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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Python 中建立滑動視窗

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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 Managed Service for Apache Flink 中的 Python 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳 Apache Flink 串流 Python 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

Note

本節中的 Python 指令碼會使用 AWS CLI。您必須 AWS CLI 將設定為使用您的帳戶登入資料和預設區域。若要設定您的 AWS CLI，請輸入下列項目：

```
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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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")
)

```

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

壓縮並上傳 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

本節說明如何封裝 Python 應用程式。

1. 使用偏好的壓縮應用程式來壓縮 sliding-windows.py 和 flask-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-**<username>** 儲存貯體，並選擇上傳。
3. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 myapp.zip 檔案。

4. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 [指定您的程式碼檔案](#)。
9. 輸入以下應用程式屬性和數值：

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

10. 在監控下，確保監控指標層級設為應用程式。
11. 針對 CloudWatch 記錄，選取啟用核取方塊。
12. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 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**)。

JSON

```
{
  "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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在滑動視窗教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。

2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Python 中將串流資料傳送至 Amazon S3

 Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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：開始使用 Managed Service for Apache Flink 中的 Python 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳 Apache Flink 串流 Python 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- Kinesis 資料串流 (ExampleInputStream)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼和輸出 (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。

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

 Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

 Note

本節中的 Python 指令碼會使用 AWS CLI。您必須 AWS CLI 將設定為使用您的帳戶登入資料和預設區域。若要設定您的 AWS CLI，請輸入下列項目：

```
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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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)
```

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

壓縮並上傳 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 使用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-**<username>** 儲存貯體，並選擇上傳。
3. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 myapp.zip 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 `ka-app-code-<username>`。
 - 對於 Amazon S3 物件的路徑，請輸入 `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>

群組 ID	金鑰	值
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

- 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 [指定您的程式碼檔案](#)。
- 輸入以下應用程式屬性和數值：

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

- 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 [指定您的程式碼檔案](#)。
- 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
sink.config.0	output.bucket.name	<i>bucket-name</i>

- 在監控下，確保監控指標層級設為應用程式。
- 針對 CloudWatch 記錄，選取啟用核取方塊。
- 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 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**)。

JSON

```
{
  "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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在滑動視窗教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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` 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 `kinesis-analytics-MyApplication-us-west-2` 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 `/aws/kinesis-analytics/MyApplication` 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

Scala 範例

下列範例示範如何搭配使用 Scala 與 Apache Flink 來建立應用程式。

主題

- [範例：在 Scala 中建立輪轉視窗](#)
- [範例：在 Scala 中建立滑動視窗](#)
- [範例：在 Scala 中將串流資料傳送至 Amazon S3](#)

範例：在 Scala 中建立輪轉視窗

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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，您將建立一個使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的簡單串流應用程式。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](#)
- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清除 AWS 資源](#)

下載並檢查應用程式程式碼

此範例的 Python 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，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與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行期應用程式的屬性，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. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。

2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 `ka-app-code-
<username>`。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇建立儲存貯體。
7. 選擇 `ka-app-code-
<username>` 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `tumbling-window-
scala-1.0.jar` 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

建立並執行應用程式（主控台）

依照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. 登入 AWS 管理主控台，並在 `https://console.aws.amazon.com/flink` 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 `ka-app-code-<username>`。
 - 對於 Amazon S3 物件的路徑，請輸入 `tumbling-window-scala-1.0.jar`。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 `kinesis-analytics-MyApplication-us-west-2`。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
ConsumerConfigProperties	<code>input.stream.name</code>	<code>ExampleInputStream</code>
ConsumerConfigProperties	<code>aws.region</code>	<code>us-west-2</code>
ConsumerConfigProperties	<code>flink.stream.initializers</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
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

8. 在監控下，確保監控指標層級設為應用程式。
9. 針對 CloudWatch 記錄，選取啟用核取方塊。
10. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 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**)。

JSON

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

```

```

        "Action": [
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

執行應用程式

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

停止應用程式

若要停止應用程式，請在 MyApplication 頁面上選擇停止。確認動作。

建立並執行應用程式 (CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 read 動作的許可，而另一條則是授與目的地串流上 write 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

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。

如需建立角色的逐步指示，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"
    }
]
}

```

使用下列請求執行 [CreateApplication](#) 以建立應用程式：

```
aws kinesisanalyticv2 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": "tumbling_window",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

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

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用 [StopApplication](#) 動作來停止應用程式。

停止應用程式

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

```
{
  "ApplicationName": "tumbling_window"
}
```

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

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

現在已停止應用程式。

新增 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱[設定應用程式記錄](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

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

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

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 [UpdateApplication](#) CLI 動作。

Note

若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需如何使用 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"
        }
      }
    }
  }
}
```

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)

- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。

7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Scala 中建立滑動視窗

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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，您將建立一個使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的簡單串流應用程式。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](#)

- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清除 AWS 資源](#)

下載並檢查應用程式程式碼

此範例的 Python 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，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與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行期應用程式的屬性，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](#)。

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

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

建立並執行應用程式（主控台）

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **sliding-window-scala-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

選擇儲存。

- 在屬性下，再次選擇新增群組。
- 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

- 在監控下，確保監控指標層級設為應用程式。
- 針對 CloudWatch 記錄，選取啟用核取方塊。
- 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 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**)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/sliding-window-  
scala-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
```

```

        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogStreams"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        ]
    },
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": [
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

執行應用程式

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

停止應用程式

若要停止應用程式，請在 MyApplication 頁面上選擇停止。確認動作。

建立並執行應用程式 (CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。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逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

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。

如需建立角色的逐步指示，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"
      }
    ]
  }
}

```

使用下列請求執行 [CreateApplication](#) 以建立應用程式：

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

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

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

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用 [StopApplication](#) 動作來停止應用程式。

停止應用程式

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

```
{
  "ApplicationName": "sliding_window"
}
```

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

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

現在已停止應用程式。

新增 CloudWatch 記錄選項

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 Logs 的相關資訊，請參閱[設定應用程式記錄](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

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

```
}  
}
```

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

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 [UpdateApplication](#) CLI 動作。

Note

若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"  
        }  
      }  
    }  
  }  
}
```

```
}  
}
```

清除 AWS 資源

本節包含清除在滑動視窗教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Scala 中將串流資料傳送至 Amazon S3

Note

如需目前範例，請參閱 [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)。

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，您將建立一個使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的簡單串流應用程式。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 儲存貯體 ka-app-code-<username> 中建立一個額外的資料夾 **data/**。

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](#)
- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清除 AWS 資源](#)

下載並檢查應用程式程式碼

此範例的 Python 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，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)
}
```

應用程式也使用 `StreamingFileSink` 來寫入 Amazon S3 儲存貯體：

```
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與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行期應用程式的屬性，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. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 ka-app-code-`<username>`。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇建立儲存貯體。
7. 選擇 ka-app-code-`<username>` 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 s3-sink-scala-1.0.jar 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

建立並執行應用程式（主控台）

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 `ka-app-code-<username>`。
 - 對於 Amazon S3 物件的路徑，請輸入 `s3-sink-scala-1.0.jar`。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 `kinesis-analytics-MyApplication-us-west-2`。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
ConsumerConfigProperties	<code>input.stream.name</code>	<code>ExampleInputStream</code>
ConsumerConfigProperties	<code>aws.region</code>	<code>us-west-2</code>
ConsumerConfigProperties	<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
ProducerConfigProperties	s3.sink.path	s3a://ka-app-code- <user-name> /data

8. 在監控下，確保監控指標層級設為應用程式。
9. 針對 CloudWatch 記錄，選取啟用核取方塊。
10. 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 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**)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:Abort*",
        "s3:DeleteObject*",
        "s3:GetObject*",
        "s3:GetBucket*",
        "s3:List*",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-<username>",
        "arn:aws:s3:::ka-app-code-<username>/*"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    }
  ]
}
```

```
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
      ]
    },
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    }
  ]
}
```

執行應用程式

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

停止應用程式

若要停止應用程式，請在 MyApplication 頁面上選擇停止。確認動作。

建立並執行應用程式 (CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 read 動作的許可，而另一條則是授與目的地串流上 write 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

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

JSON

```
{
  "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-2:123456789012:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
```

```

        "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ],
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": [
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-
west-2:123456789012:stream/ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-
west-2:123456789012:stream/ExampleOutputStream"
    }
]
}

```

如需建立許可政策的逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

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。

如需建立角色的逐步指示，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"  
  }  
]  
}
```

使用下列請求執行 [CreateApplication](#) 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": "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 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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用 [StopApplication](#) 動作來停止應用程式。

停止應用程式

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

```
{
  "ApplicationName": "s3_sink"
}
```

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

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

現在已停止應用程式。

新增 CloudWatch 記錄選項

您可以使用 AWS CLI 將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch Logs 的相關資訊，請參閱[設定應用程式記錄](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

```
{
  "ApplicationName": "s3_sink",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleInputStream",
            "flink.stream.initpos": "LATEST"
          }
        }
      ]
    }
  }
}
```

```

        "PropertyGroupId": "ProducerConfigProperties",
        "PropertyMap" : {
            "s3.sink.path" : "s3a://ka-app-code-<username>/data"
        }
    }
]
}
}
}

```

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

```

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

```

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 [UpdateApplication](#) CLI 動作。

Note

若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"
    }
}
}
```

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：<https://console.aws.amazon.com/s3/>。

2. 選擇 ka-app-code-**<username>** 儲存貯體。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

使用 Studio 筆記本搭配 Managed Service for Apache Flink

適用於 Managed Service for Apache Flink 的 Studio 筆記本可讓您即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。只要在 AWS 管理主控台中按幾下，您就可以啟動無伺服器筆記本來查詢資料串流，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 Managed Service for Apache Flink 中建立 Studio 筆記本](#)。如需 Apache Zeppelin 的詳細資訊，請參閱[Apache Zeppelin 文件](#)。

借助筆記本，您可以使用 SQL、Python 或 Scala 中的 Apache Flink [資料表 API 和 SQL](#) 或 Scala 中的 [DataStream API](#) 進行查詢建模。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 筆記本執行期版本](#)
- [建立 Studio 筆記本](#)
- [執行串流資料的互動式分析](#)
- [將 部署為具有持久狀態的應用程式](#)
- [檢閱 Studio 筆記本的 IAM 許可](#)
- [使用連接器和相依性](#)
- [實作使用者定義的函數](#)
- [啟用檢查點](#)
- [升級 Studio 執行期](#)

- [使用 AWS Glue](#)
- [Managed Service for Apache Flink 中 Studio 筆記本的範例和教學課程](#)
- [Managed Service for Apache Flink 的 Studio 筆記本疑難排解](#)
- [為 Managed Service for Apache Flink Studio 筆記本建立自訂 IAM 政策](#)

使用正確的 Studio 筆記本執行期版本

透過 Amazon Managed Service for Apache Flink Studio，您可以在互動式筆記本中即時查詢資料串流，並使用標準 SQL、Python 和 Scala 建置和執行串流處理應用程式。Studio 筆記本採用 [Apache Zeppelin](#) 技術，並使用 [Apache Flink](#) 做為串流處理引擎。

 Note

我們將於 2024 年 11 月 5 日以 Apache Flink 1.11 版取代 Studio 執行期。從這個日期開始，您將無法執行新的筆記本或使用此版本建立新的應用程式。我們建議您在該時間之前升級至最新的執行時間 (Apache Flink 1.15 和 Apache Zeppelin 0.10)。如需如何升級筆記本的指引，請參閱 [升級 Studio 執行期](#)。

Studio 執行期

Apache Flink 版本	Apache Zeppelin 版本	Python 版本	
1.15	0.1	3.8	建議
1.13	0.9	3.8	支援至 2024 年 10 月 16 日
1.11	0.9	3.7	已於 2025 年 2 月 24 日棄用

建立 Studio 筆記本

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

應用程式具有下列元件：

- 資料來源，例如 Amazon MSK 叢集、Kinesis 資料串流或 Amazon S3 儲存貯體。
 - AWS Glue 資料庫。此資料庫包含儲存資料來源、目標結構描述和端點的資料表。如需詳細資訊，請參閱 [使用 AWS Glue](#)。
 - 應用程式的程式碼。您的程式碼會實作您的分析查詢或程式。
 - 您的應用程式設定和執行期屬性。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 管理主控台 或 建立 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 MSK 主控台中，選擇您的叢集，然後選擇即時處理資料。
- 在 Kinesis Data Streams 主控台中，選擇您的資料串流，然後在應用程式標籤上選擇即時處理資料。
- 在 Managed Service for Apache Flink 主控台中，選擇 Studio 標籤，然後選擇建立 Studio 筆記本。

執行串流資料的互動式分析

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

下列範例 SQL 查詢顯示如何從資料來源擷取資料：

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

如需 Flink 串流 SQL 查詢的更多範例，請參閱 Apache Flink 文件中的[Managed Service for Apache Flink 中 Studio 筆記本的範例和教學課程](#)以下內容和[查詢](#)。

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

Flink 解譯器

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

名稱	類別	Description
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipynb	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

如需 Flink 解譯器的詳細資訊，請參閱[Apache Zeppelin 的 Flink 解譯器](#)。

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

如需更多的 PyFlink 具體範例，請參閱[使用適用於 Studio 和 Python 的 Managed Service for Apache Flink 以互動方式查詢資料串流](#)。

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>senv</code>	<code>StreamExecutionEnvironment</code>
<code>stenv</code>	<code>Blink ##### StreamTableEnvironment</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_env</code>	<code>StreamExecutionEnvironment</code>
<code>st_env</code>	<code>Blink ##### StreamTableEnvironment</code>

如需使用資料表環境的詳細資訊，請參閱 Apache Flink 文件中的[概念和常見 API](#)。

將 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。部署應用程式時，筆記中的所有段落都會依序執行 (從左至右、從上至下)，如同它們在筆記中顯示的一樣。您可以透過在筆記中選擇執行所有段落來檢查此順序。
- 你的程式碼是 Python 和 SQL 或 Scala 和 SQL 的組合。對於部署即應用程式，目前不支援 Python 和 Scala 的組合。
- 您的筆記必須只包含下列解譯器：`%flink`、`%flink.sql`、`%flink.pyflink`、`%flink.ipynb`、`%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.sql` 解譯器的筆記中的 `%flink.ipynb`)，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。
- 來源程式碼 (Flink SQL、PyFlink 或 Flink Scala) 中必須至少有一個段落處於啟用且可執行狀態，才能成功部署應用程式。
- 在某個段落內的解譯器指令中設定平行處理 (例如 `%flink.sql(parallelism=32)`) 將在從筆記部署的應用程式中略過。反之，您可以透過 AWS 管理主控台、AWS Command Line Interface 或 AWS API 更新已部署的應用程式，以根據應用程式所需的平行處理層級變更平行處理和/或 `ParallelismPerKPU` 設定，或者您可以為已部署的應用程式啟用自動擴展。
- 如果要部署為具有持久狀態的應用程式，則您的 VPC 必須具有網際網路存取。如果您的 VPC 無法存取網際網路，請參閱[在無網際網路存取的 VPC 中，部署為具有持久狀態的應用程式](#)。

Scala/Python 條件

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

SQL 條件

- 不允許使用簡單的 SELECT 陳述式，因為沒有相當於段落的輸出部分可以傳遞資料。
- 在任何指定段落中，DDL 陳述式 (USE、CREATE、ALTER、DROP、SET、RESET) 都必須放在 DML (INSERT) 陳述式前面。這是因為，段落中的 DML 陳述式必須作為單一 Flink 作業一起提交。
- 最多只能有一個段落中包含 DML 陳述式。這是因為，對於「部署即應用程式」功能，我們僅支援提交單一作業至 Flink。

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

檢閱 Studio 筆記本的 IAM 許可

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

服務	存取
CloudWatch Logs	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。如需連接器的詳細資訊，請參閱《Apache Flink 文件》中的[資料表和 SQL 連接器](#)。

預設連接器

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

如果您使用 [CreateApplication](#) API 來建立 Studio 筆記本，預設不包含 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 MSK 搭配使用的連接器，其中包含可透過 IAM 自動驗證的功能。

Note

下列範例中顯示的連接器版本是我們支援的唯一版本。

For the Kinesis connector:

```
"CustomArtifactsConfiguration": [{  
  "ArtifactType": "DEPENDENCY_JAR",  
  "MavenReference": {  
    "GroupId": "org.apache.flink",  
  
    "ArtifactId": "flink-sql-connector-kinesis",  
    "Version": "1.15.4"  
  }  
}]
```

For authenticating with AWS MSK through AWS IAM:

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

```
}]
```

For the Apache Kafka connector:

```
"CustomArtifactsConfiguration": [{  
  "ArtifactType": "DEPENDENCY_JAR",  
    "MavenReference": {  
    "GroupId": "org.apache.flink",  
  
      "ArtifactId": "flink-connector-kafka",  
      "Version": "1.15.4"  
    }  
}]
```

若要將這些連接器新增至現有的筆記本，請使用 [UpdateApplication](#) API 作業，並將它們指定為 CustomArtifactsConfigurationUpdate 資料類型的 MavenReference。

Note

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

新增相依性和自訂連接器

若要使用 AWS 管理主控台 將相依性或自訂連接器新增至 Studio 筆記本，請遵循下列步驟：

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

若要在使用 [CreateApplication](#) API 建立新的 Studio 筆記本時新增相依性 JAR 或自訂連接器，請在 CustomArtifactsConfiguration 資料類型中指定相依性 JAR 或自訂連接器的 Amazon S3 位置。若要將相依性或自訂連接器新增至現有的 Studio 筆記本，請調用 [UpdateApplication](#) API，並在

CustomArtifactsConfigurationUpdate 資料類型中指定相依性 JAR 或自訂連接器的 Amazon S3 位置。

Note

包含相依性或自訂連接器時，還必須包含其中未綁定的所有可轉移相依性。

實作使用者定義的函數

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

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

Note

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

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

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

若要在使用 [CreateApplication](#) API 建立新的 Studio 筆記本時新增 UDF JAR，請在 CustomArtifactConfiguration 資料類型中指定 JAR 位置。若要將 UDF JAR 新增至現有的 Studio 筆記本，請調用 [UpdateApplication](#) API 作業，並在 CustomArtifactsConfigurationUpdate 資料類型中指定 JAR 位置。或者，您可以使用 AWS 管理主控台 將 UDF JAR 檔案新增至 Studio 筆記本。

使用使用者定義函數的考量

- Managed Service for Apache Flink Studio 使用 [Apache Zeppelin 術語](#)，其中筆記本是指一個 Zeppelin 執行個體，可以包含多條筆記。然後，每條筆記可以包含多個段落。借助 Managed Service for Apache Flink Studio，解譯器過程在筆記本中的所有筆記間共用。因此，如果您在一條筆記中使用 [createTemporarySystemFunction](#) 執行明確的函數註冊，則可以在同一筆記本的另一條筆記中按原樣引用相同的函數註冊。

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

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

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

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

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

```
%flink(parallelism=1)  
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF  
# re-register the JAR for UDF with custom name  
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

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

- 如果 UDF JAR 包含 Flink SDK，請設定您的 Java 專案，以便 UDF 來源程式碼可以針對 Flink SDK 進行編譯，但 Flink SDK 類別本身不包含在建置成品中，例如 JAR。

您可以使用 Apache Maven 中的 `provided` 範圍、Gradle 中的 `compileOnly` 相依性宣告、SBT 中的 `provided` 範圍或 UDF 專案建置組態中的等效指令。您可以參閱 UDF jar 範例中的此 [pom](#)，該範例符合 maven 專案上的這種先決條件。如需完整的逐步教學課程，請參閱[搭配使用 SQL 函數與 Amazon Managed Service for Apache Flink、Amazon Translate 和 Amazon Comprehend 來翻譯、修訂和分析串流資料](#)。

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 Studio 執行期

本節包含如何升級 Studio 筆記本執行期的相關資訊。我們建議您一律升級至最新支援的 Studio 執行期。

將您的筆記本升級至新的 Studio 執行期

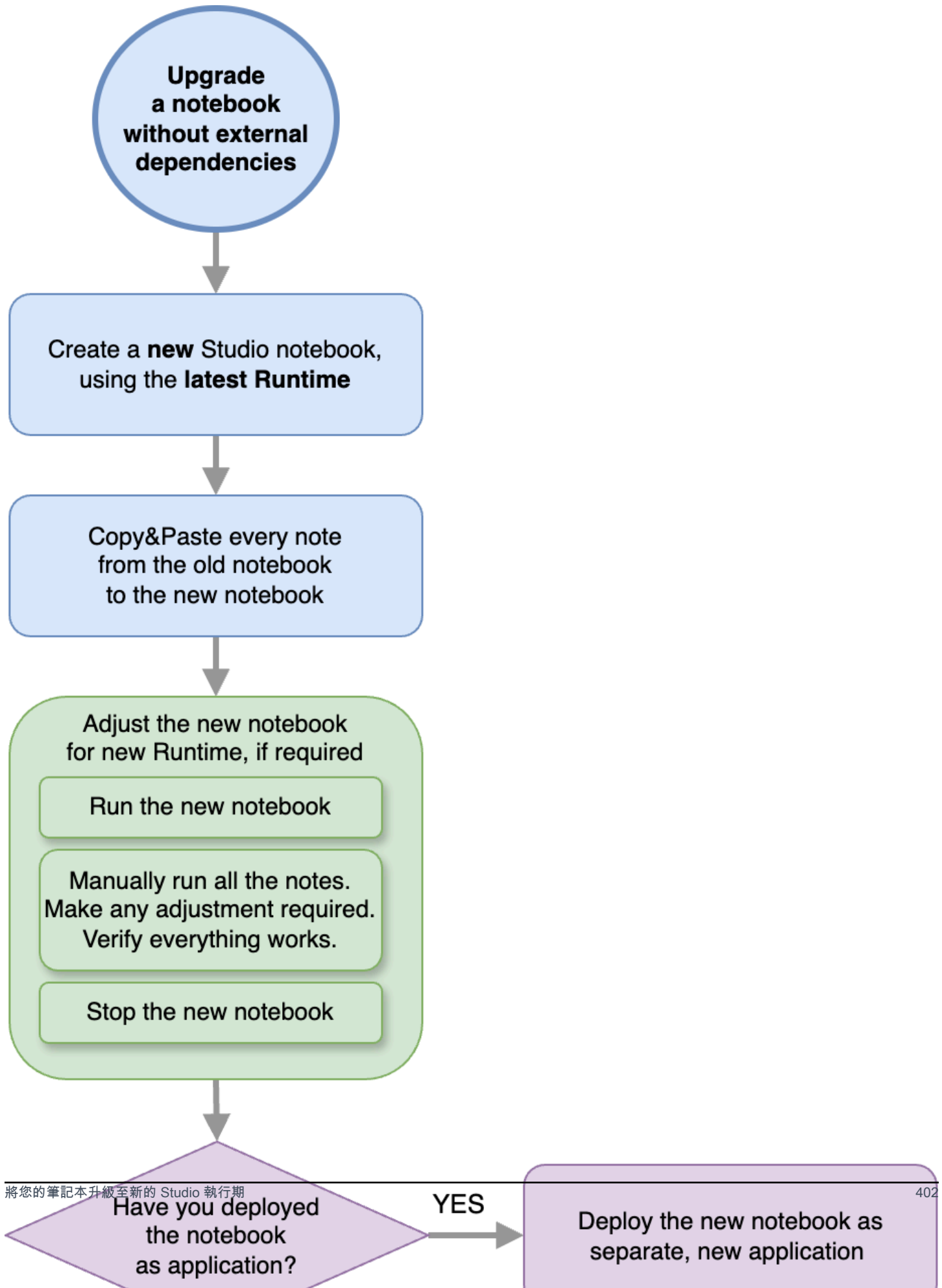
視您使用 Studio 的方式而定，升級執行期的步驟會有所不同。選取符合您使用案例的選項。

沒有外部相依性的 SQL 查詢或 Python 程式碼

如果您使用的 SQL 或 Python 沒有任何外部相依性，請使用下列執行期升級程序。我們建議您升級至最新的 Runtime 版本。升級程序與您要升級的執行期版本相同，無延遲。

1. 使用最新的執行期建立新的 Studio 筆記本。
2. 將每個備註的程式碼從舊筆記本複製並貼到新筆記本。
3. 在新的筆記本中，調整程式碼，使其與先前版本變更的任何 Apache Flink 功能相容。
 - 執行新的筆記本。開啟筆記本並依備註依序執行，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. 如果您已將舊筆記本部署為應用程式：
 - 將新的筆記本部署為獨立的新應用程式。
 - 停止舊的應用程式。
 - 在沒有快照的情況下執行新應用程式。
5. 如果舊筆記本正在執行，請將其停止。視需要啟動新的筆記本以供互動式使用。

在沒有外部相依性的情況下升級的處理流程

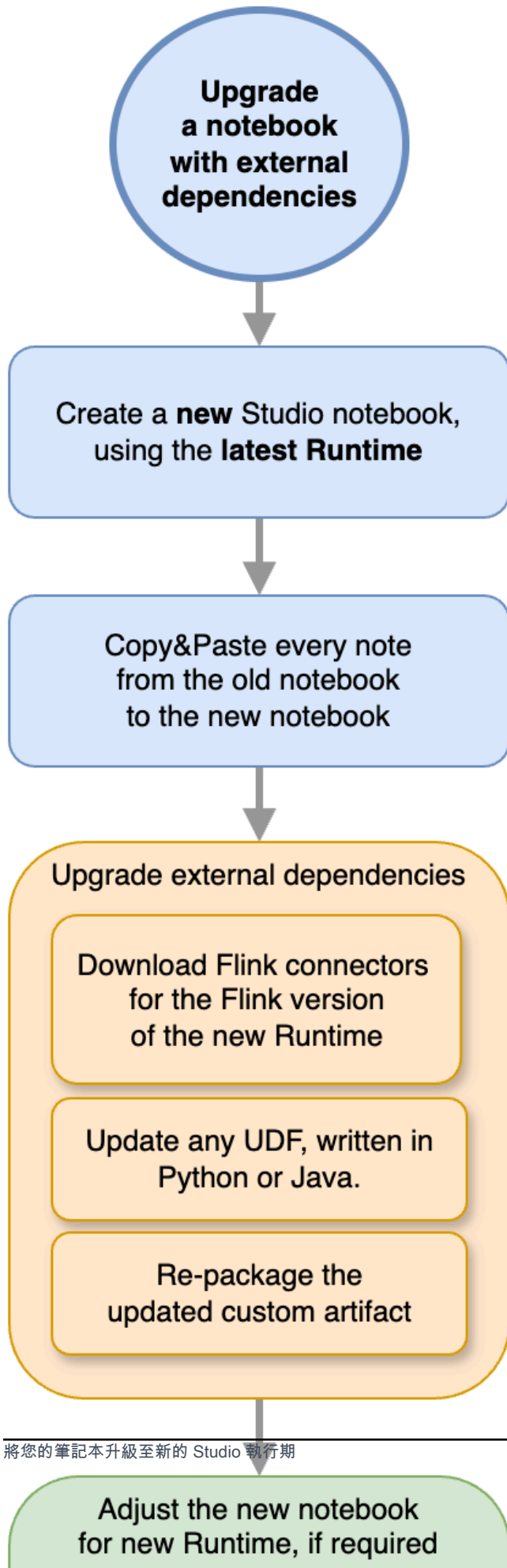


具有外部相依性的 SQL 查詢或 Python 程式碼

如果您使用 SQL 或 Python，並使用連接器或自訂成品等外部相依性，例如在 Python 或 Java 中實作的使用者定義函數，請遵循此程序。我們建議您升級至最新的執行期。無論您要從哪個執行期版本升級，此程序都相同。

1. 使用最新的執行期建立新的 Studio 筆記本。
2. 將每個備註的程式碼從舊筆記本複製並貼到新筆記本。
3. 更新外部相依性和自訂成品。
 - 尋找與新執行時間的 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 API 中的變更，以及使用者定義函數使用的任何 Python 或 JAR 相依性。重新封裝更新後的自訂成品。
 - 將這些新的連接器和成品新增至新的筆記本。
4. 在新的筆記本中，調整程式碼，使其與先前版本變更的任何 Apache Flink 功能相容。
 - 執行新的筆記本。開啟筆記本並依備註依序執行，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. 如果您已將舊筆記本部署為應用程式：
 - 將新的筆記本部署為獨立的新應用程式。
 - 停止舊的應用程式。
 - 在沒有快照的情況下執行新應用程式。
6. 如果舊筆記本正在執行，請將其停止。視需要啟動新的筆記本以供互動式使用。

使用外部相依性進行升級的程序流程



使用 AWS Glue

您的 Studio 筆記本會儲存並取得其資料來源和目的地的相關資訊 AWS Glue。建立 Studio 筆記本時，您可以指定包含連線資訊的 AWS Glue 資料庫。當您存取資料來源和目的地時，您可以指定資料庫中包含的 AWS Glue 資料表。您的 AWS Glue 資料表可讓您存取定義資料來源和目的地位置、結構描述和參數的 AWS Glue 連線。

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

如需如何設定 AWS Glue 連線、資料庫和資料表以搭配 Studio 筆記本使用的範例，請參閱[教學課程：在 Managed Service for Apache Flink 中建立 Studio 筆記本](#)教學[建立 AWS Glue 資料庫](#)課程中的。

資料表屬性

除了資料欄位，您的 AWS Glue 資料表會使用資料表屬性，將其他資訊提供給 Studio 筆記本。Managed Service for Apache Flink 使用以下 AWS Glue 資料表屬性：

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

若要將 屬性新增至 AWS Glue 資料表，請執行下列動作：

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

定義 Apache Flink 時間值

Apache Flink 提供描述何時發生串流處理事件的時間值，例如[處理時間](#)和[事件時間](#)。若要將這些值包含在應用程式輸出中，請在 AWS Glue 資料表上定義屬性，指示 Managed Service for Apache Flink 執行時間將這些值發出到指定的欄位中。

您在資料表屬性中使用的索引鍵和值如下所示：

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

使用 Flink 連接器和格式屬性

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

連接器類型	金鑰	值
Kafka	##	The format used to deserialize and serialize Kafka messages, e.g. json or csv.
	scan.startup.mode	The startup mode for the Kafka consumer, e.g. earliest-offset or timestamp .
Kinesis	##	The format used to deserialize and serialize Kinesis data

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

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。

Managed Service for Apache Flink 中 Studio 筆記本的範例和教學課程

主題

- [教學課程：在 Managed Service for Apache Flink 中建立 Studio 筆記本](#)
- [教學課程：將 Studio 筆記本部署為具有持久狀態的 Managed Service for Apache Flink 應用程式](#)
- [檢視 Studio 筆記本中分析資料的範例查詢](#)

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

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

本教學課程包含下列章節：

- [完成先決條件](#)
- [建立 AWS Glue 資料庫](#)
- [後續步驟：使用 Kinesis Data Streams 或 Amazon MSK 建立 Studio 筆記本](#)
- [使用 Kinesis Data Streams 建立 Studio 筆記本](#)
- [使用 Amazon MSK 建立 Studio 筆記本](#)

- [清除您的應用程式和相依資源](#)

完成先決條件

請確定您的 AWS CLI 是第 2 版或更新版本。若要安裝最新的 AWS CLI，請參閱[安裝、更新和解除安裝第 2 AWS CLI 版](#)。

建立 AWS Glue 資料庫

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

建立 AWS Glue 資料庫

1. 在 <https://console.aws.amazon.com/glue/> 開啟 AWS Glue 主控台。
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**。選擇建立。

後續步驟：使用 Kinesis Data Streams 或 Amazon MSK 建立 Studio 筆記本

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

- [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 MSK 建立 Studio 筆記本](#)：使用 Amazon MSK，您可以建立使用 Amazon MSK 叢集做為來源的應用程式。您需要建立一個 Amazon VPC、一個 Amazon EC2 用戶端執行個體和一個 Amazon MSK 叢集作為相依資源。

使用 Kinesis Data Streams 建立 Studio 筆記本

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

本教學課程包含下列章節：

- [完成先決條件](#)
- [建立 AWS Glue 資料表](#)
- [使用 Kinesis Data Streams 建立 Studio 筆記本](#)
- [將資料傳送至 Kinesis 資料串流](#)
- [測試 Studio 筆記本](#)

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
```

建立 AWS Glue 資料表

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

Note

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

建立資料表

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/glue/> 開啟 AWS Glue 主控台。
2. 如果您還沒有 AWS Glue 資料庫，請從左側導覽列中選擇資料庫。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，請選擇輸入 **https://kinesis.us-east-1.amazonaws.com**。如果您複製並貼上 Kinesis 來源 URL，請務必刪除任何前置或尾端空格。選擇 Next (下一步)。
6. 在分類頁面，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
####	string
##	double

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。

若要建立應用程式，您可以使用 AWS 管理主控台 或 AWS CLI。

- [使用 建立 Studio 筆記本 AWS 管理主控台](#)
- [使用 建立 Studio 筆記本 AWS CLI](#)

使用 建立 Studio 筆記本 AWS 管理主控台

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

 Note

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

3. 在建立 Studio 筆記本頁面，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，選擇執行。等待狀態顯示為執行中。筆記本執行時需支付費用。

使用 建立 Studio 筆記本 AWS CLI

若要使用 建立 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/ZeppelinRole`，並將下列許可新增至主控台自動建立的角色。

`"kinesis:GetShardIterator",`

`"kinesis:GetRecords",`

`"kinesis:ListShards"`

3. 建立稱為 `create.json` 的檔案，其中具有以下內容。使用您的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/ZeppelinRole",
  "ApplicationConfiguration": {
    "ApplicationSnapshotConfiguration": {
      "SnapshotsEnabled": false
    },
    "ZeppelinApplicationConfiguration": {
```



```

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

```

4. 若要建立應用程式，請執行下列命令：

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

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

```

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

```

6. 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:us-east-1:012345678901:application/MyNotebook\
```

將資料傳送至 Kinesis 資料串流

若要將測試資料傳送至 Kinesis 資料串流，請執行下列動作：

1. 開啟 [Kinesis 資料產生器](#)。
2. 選擇使用 CloudFormation 建立 Cognito 使用者。
3. CloudFormation 主控台會開啟 Kinesis Data Generator 範本。選擇 Next (下一步)。
4. 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)。

5. 在設定堆疊選項頁面，選擇下一步。
6. 在檢閱 Kinesis-Data-Generator-Cognito-User 頁面中，選擇我確認 AWS CloudFormation 可能會建立 IAM 資源。核取方塊。選擇 Create Stack (建立堆疊)。
7. 等待 CloudFormation 堆疊完成建立。堆疊完成後，在 CloudFormation 主控台中開啟 Kinesis-Data-Generator-Cognito-User 堆疊，然後選擇輸出索引標籤。開啟針對 KinesisDataGeneratorUrl 輸出值所列出的 URL。
8. 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。
9. 在下一頁面，提供下列值：

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

為記錄範本貼上下列程式碼：

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

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

完成下一節時，讓產生器保持執行狀態。

測試 Studio 筆記本

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

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

Apache Zeppelin 介面會在新標籤中開啟。

4. 在歡迎來到 Zeppelin! 頁面，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。

若要為應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINK 作業。如需 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 MSK 建立 Studio 筆記本

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

本教學課程包含下列章節：

- [設定 Amazon MSK 叢集](#)
- [將 NAT 閘道新增至 VPC](#)
- [建立 AWS Glue 連線和資料表](#)
- [使用 Amazon MSK 建立 Studio 筆記本](#)
- [將資料傳送至 Amazon MSK 叢集](#)
- [測試 Studio 筆記本](#)

設定 Amazon MSK 叢集

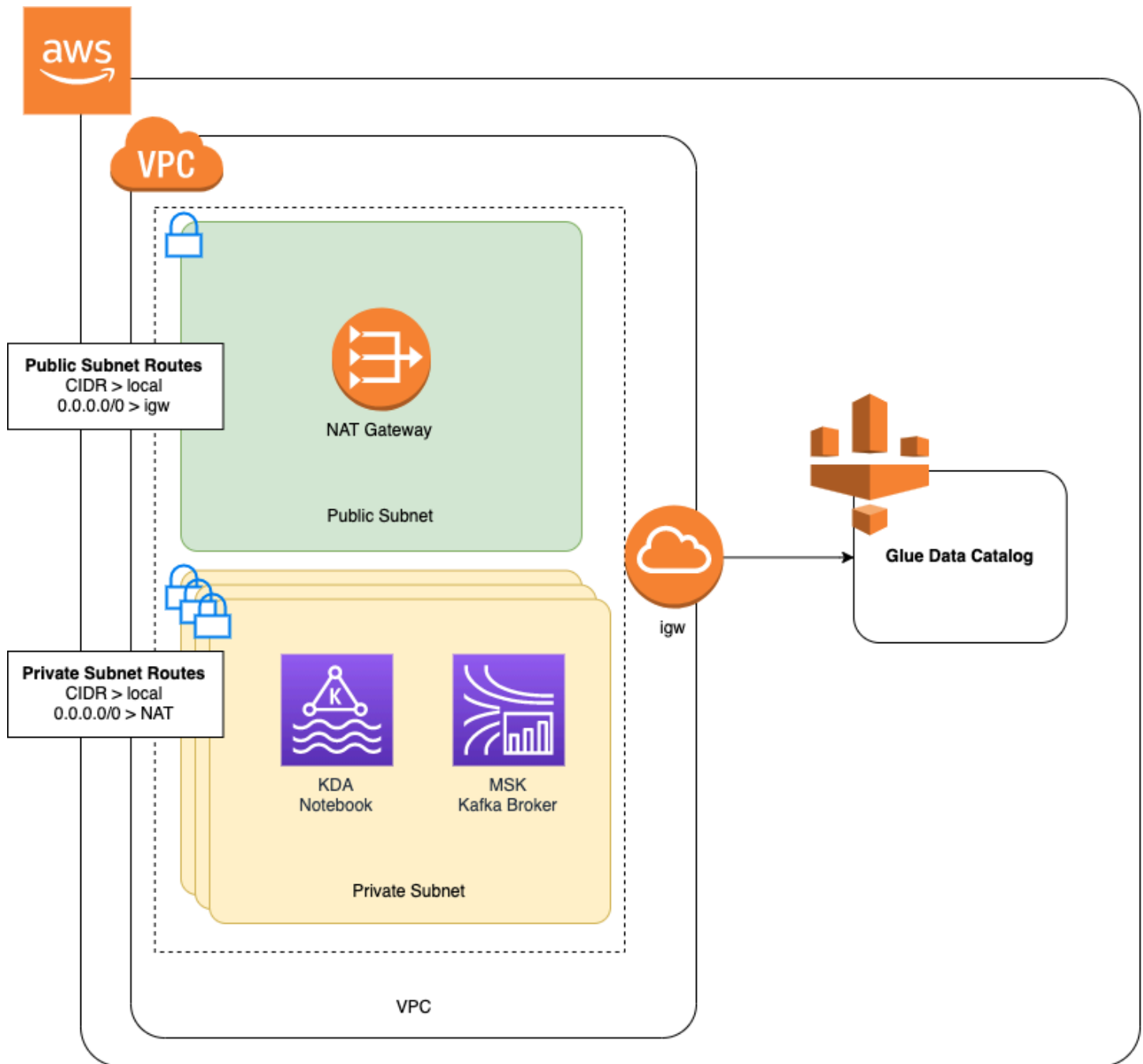
在本教學課程中，您需要一個允許純文字存取的 Amazon MSK 叢集。如果尚未設定 Amazon MSK 叢集，請依照 [Amazon MSK 使用入門](#) 教學課程來建立 Amazon VPC、Amazon MSK 叢集、主題和 Amazon EC2 用戶端執行個體。

跟隨教學課程學習時，請執行下列動作：

- 在 [步驟 3：建立 Amazon MSK 叢集](#) 的步驟 4 中，將 ClientBroker 值從 TLS 變更為 **PLAINTEXT**。

將 NAT 閘道新增至 VPC

如果依照 [Amazon MSK 使用入門](#) 教學課程建立 Amazon MSK 叢集，或者您現有的 Amazon VPC 還沒有適用於其私有子網路的 NAT 閘道，則必須將 NAT 閘道新增到 Amazon VPC。下圖顯示一般架構。



若要為您的 Amazon VPC 建立 NAT 閘道，請執行下列動作：

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

名稱：選用。

ZeppelinGateway

子網路

AWS KafkaTutorialSubnet1

彈性 IP 配置 ID

Choose an available Elastic IP. If there are no Elastic IPs available, choose 配置彈性 IP, and then choose the Elastic IP that the console creates.

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 VPC (例如AWS KafkaTutorialVPC)。

選擇建立。

8. 在路由表清單中，選擇 ZeppelinRouteTable。選擇路由標籤，然後選擇編輯路由。
9. 在編輯路由標籤中，選擇新增路由。
10. 在 中，為目標輸入 **0.0.0.0/0**。為目標選擇 NAT 閘道、ZeppelinGateway。選擇儲存路由。選擇關閉。
11. 在「路由表」頁面，已選取 ZeppelinRouteTable 時，選擇子網路關聯標籤。選擇編輯子網路關聯。
12. 在編輯子網路關聯頁面，選擇 AWS KafkaTutorialSubnet2 和 AWS KafkaTutorialSubnet3。選擇 Save (儲存)。

建立 AWS Glue 連線和資料表

您的 Studio 筆記本使用 [AWS Glue](#) 資料庫取得有關 Amazon MSK 資料來源的中繼資料。在本節中，您會建立 AWS Glue 連線，說明如何存取 Amazon MSK 叢集，以及說明如何將資料來源中的資料呈現給 Studio 筆記本等用戶端的 AWS Glue 資料表。

建立連線

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/glue/> 開啟 AWS Glue 主控台。
2. 如果您還沒有 AWS Glue 資料庫，請從左側導覽列中選擇資料庫。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 啟動伺服器 URL，請為叢集提供啟動代理程式字串。您可以從 MSK 主控台或輸入下列 CLI 命令來取得啟動代理程式：

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

- 取消核取需要 SSL 連線核取方塊。

選擇下一步。

5. 在 VPC 頁面，提供下列值：
 - 針對 VPC，選擇 VPC 的名稱（例如 AWS KafkaTutorialVPC）。
 - 對於子網路，請選擇 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所述手動建立資料表，也可以在 Apache Zeppelin 的筆記本中，使用針對 Managed Service for Apache Flink 的建立資料表連接器程式碼，透過 DDL 陳述式建立資料表。然後，您可以簽入 AWS Glue，以確保正確建立資料表。

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，請輸入您的主題名稱 (例如 AWS KafkaTutorialTopic)。針對連線，選擇 ZeppelinConnection。
4. 在分類頁面，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
####	string
##	double

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. 選擇編輯資料表並新增下列屬性：
 - 金鑰：managed-flink.proctime，值：proctime
 - 金鑰：flink.properties.group.id，值：test-consumer-group
 - 金鑰：flink.properties.auto.offset.reset，值：latest
 - 金鑰：classification，值：json

如果沒有這些鍵/值對，Flink 筆記本會執行為錯誤。

9. 選擇套用。

使用 Amazon MSK 建立 Studio 筆記本

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

您可以使用 AWS 管理主控台 或 來建立應用程式 AWS CLI。

- [使用 建立 Studio 筆記本 AWS 管理主控台](#)
- [使用 建立 Studio 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。

使用 建立 Studio 筆記本 AWS 管理主控台

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

Note

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

3. 在建立 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，選擇組態標籤。在網路模式區段中，選擇編輯。
5. 在編輯 MyNotebook 聯網頁面，選擇以 Amazon MSK 叢集為基礎的 VPC 組態。為 Amazon MSK 叢集選擇 Amazon MSK 叢集。選擇儲存變更。
6. 在 MyNotebook 頁面，選擇執行。等待狀態顯示為執行中。

使用 建立 Studio 筆記本 AWS CLI

若要使用 建立 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。
 - 子網路 ID 以及包含 Amazon MSK 叢集的 Amazon VPC 的安全群組 ID。

2. 建立稱為 `create.json` 的檔案，其中具有以下內容。使用您的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/ZeppelinRole",
  "ApplicationConfiguration": {
    "ApplicationSnapshotConfiguration": {
      "SnapshotsEnabled": false
    },
    "VpcConfigurations": [
      {
        "SubnetIds": [
          "SubnetID 1",
          "SubnetID 2",
          "SubnetID 3"
        ],
        "SecurityGroupIds": [
          "VPC Security Group ID"
        ]
      }
    ],
    "ZeppelinApplicationConfiguration": {
      "CatalogConfiguration": {
        "GlueDataCatalogConfiguration": {
          "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/default"
        }
      }
    }
  }
}
```

3. 若要建立應用程式，請執行下列命令：

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

將資料傳送至 Amazon MSK 叢集

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

- 連線到 Amazon EC2 用戶端。
- 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
```

- 輸入下列命令，在用戶端機器 AWS CLI 上設定：

```
aws configure
```

提供帳戶憑證，並為 region 提供 **us-east-1**。

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

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

BROKERS = "<<Bootstrap Broker List>>"
```

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

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

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

5. 使用下列命令執行指令碼：

```
$ python3 stock.py
```

6. 完成下一節時，讓指令碼保持執行狀態。

測試 Studio 筆記本

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

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

Apache Zeppelin 介面會在新標籤中開啟。

4. 在歡迎來到 Zeppelin! 頁面，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 Amazon MSK 叢集中的資料。

若要為應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINK 作業。如需 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](#)。

清除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 資料庫和連線

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

刪除 IAM 角色和政策

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

刪除 CloudWatch 日誌群組

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

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 從左側導覽列選擇日誌群組。
3. 選擇 /AWS/KinesisAnalytics/MyNotebook 日誌群組。
4. 選擇動作、刪除日誌群組。確認刪除。

清除 Kinesis Data Streams 資源

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

清除 MSK 資源

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

刪除您的 Amazon MSK 叢集

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

1. 開啟 Amazon MSK 主控台，網址為 <https://console.aws.amazon.com/msk/home?region=us-east-1#/home/>。
2. 選擇 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

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

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 從左側導覽列中選擇執行個體。
3. 選擇 ZeppelinClient 旁邊的核取方塊以選取它。
4. 依序選擇執行個體狀態和終止執行個體。

刪除 Amazon VPC

如果您已在本教學課程中建立 Amazon VPC，請按照以下步驟進行操作。

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

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

本教學課程包含下列章節：

- [完成事前準備](#)
- [使用 部署具有持久狀態的應用程式 AWS 管理主控台](#)
- [使用 部署具有持久狀態的應用程式 AWS CLI](#)

完成事前準備

按照[教學課程：在 Managed Service for Apache Flink 中建立 Studio 筆記本](#)建立新的 Studio 筆記本，使用 Kinesis Data Streams 或 Amazon MSK。命名 Studio 筆記本 ExampleTestDeploy。

使用 部署具有持久狀態的應用程式 AWS 管理主控台

1. 在主控台中的應用程式程式碼位置 - 選用下，新增您希望將封裝程式碼存放到的 S3 儲存貯體位置。這可讓步驟直接從筆記本部署和執行應用程式。
2. 將必要的許可新增至應用程式角色，以啟用您要用來讀取和寫入 Amazon S3 儲存貯體以及啟動 Managed Service for Apache Flink 應用程式的角色：

- AmazonS3FullAccess
- Amazonmanaged-flinkFullAccess
- 視情況存取您的來源、目的地和 VPC。如需詳細資訊，請參閱[檢閱 Studio 筆記本的 IAM 許可](#)。

3. 請參閱以下範例程式碼：

```
%flink.sql(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。您可以執行下列動作：

- 在 AWS 管理主控台中檢視 Studio 筆記本設定。
- 建立 Zeppelin 筆記，並將其匯出至 Amazon S3。此時，請為您的應用程式提供名稱，然後選擇建置和匯出。匯出完成後，您會收到通知。
- 如有需要，您可以在 Amazon S3 中的可執行檔上檢視和執行任何其他測試。

- 建置完成後，您將能夠將程式碼部署為具有持久狀態和自動調度資源的 Kinesis 串流應用程式。
- 使用下拉式選單並選擇將 Zeppelin 筆記部署為 Kinesis 串流應用程式。檢閱應用程式名稱，然後選擇透過 AWS 主控台部署。
- 這將引導您前往建立 Managed Service for Apache Flink 應用程式的 AWS 管理主控台 頁面。請注意，已預先填入應用程式名稱、平行處理層級、程式碼位置、預設 Glue DB、VPC (如果適用) 和 IAM 角色。驗證 IAM 角色是否具有來源和目的地的必要許可。快照預設啟用，以進行持久的應用程式狀態管理。
- 選擇建立應用程式。
- 您可以選擇設定並修改任何設定，然後選擇執行以啟動串流應用程式。

使用 部署具有持久狀態的應用程式 AWS CLI

若要使用 部署應用程式 AWS CLI，您必須更新 AWS CLI 以使用 Beta 2 資訊隨附的服務模型。如需如何使用更新的服務模型之相關資訊，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

```

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

```

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

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

```

檢視 Studio 筆記本中分析資料的範例查詢

下列範例查詢示範如何在 Studio 筆記本中使用視窗查詢來分析資料。

- [使用 Amazon MSK/Apache Kafka 建立資料表](#)
- [使用 Kinesis 建立資料表](#)
- [查詢輪轉時段](#)

- [查詢滑動視窗](#)
- [使用互動式 SQL](#)
- [使用 BlackHole SQL 連接器](#)
- [使用 Scala 產生範例資料](#)
- [使用互動式 Scala](#)
- [使用互動式 Python](#)
- [使用互動式 Python、SQL 和 Scala 的組合](#)
- [使用跨帳戶 Kinesis 資料串流](#)

如需 Apache Flink SQL 查詢設定的相關資訊，請參閱[在 Zeppelin 筆記本上使用 Flink 進行互動式資料分析](#)。

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

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

使用 Amazon MSK/Apache Kafka 建立資料表

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

```
-- Plaintext connection

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

-- SSL connection

CREATE TABLE your_table (
```

```

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

-- IAM connection (or for MSK Serverless)

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

```

您可以將這些與 [Apache Kafka SQL 連接器](#) 的其他屬性結合使用。

使用 Kinesis 建立資料表

下列範例示範如何使用 Kinesis 建立資料表：

```

CREATE TABLE KinesisTable (
  `column1` BIGINT,

```

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

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

查詢輪轉時段

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

```
%flink.sql(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 ZeppelinTopic 資料表中選取每個五秒滑動視窗中的最高價格：

```
%flink.sql(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 Scala 產生範例資料”](#) 執行中的範例資料產生指令碼。若要在您的 Studio 筆記本中嘗試其他 SQL 查詢（例如篩選和聯結），請參閱《Apache Flink 文件》中的[查詢](#)。

```
%flink.sql(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.sql(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 BlackHole SQL 連接器的相關資訊，請參閱《Apache Flink 文件》中的 [BlackHole SQL 連接器](#)。在此範例中，預設目錄是記憶體內目錄。

```
%flink.sql

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 Scala 產生範例資料

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`

```

使用互動式 Scala

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

```

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

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

```



```
    return table;
  }
}
```

```
%flink(parallelism=4)

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

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

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

```
%flink(parallelism=4)

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

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

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

```
SELECT * FROM `query02`
```

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》中的[資料表 API](#)。

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

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

Table.as_view = as_view
```

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

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

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

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

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

# A view that computes many records from the `key-values` we have seen so far, along
# with the current processing and event time
```

```

st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")

```

```

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

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

```

使用互動式 Python、SQL 和 Scala 的組合

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

```

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

```

```
%flink.ssql
```

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

```
%flink()

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

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

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

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

```
%flink.sql(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 Kinesis 連接器中的 create table DDL 陳述式中，使用 `aws.credentials.provider`、`aws.credentials.role.arn` 和 `aws.credentials.role.sessionName`，針對資料串流建立資料表。

對 Studio 筆記本帳戶使用下列服務執行角色。

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

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012: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'  
)
```

Managed Service for Apache Flink 的 Studio 筆記本疑難排解

本節包含 Studio 筆記本的疑難排解資訊。

停止停滯的應用程式

若要停止停滯在暫時狀態的應用程式，請在 Force 參數設定為 true 的情況下呼叫 [StopApplication](#) 動作。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](#)。

在無網際網路存取的 VPC 中，部署為具有持久狀態的應用程式

Managed Service for Apache Flink Studio 部署為應用程式功能不支援沒有網際網路存取的 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. `jarfile: lib/PythonApplicationDependencies.jar`
4. `managed.deploy_as_app.options`
 - `DatabaseARN: <glue database ARN (Amazon Resource Name)>`
5. 您可能需要為應用程式使用的服務授與使用 Managed Service for Apache Flink Studio 和 Managed Service for Apache Flink IAM 角色的許可。您可以為兩個應用程式使用相同的 IAM 角色。

部署為應用程式的大小和建置時間減少

適用於 Python 應用程式的 Studio 部署作為應用程式功能會封裝 Python 環境中的所有可用內容，因為我們無法確定您需要哪些程式庫。這可能導致超出需要的部署即應用程式大小。下列程序示範如何透過解除安裝相依性來減少部署即應用程式形式之 Python 應用程式的大小。

如果您正在建置具有 Studio 部署即應用程式功能的 Python 應用程式，不妨考慮從系統中移除預先安裝的 Python 套件，如果您的應用程式不依賴該套件的話。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，還可以縮短建置具有部署即應用程式功能之應用程式的時間。

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 範例：在使用部署即應用程式功能部署 Python 應用程式之前移除 scipy 套件。

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
```

取消任務

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

取消單一作業：

```
%flink.pyflink
import requests

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

取消所有執行中作業：

```
%flink.pyflink
import requests

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

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

取消所有作業：

```
%flink.pyflink
import requests

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

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

重新啟動 Apache Flink 解譯器

在 Studio 筆記本中重新啟動 Apache Flink 解譯器

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

為 Managed Service for Apache Flink Studio 筆記本建立自訂 IAM 政策

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

Note

在下列政策範例中，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 Logs](#)
- [Kinesis 串流](#)
- [Amazon MSK 叢集](#)

AWS Glue

下列範例政策會授予存取 AWS Glue 資料庫的許可。

JSON

```
{
  "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:us-east-1:123456789012:connection/*",
        "arn:aws:glue:us-east-1:123456789012:table/<database-name>/*",
        "arn:aws:glue:us-east-1:123456789012:database/<database-name>",
        "arn:aws:glue:us-east-1:123456789012:database/hive",
        "arn:aws:glue:us-east-1:123456789012:catalog"
      ]
    },
    {
      "Sid": "GlueDatabase",
      "Effect": "Allow",
      "Action": "glue:GetDatabases",
      "Resource": "*"
    }
  ]
}
```

CloudWatch Logs

下列範例授與存取 CloudWatch 的許可。

```
{
  "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

如果使用主控台建立應用程式，則主控台會為應用程式角色新增必要的政策，以存取 CloudWatch Logs。

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：

JSON

```
{
  "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:us-east-1:123456789012:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012: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：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kinesis:PutRecord",
        "kinesis:PutRecords",
        "kinesis:DescribeStreamSummary",
        "kinesis:DescribeStream"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012: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 叢集

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

開始使用 Amazon Managed Service for Apache Flink (DataStream API)

本節向您介紹 Managed Service for Apache Flink 的基本概念，以及使用 DataStream API 在 Java 中實作應用程式。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 Managed Service for Apache Flink 應用程式的元件](#)
- [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](#)
- [設定 AWS Command Line Interface \(AWS CLI\)](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)
- [探索其他資源](#)

檢閱 Managed Service for Apache Flink 應用程式的元件

Note

Amazon Managed Service for Apache Flink 支援所有 Apache Flink APIs 並可能支援所有 JVM 語言。如需詳細資訊，請參閱 [Flink APIs](#)。

根據您選擇的 API，應用程式和實作的結構略有不同。本入門教學課程涵蓋在 Java 中使用 DataStream API 的應用程式實作。

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java 應用程式來處理輸入，並使用 Apache Flink 執行時間產生輸出。

典型的 Managed Service for Apache Flink 應用程式具有下列元件：

- **執行期屬性：**您可以使用執行期屬性將組態參數傳遞至應用程式，以變更這些參數，而無需修改和重新發佈程式碼。

- 來源：應用程式會耗用來自一或多個來源的資料。來源使用[連接器](#)從外部系統讀取資料，例如 Kinesis 資料串流或 Kafka 儲存貯體。如需詳細資訊，請參閱[新增串流資料來源](#)。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱[運算子](#)。
- 接收器：應用程式會透過接收器將資料傳送至外部來源。接收器使用[連接器](#) v 將資料傳送至 Kinesis 資料串流、Kafka 主題、Amazon S3 或關聯式資料庫。您也可以使用特殊連接器來列印輸出，僅用於開發目的。如需詳細資訊，請參閱[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，或可能的 Java 程式庫。若要在 Amazon Managed Service for Apache Flink 中執行，應用程式必須與相依性一起封裝在 fat-jar 中，並上傳至 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，以及任何其他執行時間組態參數。

本教學課程示範如何使用 Apache Maven 封裝應用程式，以及如何在您選擇的 IDE 中於本機執行應用程式。

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 開發套件 \(JDK\) 第 11 版](#)。安裝 Java JDK 11 並設定 JAVA_HOME 環境變數以指向您的 JDK 安裝位置。如果您沒有 JDK 11，您可以使用 [Amazon Corretto 11](#) 或您選擇的任何其他標準 JDK。
 - 若要驗證是否已正確安裝 JDK，請執行下列命令。如果您使用的是 Amazon Corretto 以外的 JDK，則輸出會有所不同。請確定版本為 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](#)。如果您尚未安裝 Apache Maven。若要了解如何安裝，請參閱[安裝 Apache Maven](#)。
 - 若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

- 適用於本機開發的 IDE。我們建議您使用 [Eclipse Java Neon](#) 或 [IntelliJ IDEA](#) 等開發環境來開發和編譯應用程式。
- 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 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。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 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者[AWS 管理主控台](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶 根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

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 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 - AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的設定 AWS CLI 要使用 AWS IAM Identity Center的。 - AWS SDKs、工具和 AWS APIs，請參閱 AWS SDK 和工具參考指南中的 SDKs IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中將臨時登入資料與 AWS 資源搭配使用的指示。

哪個使用者需要程式設計存取權？	到	根據
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的 使用 IAM 使用者憑證進行身分驗證。 - AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的 使用長期憑證進行身分驗證。 - 對於 AWS APIs，請參閱《IAM 使用者指南》中的 管理 IAM 使用者的存取金鑰。

後續步驟

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

設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您會下載並設定 AWS CLI 以搭配 Managed Service for Apache Flink 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 AWS CLI 安裝，您可能需要升級以取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本教學課程中的練習需要以下 AWS CLI 版本 或更新版本：

```
aws-cli/1.16.63
```

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 AWS CLI config 檔案中為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[具名描述檔](#)。

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

如需可用 AWS 區域的清單，請參閱《》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 帳戶和 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

[建立並執行 Managed Service for Apache Flink 應用程式](#)

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

在此步驟中，您會建立 Managed Service for Apache Flink 應用程式，並將 Kinesis 資料串流做為來源和接收器。

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 Apache Flink 串流 Java 程式碼](#)
- [將範例記錄寫入輸入串流](#)
- [在本機執行您的應用程式](#)
- [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](#)
- [編譯和封裝您的應用程式程式碼](#)
- [上傳應用程式碼 JAR 檔案](#)
- [建立和設定 Managed Service for Apache Flink 應用程式](#)
- [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：

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

Note

本教學假設您正在 us-east-1 美國東部（維吉尼亞北部）區域中部署應用程式。如果您使用另一個區域，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，請使用下列命令，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 \
```

為應用程式碼建立 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](#)。使用全域唯一名稱命名 Amazon S3 儲存貯體，例如透過附加您的登入名稱。

Note

請確定您在用於本教學課程的 區域中建立儲存貯體 (us-east-1)。

其他資源

當您建立應用程式時，Managed Service for Apache Flink 會在資源不存在時自動建立下列 Amazon CloudWatch 資源：

- 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

對於開發和偵錯，您可以直接從您選擇的 IDE 在機器上執行 Apache Flink 應用程式。任何 Apache Flink 相依性都會像一般 Java 相依性一樣使用 Apache Maven 來處理。

Note

在您的開發機器上，您必須安裝 Java JDK 11、Maven 和 Git。我們建議您使用開發環境，例如 [Eclipse Java Neon](#) 或 [IntelliJ IDEA](#)。若要驗證您是否符合所有先決條件，請參閱 [滿足完成練習的先決條件](#)。您不需要在機器上安裝 Apache Flink 叢集。

驗證您的 AWS 工作階段

應用程式使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有有效的已 AWS 驗證工作階段，具有寫入 Kinesis 資料串流的許可。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 和具有有效登入資料的具名設定檔，請參閱 [設定 AWS Command Line Interface \(AWS CLI\)](#)。
2. 透過發佈下列測試記錄，確認您的 AWS CLI 已正確設定，且您的使用者具有寫入 Kinesis 資料串流的許可：

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. 如果您的 IDE 有要整合的外掛程式 AWS，您可以使用它將登入資料傳遞至在 IDE 中執行的應用程式。如需詳細資訊，請參閱 [AWS Toolkit for IntelliJ IDEA](#) 和 [AWS Toolkit for Eclipse](#)。

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 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。

檢閱應用程式元件

應用程式完全在 `com.amazonaws.services.msf.BasicStreamingJob` 類別中實作。`main()` 方法定義處理串流資料和執行資料的資料流程。

Note

為了獲得最佳化的開發人員體驗，應用程式設計為在 Amazon Managed Service for Apache Flink 和本機上執行，在 IDE 中進行開發時不會有任何程式碼變更。

- 若要讀取執行時間組態，以便在 Amazon Managed Service for Apache Flink 和 IDE 中執行時運作，應用程式會自動偵測它是否在 IDE 中於本機獨立執行。在這種情況下，應用程式會以不同的方式載入執行期組態：
 1. 當應用程式偵測到其在您的 IDE 中以獨立模式執行時，請形成包含在專案資源資料夾中 `application_properties.json` 的檔案。檔案的內容如下。
 2. 當應用程式在 Amazon Managed Service for Apache Flink 中執行時，預設行為會從您在 Amazon Managed Service for Apache Flink 應用程式中定義的執行期屬性載入應用程式組態。請參閱 [建立和設定 Managed Service for Apache Flink 應用程式](#)。

```
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();
    }
}

```

- `main()` 方法定義應用程式資料流程並執行它。
- 初始化預設串流環境。在此範例中，我們會示範如何建立 `StreamExecutionEnvironment` 要與 `DataStream API` 搭配使用的，以及 `StreamTableEnvironment` 要與 `SQL` 和資料表 `API` 搭配使用的。兩個環境物件是相同執行時間環境的兩個不同參考，以使用不同的 `APIs`。

```

StreamExecutionEnvironment env =
    StreamExecutionEnvironment.getExecutionEnvironment();

```

- 載入應用程式組態參數。這會自動從正確的位置載入它們，視應用程式執行的位置而定：

```

Map<String, Properties> applicationParameters = loadApplicationProperties(env);

```

- 應用程式使用 [Kinesis Consumer](#) 連接器定義來源，以從輸入串流讀取資料。輸入串流的組態在 `PropertyGroupId=` 中定義 `InputStream0`。串流的名稱和區域分別位於名為 `stream.name` 和的屬性中 `aws.region`。為了簡化，此來源會將記錄讀取為字串。

```

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");
    ...
}

```

- 應用程式接著會使用 [Kinesis Streams Sink](#) 連接器定義接收器，將資料傳送至輸出串流。輸出串流名稱和區域在 `PropertyGroupId=` 中定義 `OutputStream0`，類似於輸入串流。接收器會直接連接到從來源 `DataStream` 取得資料的內部。在實際的應用程式中，您會在來源和接收之間進行一些轉換。

```
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，執行您剛定義的資料流程。這必須是 `main()` 方法的最後一個指示，在您定義所有運算子之後，資料流程需要：

```
env.execute("Flink streaming Java API skeleton");
```

使用 pom.xml 檔案

`pom.xml` 檔案會定義應用程式所需的所有相依性，並設定 Maven Shade 外掛程式來建置包含 Flink 所需所有相依性的 `fat-jar`。

- 有些相依性具有 `provided` 範圍。當應用程式在 Amazon Managed Service for Apache Flink 中執行時，這些相依性會自動可用。它們需要用來編譯應用程式，或在 IDE 中本機執行應用程式。如需詳細資訊，請參閱 [在本機執行您的應用程式](#)。請確定您使用的 Flink 版本與您在 Amazon Managed Service for Apache Flink 中使用的執行時間相同。

```
<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>

```

- 您必須使用預設範圍將額外的 Apache Flink 相依性新增至 pom，例如此應用程式使用的 [Kinesis 連接器](#)。如需詳細資訊，請參閱[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 編譯器外掛程式可確保程式碼是根據 Apache Flink 目前支援的 JDK 版本 Java 11 編譯。
- Maven Shade 外掛程式會封裝 fat-jar，不包括執行時間提供的一些程式庫。它也會指定兩個轉換器：ServicesResourceTransformer 和 ManifestResourceTransformer。後者會設定類別，其中包含啟動應用程式的 main 方法。如果您重新命名主類別，請不要忘記更新此轉換器。

```

<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>

```

將範例記錄寫入輸入串流

在本節中，您將傳送範例記錄到串流，供應用程式處理。您可以使用 Python 指令碼或 [Kinesis Data Generator](#) 產生範例資料。

使用 Python 指令碼產生範例資料

您可以使用 Python 指令碼將範例記錄傳送至串流。

Note

若要執行此 Python 指令碼，您必須使用 Python 3.x 並安裝 [AWS 適用於 Python \(Boto\) 的 SDK](#) 程式庫。

若要開始將測試資料傳送至 Kinesis 輸入串流：

1. 從資料產生器 [GitHub 儲存庫下載資料產生器](#) stock.py Python 指令碼。
2. 執行 stock.py 指令碼：

```
$ python stock.py
```

在您完成教學課程的其餘部分時，請讓指令碼保持執行。您現在可以執行 Apache Flink 應用程式。

使用 Kinesis Data Generator 產生範例資料

或者，若要使用 Python 指令碼，您可以使用 [託管版本](#) 中也提供的 [Kinesis Data Generator](#)，將隨機範例資料傳送至串流。Kinesis Data Generator 會在您的瀏覽器中執行，您不需要在機器上安裝任何項目。

若要設定和執行 Kinesis Data Generator：

1. 遵循 [Kinesis Data Generator 文件](#) 中的指示來設定對工具的存取。您將執行設定使用者和密碼的 CloudFormation 範本。
2. 透過 CloudFormation 範本產生的 URL 存取 Kinesis Data Generator。CloudFormation 範本完成後，您可以在輸出索引標籤中找到 URL。
3. 設定資料產生器：
 - 區域：選取您用於本教學課程的區域：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，並確認產生的記錄與以下內容類似：

```
{ "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 Data Generator 現在正在將資料傳送至 `ExampleInputStream`。

在本機執行您的應用程式

您可以在 IDE 中的本機執行和偵錯 Flink 應用程式。

Note

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](#)。此外，請確認您具有從兩個串流讀取和寫入的許可。請參閱 [驗證您的 AWS 工作階段](#)。設定本機開發環境需要 Java 11 JDK、Apache Maven 和 以及適用於 Java 開發的 IDE。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 IDE

若要開始在 IDE 中使用應用程式，您必須將其匯入為 Java 專案。

您複製的儲存庫包含多個範例。每個範例都是個別的專案。在本教學課程中，將 `./java/GettingStarted` 子目錄中的內容匯入 IDE。

使用 Maven 將程式碼插入為現有的 Java 專案。

 Note

匯入新 Java 專案的確切程序取決於您使用的 IDE。

檢查本機應用程式組態

在本機執行時，應用程式會使用下專案資源資料夾中 `application_properties.json` 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"
    }
  }
]
```

設定 IDE 執行組態

您可以執行主要類別，直接從 IDE 執行和偵錯 Flink 應用程式

式 `com.amazonaws.services.msf.BasicStreamingJob`，就像執行任何 Java 應用程式一樣。

在執行應用程式之前，您必須設定執行組態。設定取決於您使用的 IDE。例如，請參閱 IntelliJ IDEA 文件中的[執行/偵錯組態](#)。特別是，您必須設定下列項目：

1. 將 **provided** 相依性新增至 classpath。這是必要的，以確保在本機執行時，具有 provided 範圍的相依性會傳遞至應用程式。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 登入資料以存取應用程式的 Kinesis 串流。最快的方法是使用 [AWS Toolkit for IntelliJ IDEA](#)。在執行組態中使用此 IDE 外掛程式，您可以選取特定 AWS 設定檔。AWS 身分驗證會使用此設定檔進行。您不需要直接傳遞 AWS 登入資料。

3. 確認 IDE 使用 JDK 11 執行應用程式。

在 IDE 中執行應用程式

設定 的執行組態後`BasicStreamingJob`，您可以像一般 Java 應用程式一樣執行或偵錯組態。

Note

您無法`java -jar ...`直接從命令列使用 執行 Maven 產生的 fat-jar。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。這後面接著一些 INFO 和一些 WARN 日誌，Flink 通常會在應用程式啟動時發出。

```
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日誌項目。資料流動時，不會發出日誌。

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日誌是 Flink 應用程式的正常行為。在每個記錄上發出日誌對於偵錯可能很方便，但在生產環境中執行時可能會增加大量的額外負荷。

觀察 Kinesis 串流中的輸入和輸出資料

您可以使用 Amazon Kinesis 主控台內的資料檢視器，觀察（產生範例 Python）或 Kinesis Data Generator（連結）傳送至輸入串流的記錄。Amazon Kinesis

若要觀察記錄

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 確認您執行本教學課程的區域相同，預設為 us-east-1 美國東部（維吉尼亞北部）。如果區域不符合，請變更區域。
3. 選擇資料串流。
4. 選取您要觀察的串流，或 ExampleInputStream ExampleOutputStream。
5. 選擇資料檢視器標籤。
6. 選擇任何碎片，保持最新為開始位置，然後選擇取得記錄。您可能會看到「找不到此請求的記錄」錯誤。若是如此，請選擇重試取得記錄。發佈至串流顯示的最新記錄。
7. 選擇資料欄中的值，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 IDE 中執行的應用程式。IDE 通常提供「停止」選項。確切的位置和方法取決於您使用的 IDE。

編譯和封裝您的應用程式程式碼

在本節中，您會使用 Apache Maven 編譯 Java 程式碼，並將其封裝成 JAR 檔案。您可以使用 Maven 命令列工具或 IDE 來編譯和封裝程式碼。

若要使用 Maven 命令列編譯和封裝：

移至包含 Java GettingStarted 專案的目錄，並執行下列命令：

```
$ mvn package
```

若要使用 IDE 編譯和封裝：

mvn package 從 IDE Maven 整合執行。

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 IDE 執行「建置專案」可能不會建立 JAR 檔案。

上傳應用程式碼 JAR 檔案

在本節中，您將您在上一節中建立的 JAR 檔案上傳至在本教學課程開始時建立的 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如果您尚未完成此步驟，請參閱（連結）。

上傳應用程式碼 JAR 檔案

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選擇您先前為應用程式碼建立的儲存貯體。
3. 選擇上傳。
4. 選擇 Add files (新增檔案)。
5. 導覽至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

請務必在 中選取正確的 JAR 檔案<repo-dir>/java/GettingStarted/target/amazon-msf-java-stream-app-1.0.jar。

`target` 目錄也包含您不需要上傳的其他 JAR 檔案。

建立和設定 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。在本教學課程中，您將使用 主控台。

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您可以分別建立這些資源。

主題

- [建立應用程式](#)
- [編輯 IAM 政策](#)
- [設定應用程式](#)
- [執行應用程式](#)
- [觀察執行中應用程式的指標](#)
- [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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
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. 開啟右側的選單，然後選擇 Apache Flink 應用程式，然後選擇建立串流應用程式。或者，在初始頁面的入門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。

- Apache Flink 組態、Application Flink 版本：選擇 Apache Flink 1.20。
5. 設定您的應用程式
 - 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 **kinesis-analytics-MyApplication-us-east-1** 使用必要政策建立/更新 IAM 角色。
 6. 設定您的範本以進行應用程式設定
 - 範本：選擇開發。
 7. 選擇頁面底部的建立串流應用程式。

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-east-1**
- 角色：**kinesisanalytics-MyApplication-us-east-1**

Amazon Managed Service for Apache Flink 先前稱為 Kinesis Data Analytics。自動建立的資源名稱會加上字首，**kinesis-analytics-**以確保回溯相容性。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

編輯政策

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-east-1** 政策。
3. 選擇編輯，然後選擇 JSON 索引標籤。
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**)。

JSON

```
{
  "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成品。

編輯組態

1. 在 MyApplication 頁面，選擇設定。
2. 在應用程式碼位置區段中：
 - 針對 Amazon S3 儲存貯體，選取您先前為應用程式碼建立的儲存貯體。選擇瀏覽並選取正確的儲存貯體，然後選取選擇。請勿按一下儲存貯體名稱。
 - 對於 Amazon S3 物件的路徑，請輸入 **amazon-msf-java-stream-app-1.0.jar**。
3. 針對存取許可，選擇 **kinesis-analytics-MyApplication-us-east-1** 使用必要政策建立/更新 IAM 角色。
4. 在執行期屬性區段中，新增下列屬性。

5. 選擇新增項目並新增下列每個參數：

群組 ID	金鑰	值
InputStream0	stream.name	ExampleInputStream
InputStream0	aws.region	us-east-1
OutputStream0	stream.name	ExampleOutputStream
OutputStream0	aws.region	us-east-1

6. 請勿修改任何其他區段。

7. 選擇儲存變更。

 Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

執行應用程式

應用程式現在已設定並準備好執行。

執行應用程式

1. 在 Amazon Managed Service for Apache Flink 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，選擇使用最新的快照執行，然後選擇執行。

應用程式詳細資訊中的狀態會從 Ready 轉換為 Starting，然後在應用程式啟動Running時轉換為。

當應用程式處於 Running 狀態時，您現在可以開啟 Flink 儀表板。

開啟 儀表板

1. 選擇開啟 Apache Flink 儀表板。儀表板會在新頁面上開啟。
2. 在執行中任務清單中，選擇您可以看到的單一任務。

Note

如果您設定執行期屬性或不正確地編輯 IAM 政策，應用程式狀態可能會變成 Running，但 Flink 儀表板會顯示任務正在持續重新啟動。如果應用程式設定錯誤或缺少存取外部資源的許可，這是常見的失敗案例。

發生這種情況時，請檢查 Flink 儀表板中的例外狀況索引標籤，以查看問題的原因。

觀察執行中應用程式的指標

在 MyApplication 頁面的 Amazon CloudWatch 指標區段中，您可以從執行中的應用程式查看一些基本指標。

檢視指標

1. 在重新整理按鈕旁，從下拉式清單中選取 10 秒。
2. 當應用程式執行正常時，您可以看到運作時間指標持續增加。
3. fullrestarts 指標應為零。如果增加，組態可能會發生問題。若要調查問題，請檢閱 Flink 儀表板上的例外狀況索引標籤。
4. 在運作狀態良好的應用程式中，失敗檢查點指標的數量應為零。

Note

此儀表板會顯示一組固定的指標，精細程度為 5 分鐘。您可以使用 CloudWatch 儀表板中的任何指標來建立自訂應用程式儀表板。

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 Data Generator 將資料發佈至輸入。

您現在可以使用 <https://console.aws.amazon.com/kinesis/> 中的資料檢視器來觀察 Managed Service for Apache Flink 上執行之應用程式的輸出，與先前已執行的操作類似。

檢視輸出

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 確認區域與您用來執行本教學課程的區域相同。根據預設，它是 us-east-1US East (N. Virginia)。視需要變更 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，保持最新為開始位置，然後選擇取得記錄。您可能會看到「找不到此請求的記錄」錯誤。若是如此，請選擇重試取得記錄。發佈至串流顯示的最新記錄。
7. 選取資料欄中的值，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，請前往名為的 Managed Service for Apache Flink 應用程式的主控台頁面MyApplication。

停止應用程式

1. 從動作下拉式清單中，選擇停止。
2. 應用程式詳細資訊中的狀態會從 轉換為 Running Stopping，然後在應用程式完全停止Ready時轉換為。

Note

別忘了也要停止從 Python 指令碼或 Kinesis Data Generator 將資料傳送至輸入串流。

下一步驟

[清除 AWS 資源](#)

清除 AWS 資源

本節包含清除本入門 (DataStream API) 教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [探索 Apache Flink 的其他資源](#)

刪除 Managed Service for Apache Flink 應用程式

請使用下列程序刪除應用程式。

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，然後開啟位於 <https://console.aws.amazon.com/flink> 的 Amazon MSF 主控台。
2. 選擇資料串流。
3. 選取您建立的兩個串流，ExampleInputStream以及 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 物件和儲存貯體

使用下列程序刪除 Amazon S3 物件和儲存貯體。

從 S3 儲存貯體刪除物件

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
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成品，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. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選取您為成品建立的儲存貯體。
3. 選擇刪除並確認刪除。

Note

S3 儲存貯體必須為空白，才能將其刪除。

刪除您的 IAM 資源

刪除 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-east-1 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-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. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

探索 Apache 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。

- [Amazon Managed Service for Apache Flink 研討會](#)：在本研討會中，您將建置end-to-end串流架構，以近乎即時的方式擷取、分析和視覺化串流資料。您著手改善紐約市一家計程車公司的運營。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。
- [建立和使用 Managed Service for Apache Flink 應用程式的範例](#)：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逐步指示，可協助您建立 Managed Service for Apache Flink 應用程式並測試結果。
- [學習 Flink：動手訓練](#)：官方介紹性 Apache Flink 訓練課程，可協助您開始撰寫可擴展的串流 ETL、分析和事件驅動型應用程式。

開始使用 Amazon Managed Service for Apache Flink (資料表 API)

本節介紹 Managed Service for Apache Flink 的基本概念，以及使用資料表 API 和 SQL 在 Java 中實作應用程式。它示範如何在相同應用程式中切換不同的 APIs，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 Managed Service for Apache Flink 應用程式的元件](#)
- [完成必要的先決條件](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [下一步驟](#)
- [清除 AWS 資源](#)
- [探索其他資源](#)

檢閱 Managed Service for Apache Flink 應用程式的元件

Note

Managed Service for Apache Flink 支援所有 [Apache Flink APIs](#) 並可能支援所有 JVM 語言。根據您選擇的 API，應用程式和實作的結構略有不同。本教學課程涵蓋使用資料表 API 和 SQL 的應用程式實作，以及與在 Java 中實作之 DataStream API 的整合。

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java 應用程式來處理輸入，並使用 Apache Flink 執行時間產生輸出。

典型的 Apache Flink 應用程式具有下列元件：

- 執行期屬性：您可以使用執行期屬性將組態參數傳遞至應用程式，而無需修改和重新發佈程式碼。
- 來源：應用程式會耗用來自一或多個來源的資料。來源使用[連接器](#)從 和外部系統讀取資料，例如 Kinesis 資料串流或 Amazon MSK 主題。對於開發或測試，您也可以讓來源隨機 **【產生測試資料】**。如需詳細資訊，請參閱[將串流資料來源新增至 Managed Service for Apache Flink](#)。使用 SQL 或資料表 API，來源會定義為來源資料表。

- 轉換：應用程式會透過一個或多個可篩選、豐富或彙總資料的轉換來處理資料。使用 SQL 或資料表 API 時，轉換會定義為透過資料表或檢視的查詢。
- 接收器：應用程式會透過接收器將資料傳送至外部系統。接收器使用[連接器](#)將資料傳送至外部系統，例如 Kinesis 資料串流、Amazon MSK 主題、Amazon S3 儲存貯體或關聯式資料庫。您也可以使用特殊連接器來列印輸出，僅用於開發目的。使用 SQL 或資料表 API 時，目的地會定義為您要插入結果的目的地資料表。如需詳細資訊，請參閱[在 Managed Service for Apache Flink 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，或可能的 Java 程式庫。若要在 Amazon Managed Service for Apache Flink 中執行，您必須將應用程式與相依性封裝在 fat-JAR 中，並將其上傳至 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，以及其他執行時間組態參數。本教學課程示範如何使用 Apache Maven 封裝應用程式，以及如何在您選擇的 IDE 中於本機執行應用程式。

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 Amazon Managed Service for Apache Flink \(DataStream API\)](#) 中的前兩個步驟：

- [滿足完成練習的先決條件](#)
- [設定 AWS Command Line Interface \(AWS CLI\)](#)

若要開始使用，請參閱 [建立應用程式](#)。

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

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

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 Apache Flink 串流 Java 程式碼](#)
- [在本機執行您的應用程式](#)

- [觀察應用程式將資料寫入 S3 儲存貯體](#)
- [停止應用程式在本機執行](#)
- [編譯和封裝您的應用程式程式碼](#)
- [上傳應用程式碼 JAR 檔案](#)
- [建立和設定 Managed Service for Apache Flink 應用程式](#)

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：

- Amazon S3 儲存貯體，用於存放應用程式的程式碼和寫入應用程式輸出。

Note

本教學假設您正在 us-east-1 區域中部署應用程式。如果您使用另一個區域，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名稱。

Note

請確定您在用於本教學課程的區域中建立儲存貯體。教學課程的預設值為 us-east-1。

其他資源

建立應用程式時，Managed Service for Apache Flink 會建立下列 Amazon CloudWatch 資源 (如果尚不存在該資源)：

- 名為 /AWS/KinesisAnalytics-java/<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

對於開發和偵錯，您可以直接從您選擇的 IDE 在機器上執行 Apache Flink 應用程式。任何 Apache Flink 相依性都會使用 Maven 做為一般 Java 相依性處理。

Note

在您的開發機器上，您必須安裝 Java JDK 11、Maven 和 Git。我們建議您使用開發環境，例如 [Eclipse Java Neon](#) 或 [IntelliJ IDEA](#)。若要驗證您是否符合所有先決條件，請參閱 [滿足完成練習的先決條件](#)。您不需要在機器上安裝 Apache Flink 叢集。

驗證您的 AWS 工作階段

應用程式使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有有效的已 AWS 驗證工作階段，具有寫入 Kinesis 資料串流的許可。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 和具有有效登入資料的具名設定檔，請參閱 [設定 AWS Command Line Interface \(AWS CLI\)](#)。
2. 如果您的 IDE 有要整合的外掛程式 AWS，您可以使用它將登入資料傳遞至在 IDE 中執行的應用程式。如需詳細資訊，請參閱 [AWS Toolkit for IntelliJ IDEA](#) 和 [AWS Toolkit for 編譯應用程式或執行 Eclipse](#)。

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的應用程式碼可從 GitHub 取得。

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。

檢閱應用程式元件

應用程式完全在 `com.amazonaws.services.msf.BasicTableJob` 類別中實作。`main()` 方法定義來源、轉換和接收器。此方法結尾的執行陳述式會啟動執行。

Note

為了獲得最佳的開發人員體驗，應用程式設計為在 Amazon Managed Service for Apache Flink 和本機上執行，在 IDE 中進行開發時不會進行任何程式碼變更。

- 若要讀取執行時間組態，以便在 Amazon Managed Service for Apache Flink 和 IDE 中執行時運作，應用程式會自動偵測它是否在 IDE 中本機執行獨立。在這種情況下，應用程式會以不同的方式載入執行時間組態：
 - 當應用程式偵測到其在您的 IDE 中以獨立模式執行時，請形成包含在專案資源資料夾中 `application_properties.json` 的檔案。檔案的內容如下。
 - 當應用程式在 Amazon Managed Service for Apache Flink 中執行時，預設行為會從您在 Amazon Managed Service for Apache Flink 應用程式中定義的執行期屬性載入應用程式組態。請參閱 [建立和設定 Managed Service for Apache Flink 應用程式](#)。

```
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();
    }
}
```

- `main()` 方法會定義應用程式資料流程並執行它。

- 初始化預設串流環境。在此範例中，我們會示範如何建立 `StreamExecutionEnvironment` 以與 `DataStream` API 搭配使用，以及建立 `StreamTableEnvironment` 以與 SQL 和資料表 API 搭配使用。這兩個環境物件是對相同執行時間環境的兩個不同參考，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。這會自動從正確的位置載入它們，視應用程式執行的位置而定：

```
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 [FileSystem 接收器連接器](#)。您必須啟用檢查點，才能將檔案寫入目的地。當應用程式在 Amazon Managed Service for Apache Flink 中執行時，應用程式組態會控制檢查點，並預設啟用它。相反地，在本機執行時，檢查點預設為停用。應用程式偵測到它在本機執行，並每 5,000 毫秒設定檢查點。

```
if (env instanceof LocalStreamEnvironment) {  
    env.enableCheckpointing(5000);  
}
```

- 此應用程式不會從實際外部來源接收資料。它會產生隨機資料，以透過 [DataGen 連接器](#) 進行處理。此連接器適用於 `DataStream` API、SQL 和資料表 API。為了示範 APIs 之間的整合，應用程式會使用 `DataStram` API 版本，因為它提供更多彈性。在此情況下，每個記錄都會由稱為 `StockPriceGeneratorFunction` 的產生器函數產生，您可以在其中放置自訂邏輯。

```
DataGeneratorSource<StockPrice> source = new DataGeneratorSource<>(  
    new StockPriceGeneratorFunction(),  
    Long.MAX_VALUE,  
    RateLimiterStrategy.perSecond(recordPerSecond),  
    TypeInformation.of(StockPrice.class));
```

- 在 `DataStream` API 中，記錄可以有自訂類別。類別必須遵循特定規則，讓 Flink 可以使用它們做為記錄。如需詳細資訊，請參閱[支援的資料類型](#)。在此範例中，`StockPrice` 類別是 [POJO](#)。
- 然後，來源會連接至執行環境，產生 `DataStream` 的 `StockPrice`。此應用程式不使用[事件時間語意](#)，也不會產生浮水印。執行平行處理為 1 的 `DataGenerator` 來源，獨立於應用程式的其餘部分平行處理。

```
DataStream<StockPrice> stockPrices = env.fromSource(  

```

```
source,
    WatermarkStrategy.noWatermarks(),
    "data-generator"
).setParallelism(1);
```

- 資料處理流程中的後續內容是使用資料表 API 和 SQL 來定義。為此，我們將 StockPrices 的 DataStream 轉換為資料表。資料表的結構描述會自動從 StockPrice 類別推斷。

```
Table stockPricesTable = tableEnv.fromDataStream(stockPrices);
```

- 下列程式碼片段說明如何使用程式設計資料表 API 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);
```

- 系統會定義接收器資料表，將結果寫入 Amazon S3 儲存貯體做為 JSON 檔案。為了說明以程式設計方式定義檢視的差異，使用資料表 API 時，系統會使用 SQL 定義目的地資料表。

```
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 executeInsert() 將篩選後的股票價格檢視插入目的地資料表。此方法會啟動執行我們到目前為止定義的資料流程。

```
filteredStockPricesTable.executeInsert("s3_sink");
```

使用 pom.xml 檔案

pom.xml 檔案會定義應用程式所需的所有相依性，並設定 Maven Shade 外掛程式來建置包含 Flink 所需所有相依性的 fat-jar。

- 有些相依性具有provided範圍。當應用程式在 Amazon Managed Service for Apache Flink 中執行時，這些相依性會自動可用。應用程式或 IDE 中的本機應用程式需要它們。如需詳細資訊，請參閱 ([更新至 TableAPI](#))[在本機執行您的應用程式](#)。請確定您使用的 Flink 版本與您在 Amazon Managed Service for Apache Flink 中使用的執行時間相同。若要使用 TableAPI 和 SQL，您必須包含 flink-table-planner-loader和 flink-table-runtime-dependencies，兩者皆包含 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>
```

- 您必須使用預設範圍將其他 Apache Flink 相依性新增至 pom。例如，[DataGen 連接器](#)、[FileSystem SQL 連接器](#)和 [JSON 格式](#)。

```
<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 wit 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 編譯器外掛程式可確保程式碼是根據 Apache Flink 目前支援的 JDK 版本 Java 11 編譯。
- Maven Shade 外掛程式會封裝 fat-jar，不包括執行時間提供的一些程式庫。它也會指定兩個轉換器：ServicesResourceTransformer 和 ManifestResourceTransformer。後者會設定類別，其中包含啟動應用程式的 main 方法。如果您重新命名主類別，請不要忘記更新此轉換器。

```
<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>
```

在本機執行您的應用程式

您可以在 IDE 中的本機執行和偵錯 Flink 應用程式。

Note

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](#)。此外，請確認您具有從兩個串流讀取和寫入的許可。請參閱 [驗證您的 AWS 工作階段](#)。

設定本機開發環境需要 Java 11 JDK、Apache Maven 和 IDE for 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 IDE

若要開始在 IDE 中使用應用程式，您必須將其匯入為 Java 專案。

您複製的儲存庫包含多個範例。每個範例都是個別的專案。在本教學課程中，將 `./jave/GettingStartedTable` 子目錄中的內容匯入 IDE。

使用 Maven 將程式碼插入為現有的 Java 專案。

Note

匯入新 Java 專案的確切程序取決於您使用的 IDE。

修改本機應用程式組態

在本機執行時，應用程式會使用下專案資源資料夾中 `application_properties.json` 檔案中的組態 `./src/main/resources`。對於此教學課程應用程式，組態參數是儲存貯體的名稱，以及寫入資料的路徑。

編輯組態並修改 Amazon S3 儲存貯體的名稱，以符合您在本教學課程開始時建立的儲存貯體。

```
[  
{
```

```
"PropertyGroupId": "bucket",
"PropertyMap": {
  "name": "<bucket-name>",
  "path": "output"
}
}
```

Note

組態屬性name只能包含儲存貯體名稱，例如 my-bucket-name。請勿包含任何字首，例如 s3://或結尾斜線。

如果您修改路徑，請省略任何正斜線或尾斜線。

設定 IDE 執行組態

您可以執行主類別，直接從 IDE 執行和偵錯 Flink 應用程式 `com.amazonaws.services.msf.BasicTableJob`，就像執行任何 Java 應用程式一樣。在執行應用程式之前，您必須設定執行組態。設定取決於您使用的 IDE。例如，請參閱 IntelliJ IDEA 文件中的[執行/偵錯組態](#)。特別是，您必須設定下列項目：

1. 將**provided**相依性新增至 classpath。這是必要的，以確保在本機執行時，具有provided範圍的相依性會傳遞至應用程式。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 登入資料以存取應用程式的 Kinesis 串流。最快的方法是使用 [AWS Toolkit for IntelliJ IDEA](#)。在執行組態中使用此 IDE 外掛程式，您可以選取特定 AWS 設定檔。AWS 身分驗證會使用此設定檔進行。您不需要直接傳遞 AWS 登入資料。
3. 確認 IDE 使用 JDK 11 執行應用程式。

在 IDE 中執行應用程式

設定 的執行組態後`BasicTableJob`，您可以像一般 Java 應用程式一樣執行或偵錯組態。

Note

您無法 `java -jar ...` 直接從命令列使用 執行 Maven 產生的 fat-jar。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。這後面接著一些 INFO 和一些 WARN 日誌，Flink 通常會在應用程式啟動時發出。

```
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日誌項目。資料流動時，不會發出日誌。

若要驗證應用程式是否正確處理資料，您可以檢查輸出儲存貯體的內容，如下節所述。

Note

不發出有關流動資料的日誌是 Flink 應用程式的正常行為。在每個記錄上發出日誌對於偵錯可能很方便，但在生產環境中執行時可能會增加大量的額外負荷。

觀察應用程式將資料寫入 S3 儲存貯體

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的地 S3 儲存貯體。除非您修改預設組態路徑，否則資料將寫入 output 路徑，後面接著資料和小時分割，格式為 `./output/<yyyy-MM-dd>/<HH>`。

[FileSystem 接收器連接器](#)會在 Flink 檢查點上建立新的檔案。在本機執行時，應用程式會每 5 秒 (5,000 毫秒) 執行檢查點，如程式碼中所指定。

```
if (env instanceof LocalStreamEnvironment) {
    env.enableCheckpointing(5000);
}
```

瀏覽 S3 儲存貯體並觀察應用程式寫入的檔案

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選擇您先前建立的儲存貯體。
3. 導覽至 output 路徑，然後導覽至與 UTC 時區中目前時間對應的日期和小時資料夾。
4. 定期重新整理以觀察每 5 秒出現的新檔案。

5. 選取並下載一個檔案來觀察內容。

Note

根據預設，檔案沒有副檔名。內容的格式為 JSON。您可以使用任何文字編輯器開啟檔案來檢查內容。

停止應用程式在本機執行

停止在 IDE 中執行的應用程式。IDE 通常提供「停止」選項。確切的位置和方法取決於 IDE。

編譯和封裝您的應用程式程式碼

在本節中，您會使用 Apache Maven 編譯 Java 程式碼，並將其封裝成 JAR 檔案。您可以使用 Maven 命令列工具或 IDE 來編譯和封裝程式碼。

使用 Maven 命令列編譯和封裝

移至包含 Java GettingStarted 專案的目錄，並執行下列命令：

```
$ mvn package
```

使用 IDE 編譯和封裝

`mvn package` 從 IDE Maven 整合執行。

在這兩種情況下，`target/amazon-msf-java-table-app-1.0.jar` 都會建立 JAR 檔案。

Note

從 IDE 執行建置專案可能不會建立 JAR 檔案。

上傳應用程式碼 JAR 檔案

在本節中，您將您在上一節中建立的 JAR 檔案上傳至在本教學課程開始時建立的 Amazon S3 儲存貯體。如果您已完成，請完成 [建立 Amazon S3 儲存貯體](#)。

上傳應用程式的程式碼

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選擇您先前為應用程式碼建立的儲存貯體。
3. 選擇上傳欄位。
4. 選擇 Add files (新增檔案)。
5. 導覽至上一節產生的 JAR 檔案：`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

請務必在 中選取正確的 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。

建立和設定 Managed Service for Apache Flink 應用程式

您可以使用 主控台或 建立和設定 Managed Service for Apache Flink 應用程式 AWS CLI。在本教學課程中，您將使用 主控台。

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 Logs 資源。當您使用 建立應用程式時 AWS CLI，您必須分別建立這些資源。

建立應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
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，完成下列操作：

- 針對選擇設定串流處理應用程式的方法，選擇從頭開始建立。
- 針對 Apache Flink 組態、Application Flink 版本，選擇 Apache Flink 1.19。
- 在應用程式組態區段中，完成下列操作：
 - 在應用程式名稱中，輸入 **MyApplication**。
 - 對於 Description (說明)，輸入 **My Java Table API test app**。
 - 針對存取應用程式資源，選擇使用必要政策建立/更新 IAM 角色 `kinesis-analytics-MyApplication-us-east-1`。
- 在應用程式設定的範本中，完成下列操作：
 - 針對範本，選擇開發。

5. 選擇建立串流應用程式。

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-east-1`
- 角色：`kinesisanalytics-MyApplication-us-east-1`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 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一節所建立的 **kinesis-analytics-service-MyApplication-us-east-1** 政策。
3. 選擇編輯，然後選擇 JSON 索引標籤。
4. 將下列政策範例的反白部分新增至政策。將範例帳戶 ID (`012345678901`) 取代為您的帳戶 ID，並將 `<bucket-name>` 取代為您建立的 S3 儲存貯體名稱。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/kinesis-analytics-
placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
```

```

        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
      ]
    },
    {
      "Sid": "WriteOutputBucket",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket2"
      ]
    }
  ]
}

```

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碼成品。

設定應用程式

1. 在 MyApplication 頁面，選擇設定。
2. 在複寫程式碼位置區段中，選擇設定。
 - 針對 Amazon S3 儲存貯體，選取您先前為應用程式碼建立的儲存貯體。選擇瀏覽並選擇正確的儲存貯體，然後選擇選擇。請勿按一下儲存貯體名稱。
 - 對於 Amazon S3 物件的路徑，請輸入 **amazon-msf-java-table-app-1.0.jar**。
3. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-east-1**。
4. 在執行期屬性區段中，新增下列屬性。
5. 選擇新增項目並新增下列每個參數：

群組 ID	金鑰	值
bucket	name	your-bucket-name
bucket	path	output

- 請勿修改任何其他設定。
- 選擇儲存變更。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

應用程式現在已設定並準備好執行。

執行應用程式

- 返回 Amazon Managed Service for Apache Flink 中的主控台頁面，然後選擇 MyApplication。
- 選擇執行以啟動應用程式。
- 在應用程式還原組態上，選擇使用最新的快照執行。
- 選擇執行。
- 應用程式詳細資訊中的狀態會在應用程式啟動Running後從 轉換為 Ready Starting，然後轉換為。

當應用程式處於 Running 狀態時，您可以開啟 Flink 儀表板。

開啟儀表板並檢視任務

- 選擇開啟 Apache Flink 破折號。儀表板會在新頁面中開啟。

2. 在執行中任務清單中，選擇您可以看到的單一任務。

Note

如果您未正確設定執行時間屬性或編輯 IAM 政策，應用程式狀態可能會變更為 Running，但 Flink 儀表板會顯示任務持續重新啟動。當應用程式設定錯誤或缺少存取外部資源的許可時，這是常見的失敗案例。發生這種情況時，請檢查 Flink 儀表板中的例外狀況索引標籤，以調查問題的原因。

觀察執行中應用程式的指標

在 MyApplication 頁面的 Amazon CloudWatch 指標區段中，您可以從執行中的應用程式查看一些基本指標。

檢視指標

1. 在重新整理按鈕旁，從下拉式清單中選取 10 秒。
2. 當應用程式執行正常時，您可以看到運作時間指標持續增加。
3. fullrestarts 指標應為零。如果增加，組態可能會發生問題。檢閱 Flink 儀表板上的例外狀況索引標籤以調查問題。
4. 在運作狀態良好的應用程式中，失敗檢查點指標的數量應為零。

Note

此儀表板會顯示一組固定的指標，精細程度為 5 分鐘。您可以使用 CloudWatch 儀表板中的任何指標來建立自訂應用程式儀表板。

觀察應用程式將資料寫入目的地儲存貯體

您現在可以觀察在 Amazon Managed Service for Apache Flink 中執行的應用程式，將檔案寫入 Amazon S3。

若要觀察檔案，請遵循您在本機執行應用程式時用來檢查所寫入檔案的相同程序。請參閱 [觀察應用程式將資料寫入 S3 儲存貯體](#)。

請記住，應用程式會在 Flink 檢查點上寫入新檔案。在 Amazon Managed Service for Apache Flink 上執行時，預設會啟用檢查點，並每 60 秒執行一次。應用程式大約每 1 分鐘建立新檔案。

停止應用程式

若要停止應用程式，請前往名為 的 Managed Service for Apache Flink 應用程式的主控台頁面MyApplication。

停止應用程式

1. 從動作下拉式清單中，選擇停止。
2. 應用程式詳細資訊中的狀態會從 轉換為 Running Stopping，然後在應用程式完全停止Ready時轉換為 。

Note

別忘了也要停止從 Python 指令碼或 Kinesis Data Generator 將資料傳送至輸入串流。

下一步驟

[清除 AWS 資源](#)

清除 AWS 資源

本節包含在入門（資料表 API）教學課程中建立 AWS 的資源清除程序。

本主題包含下列章節。

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

刪除 Managed Service for Apache Flink 應用程式

請使用下列程序刪除應用程式。

刪除應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在 Managed Service for Apache Flink 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 物件和儲存貯體

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. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選取您建立的 S3 儲存貯體。
3. 選取您上傳的名為的應用程式成品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. 確認您想要永久刪除內容。

刪除 S3 儲存貯體

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
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

使用下列程序刪除 IAM 資源。

刪除 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-east-1 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-east-1 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

使用下列程序刪除 CloudWatch 資源。

刪除 CloudWatch 資源

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

下一步驟

[探索其他資源](#)

探索其他資源

現在您已建立並執行使用資料表 API 的 Managed Service for Apache Flink 應用程式，請參閱[探索其他資源](#)中的[開始使用 Amazon Managed Service for Apache Flink \(DataStream API\)](#)。

Amazon Managed Service for Apache Flink for Python 入門

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

主題

- [檢閱 Managed Service for Apache Flink 應用程式的元件](#)
- [滿足先決條件](#)
- [建立並執行 Managed Service for Apache Flink for Python 應用程式](#)
- [清除 AWS 資源](#)

檢閱 Managed Service for Apache Flink 應用程式的元件

Note

Amazon Managed Service for Apache Flink 支援所有 [Apache Flink APIs](#)。根據您選擇的 API，應用程式的結構略有不同。在 Python 中開發 Apache Flink 應用程式時，一種常用的方法是使用內嵌在 Python 程式碼中的 SQL 定義應用程式流程。這是我們在下列 Gettgin Started 教學課程中遵循的方法。

為了處理資料，您的 Managed Service for Apache Flink 應用程式會使用 Python 指令碼來定義處理輸入並使用 Apache Flink 執行期產生輸出的資料流程。

典型的 Managed Service for Apache Flink 應用程式具有下列元件：

- **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：**應用程式會耗用來自一或多個來源的資料。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。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 SQL 時，應用程式會將來源定義為來源資料表。
- **轉換：**應用程式使用可篩選、擴充或彙總資料的一或多個轉換來處理資料。當您使用 SQL 時，應用程式會將轉換定義為 SQL 查詢。
- **接收器：**應用程式會透過接收器將資料傳送至外部來源。接收器使用[連接器](#)將資料傳送至外部系統，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

特殊連接器來列印輸出以供開發之用。當您使用 SQL 時，應用程式會將接收器定義為插入結果的接收器資料表。如需詳細資訊，請參閱[在 Managed Service for Apache Flink 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，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在 Amazon Managed Service for Apache Flink 上部署。

滿足先決條件

若要完成本教學課程，您必須具備下列項目：

- Python 3.11，最好使用 [VirtualEnv \(venv\)](#)、[Conda](#) 或 [Miniconda](#) 等獨立環境。
- [Git 用戶端](#) - 如果您尚未安裝 Git 用戶端。
- [Java 開發套件 \(JDK\) 第 11 版](#) - 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，您可以使用 [Amazon Corretto](#) 或我們選擇的任何標準 JDK。
- 若要確認您已正確安裝 JDK，請執行下列命令。如果您使用的是 Amazon Corretto 11 以外的 JDK，輸出將會不同。請確定版本為 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](#) - 如果您尚未安裝 Apache Maven。如需詳細資訊，請參閱[安裝 Apache Maven](#)。
- 若要測試您的 Apache Maven 安裝，請使用下列命令：

```
$ mvn -version
```

Note

雖然您的應用程式是以 Python 撰寫，但 Apache Flink 在 Java 虛擬機器 (JVM) 中執行。它將 Kinesis 連接器等大多數相依性分發為 JAR 檔案。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](#)。本教學課程說明如何執行此操作。

Warning

我們建議您使用 Python 3.11 進行本機開發。這是 Amazon Managed Service for Apache Flink 搭配 Flink 執行期 1.19 使用的相同 Python 版本。

在 Python 3.12 上安裝 Python Flink 程式庫 1.19 可能會失敗。

如果您的機器預設安裝了另一個 Python 版本，我們建議您使用 Python 3.11 建立 VirtualEnv 等獨立環境。

適用於本機開發的 IDE

我們建議您使用 [PyCharm](#) 或 [Visual Studio Code](#) 等開發環境來開發和編譯您的應用程式。

然後，完成的前兩個步驟[開始使用 Amazon Managed Service for Apache Flink \(DataStream API\)](#)：

- [設定 AWS 帳戶並建立管理員使用者](#)
- [設定 AWS Command Line Interface \(AWS CLI\)](#)

若要開始使用，請參閱[建立應用程式](#)。

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

在本節中，您會建立適用於 Python 應用程式的 Managed Service for Apache Flink 應用程式，並以 Kinesis 串流做為來源和接收器。

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 Apache Flink 串流 Python 程式碼](#)
- [管理 JAR 相依性](#)
- [將範例記錄寫入輸入串流](#)
- [在本機執行您的應用程式](#)
- [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](#)
- [封裝您的應用程式程式碼](#)

- [將應用程式套件上傳至 Amazon S3 儲存貯體](#)
- [建立和設定 Managed Service for Apache Flink 應用程式](#)
- [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。
- 存放應用程式程式碼的 Amazon S3 儲存貯體。

Note

本教學假設您正在 us-east-1 區域中部署應用程式。如果您使用另一個區域，則必須相應地調整所有步驟。

建立兩個 Kinesis 串流

為此練習建立 Managed Service for Apache Flink 應用程式之前，請在將用於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 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 儲存貯體提供全域唯一名稱，例如透過附加您的登入名稱。

Note

請確定您在用於本教學課程的區域中建立 S3 儲存貯體 (us-east-1)。

其他資源

建立應用程式時，Managed Service for Apache Flink 會建立下列 Amazon CloudWatch 資源 (如果尚不存在該資源)：

- 名為 /AWS/KinesisAnalytics-java/<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

對於開發和偵錯，您可以在機器上執行 Python Flink 應用程式。您可以在您選擇的 Python IDE 中使用 `python main.py` 或從命令列啟動應用程式。

Note

在您的開發機器上，您必須安裝 Python 3.10 或 3.11、Java 11、Apache Maven 和 Git。我們建議您使用 IDE，例如 [PyCharm](#) 或 [Visual Studio Code](#)。若要驗證您是否符合所有先決條件，請先參閱 [滿足完成練習的先決條件](#) 再繼續。

安裝 PyFlink 程式庫

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、Conda 或任何類似的 Python 工具建立獨立的 Python 環境。
2. 在該環境中安裝 PyFlink 程式庫。使用您在 Amazon Managed Service for Apache Flink 中使用的相同 Apache Flink 執行時間版本。目前，建議的執行時間為 1.19.1。

```
$ pip install apache-flink==1.19.1
```

3. 執行應用程式時，請確定環境處於作用中狀態。如果您在 IDE 中執行應用程式，請確定 IDE 使用環境做為執行時間。程序取決於您使用的 IDE。

Note

您只需要安裝 PyFlink 程式庫。您不需要在機器上安裝 Apache Flink 叢集。

驗證您的 AWS 工作階段

應用程式使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有有效的已 AWS 驗證工作階段，具有寫入 Kinesis 資料串流的許可。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 和具有有效登入資料的具名設定檔，請參閱 [設定 AWS Command Line Interface \(AWS CLI\)](#)。
2. 透過發佈下列測試記錄，確認您的 AWS CLI 已正確設定，且您的使用者具有寫入 Kinesis 資料串流的許可：

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. 如果您的 IDE 有要整合的外掛程式 AWS，您可以使用它將登入資料傳遞至在 IDE 中執行的應用程式。如需詳細資訊，請參閱 [AWS Toolkit for PyCharm](#)、[AWS Toolkit for Visual Studio Code](#) 和 [AWS Toolkit for IntelliJ IDEA](#)。

下載並檢查 Apache Flink 串流 Python 程式碼

此範例的 Python 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，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。

檢閱應用程式元件

應用程式碼位於 `main.py`。我們使用內嵌在 Python 中的 SQL 來定義應用程式的流程。

Note

為了獲得最佳化的開發人員體驗，應用程式的設計可在 Amazon Managed Service for Apache Flink 和本機上執行，無需變更任何程式碼，即可在機器上進行開發。應用程式使用環境變數 `IS_LOCAL = true` 來偵測它何時在本機執行。您必須在 shell `IS_LOCAL = true` 或 IDE 的執行組態中設定環境變數。

- 應用程式會設定執行環境並讀取執行時間組態。若要同時在 Amazon Managed Service for Apache Flink 和本機上運作，應用程式會檢查 `IS_LOCAL` 變數。
- 以下是應用程式在 Amazon Managed Service for Apache Flink 中執行時的預設行為：
 1. 載入與應用程式一起封裝的相依性。如需詳細資訊，請參閱（[連結](#)）
 2. 從您在 Amazon Managed Service for Apache Flink 應用程式中定義的執行期屬性載入組態。如需詳細資訊，請參閱（[連結](#)）
- 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 `application_properties.json` 的檔案載入組態。

```
...
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",
)
```

- 應用程式使用 [Kinesis Connector](#) 定義具有CREATE TABLE陳述式的來源資料表。此資料表會從輸入Kinesis 串流讀取資料。應用程式會從執行時間組態中取得串流的名稱、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 Connector](#) 定義接收器資料表。此故事會將資料傳送至輸出Kinesis 串流。

```
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執行INSERT INTO...接收資料表的 SQL。在更複雜的應用程式中，您可能會有額外的步驟在寫入目的地之前轉換資料。

```
table_result = table_env.execute_sql("""INSERT INTO output
    SELECT ticker, price, event_time FROM prices""")
```

- 您必須在main()函數結尾新增另一個步驟，才能在本機執行應用程式：

```
if is_local:
    table_result.wait()
```

如果沒有此陳述式，應用程式會在您於本機執行時立即終止。當您在 Amazon Managed Service for Apache Flink 中執行應用程式時，不得執行此陳述式。

管理 JAR 相依性

PyFlink 應用程式通常需要一或多個連接器。本教學課程中的應用程式使用 [Kinesis Connector](#)。由於 Apache Flink 在 Java JVM 中執行，無論您是否在 Python 中實作應用程式，連接器都會以 JAR 檔案形式分佈。在 Amazon Managed Service for Apache Flink 上部署這些相依性時，您必須將這些相依性與應用程式一起封裝。

在此範例中，我們會示範如何使用 Apache Maven 來擷取相依性，並封裝應用程式以在 Managed Service for Apache Flink 上執行。

Note

有擷取和封裝相依性的其他方法。此範例示範可正確搭配一或多個連接器使用的方法。它還可讓您在本地、用於開發以及在 Managed Service for Apache Flink 上執行應用程式，而無需變更程式碼。

使用 pom.xml 檔案

Apache Maven 使用 pom.xml 檔案來控制相依性和應用程式封裝。

任何 JAR 相依性都會在 <dependencies>...</dependencies>區塊的 pom.xml 檔案中指定。

```
<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連接器成品和版本，請參閱 [將 Apache Flink 連接器與 Managed Service for Apache Flink 搭配使用](#)。請務必參考您正在使用的 Apache Flink 版本。在此範例中，我們使用 Kinesis 連接器。對於 Apache 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，則沒有為此版本特別發行的連接器版本。使用針對 1.18 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，並為 Python Flink 應用程式封裝相依性。

1. 導覽至包含 Python 入門專案的目錄，稱為 python/GettingStarted。
2. 執行以下命令：

```
$ mvn package
```

Maven 會建立新的檔案，名為 ./target/pyflink-dependencies.jar。當您在機器上進行本機開發時，Python 應用程式會尋找此檔案。

Note

如果您忘記執行此命令，當您嘗試執行應用程式時，它會失敗並顯示錯誤：找不到任何識別符為 "kinesis" 的工廠。

將範例記錄寫入輸入串流

在本節中，您將傳送範例記錄到串流，供應用程式處理。您可以使用 Python 指令碼或 [Kinesis Data Generator](#) 產生範例資料。

使用 Python 指令碼產生範例資料

您可以使用 Python 指令碼將範例記錄傳送至串流。

Note

若要執行此 Python 指令碼，您必須使用 Python 3.x 並安裝 [AWS 適用於 Python \(Boto\) 的 SDK](#) 程式庫。

若要開始將測試資料傳送至 Kinesis 輸入串流：

1. 從資料產生器 [GitHub 儲存庫下載資料產生器](#) stock.py Python 指令碼。
2. 執行 stock.py 指令碼：

```
$ python stock.py
```

在您完成教學課程的其餘部分時，請讓指令碼保持執行。您現在可以執行 Apache Flink 應用程式。

使用 Kinesis Data Generator 產生範例資料

或者，若要使用 Python 指令碼，您可以使用 [託管版本](#) 中也提供的 [Kinesis Data Generator](#)，將隨機範例資料傳送至串流。Kinesis Data Generator 會在您的瀏覽器中執行，您不需要在機器上安裝任何項目。

若要設定和執行 Kinesis Data Generator：

1. 遵循 [Kinesis Data Generator 文件](#) 中的指示來設定對工具的存取。您將執行設定使用者和密碼的 CloudFormation 範本。
2. 透過 CloudFormation 範本產生的 URL 存取 Kinesis Data Generator。CloudFormation 範本完成後，您可以在輸出索引標籤中找到 URL。
3. 設定資料產生器：
 - 區域：選取您用於本教學課程的區域：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，並確認產生的記錄與以下內容類似：

```
{ "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 Data Generator 現在正在將資料傳送至 ExampleInputStream。

在本機執行您的應用程式

您可以在本機測試應用程式，使用 `python main.py` 從命令列執行或從 IDE 執行。

若要在本機執行應用程式，您必須安裝正確的 PyFlink 程式庫版本，如上一節所述。如需詳細資訊，請參閱（[連結](#)）

Note

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](#)。此外，請確認您具有從兩個串流讀取和寫入的許可。請參閱 [驗證您的 AWS 工作階段](#)。

將 Python 專案匯入 IDE

若要開始在 IDE 中使用應用程式，您必須將其匯入為 Python 專案。

您複製的儲存庫包含多個範例。每個範例都是個別的專案。在本教學課程中，將 `./python/GettingStarted` 子目錄中的內容匯入 IDE。

將程式碼匯入為現有的 Python 專案。

Note

匯入新 Python 專案的確切程序取決於您使用的 IDE。

檢查本機應用程式組態

在本機執行時，應用程式會使用下專案資源資料夾中 `application_properties.json` 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"
    }
  }
]
```

在本機執行您的 Python 應用程式

您可以從命令列做為一般 Python 指令碼，或從 IDE 在本機執行應用程式。

從命令列執行您的應用程式

1. 請確定安裝 Python Flink 程式庫的獨立 Python 環境目前處於作用中狀態，例如 Conda 或 VirtualEnv。
2. 請確定您 `mvn package` 至少執行一次。
3. 設定 `IS_LOCAL = true` 環境變數：

```
$ export IS_LOCAL=true
```

4. 以一般 Python 指令碼執行應用程式。

```
$python main.py
```

從 IDE 中執行應用程式

1. 將 IDE 設定為使用以下組態執行 `main.py` 指令碼：
 1. 使用安裝 PyFlink 程式庫的獨立 Python 環境，例如 Conda 或 VirtualEnv。
 2. 使用 AWS 登入資料來存取輸入和輸出 Kinesis 資料串流。
 3. 設定 `IS_LOCAL = true`。
2. 設定執行組態的確切程序取決於您的 IDE 和。
3. 當您設定 IDE 後，請執行 Python 指令碼，並在應用程式執行時使用 IDE 提供的工具。

在本機檢查應用程式日誌

在本機執行時，除了應用程式啟動時列印和顯示的幾行以外，應用程式不會在主控台中顯示任何日誌。PyFlink 會將日誌寫入安裝 Python Flink 程式庫的目錄中的檔案。應用程式會在日誌啟動時列印日誌的位置。您也可以執行下列命令來尋找日誌：

```
$ 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。您通常會找到單一 `.log` 檔案。
2. 在應用程式執行時自訂檔案：`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 主控台中的資料檢視器，觀察（產生範例 Python）或 Kinesis Data Generator（連結）傳送至輸入串流的記錄。Amazon Kinesis

若要觀察記錄：

停止應用程式在本機執行

停止在 IDE 中執行的應用程式。IDE 通常提供「停止」選項。確切的位置和方法取決於 IDE。

封裝您的應用程式程式碼

在本節中，您會使用 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。

再次執行 Maven 套件命令：

```
$ mvn package
```

此命令會產生檔案 `target/managed-flink-pyflink-getting-started-1.0.0.zip`。

將應用程式套件上傳至 Amazon S3 儲存貯體

在本節中，您將上一節中建立的 .zip 檔案上傳至在本教學課程開始時建立的 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如果您尚未完成此步驟，請參閱（連結）。

上傳應用程式碼 JAR 檔案

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選擇您先前為應用程式碼建立的儲存貯體。
3. 選擇上傳。
4. 選擇 Add files (新增檔案)。
5. 導覽至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。

建立和設定 Managed Service for Apache Flink 應用程式

您可以使用 主控台 或 建立和設定 Managed Service for Apache Flink 應用程式 AWS CLI。在本教學課程中，我們將使用 主控台。

建立應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
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. 開啟右側選單，然後選擇 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。
 - 針對 Apache Flink 組態、Application Flink 版本，選擇 Apache Flink 1.19。
 - 對於應用程式組態：
 - 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，選擇使用必要政策建立/更新 IAM 角色 `kinesis-analytics-MyApplication-us-east-1`。
 - 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

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`

Amazon Managed Service for Apache Flink 先前稱為 Kinesis Data Analytics。自動產生的資源名稱會加上字首，kinesis-analytics以確保回溯相容性。

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 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一節所建立的 **kinesis-analytics-service-MyApplication-us-east-1** 政策。
3. 選擇編輯，然後選擇 JSON 索引標籤。
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**)。

JSON

```
{
  "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成品。

設定應用程式

1. 在 MyApplication 頁面，選擇設定。
2. 在應用程式碼位置區段中：
 - 針對 Amazon S3 儲存貯體，選取您先前為應用程式碼建立的儲存貯體。選擇瀏覽並選擇正確的儲存貯體，然後選擇選擇。請勿選取儲存貯體名稱。
 - 對於 Amazon S3 物件的路徑，請輸入 **managed-flink-pyflink-getting-started-1.0.0.zip**。
3. 針對存取許可，選擇 **kinesis-analytics-MyApplication-us-east-1** 使用必要政策建立/更新 IAM 角色。
4. 移至執行期屬性，並保留所有其他設定的預設值。
5. 選擇新增項目並新增下列每個參數：

群組 ID	金鑰	值
InputStream0	stream.name	ExampleInputStream
InputStream0	flink.stream.initpos	LATEST
InputStream0	aws.region	us-east-1
OutputStream0	stream.name	ExampleOutputStream
OutputStream0	aws.region	us-east-1
kinesis.analytics.flink.run.options	python	main.py
kinesis.analytics.flink.run.options	jarfile	lib/pyflink-dependencies.jar

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 CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

執行應用程式

應用程式現在已設定並準備好執行。

執行應用程式

1. 在 Amazon Managed Service for Apache Flink 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，選擇使用最新的快照執行，然後選擇執行。

應用程式詳細資訊中的狀態會從 Ready 轉換為 Starting，然後在應用程式啟動Running時轉換為。

當應用程式處於 Running 狀態時，您現在可以開啟 Flink 儀表板。

開啟 儀表板

1. 選擇開啟 Apache Flink 儀表板。儀表板會在新頁面上開啟。
2. 在執行中任務清單中，選擇您可以看到的單一任務。

Note

如果您未正確設定執行期屬性或編輯 IAM 政策，應用程式狀態可能會變成 Running，但 Flink 儀表板會顯示任務正在持續重新啟動。如果應用程式設定錯誤或缺少存取外部資源的許可，這是常見的失敗案例。

發生這種情況時，請檢查 Flink 儀表板中的例外狀況索引標籤，以查看問題的原因。

觀察執行中應用程式的指標

在 MyApplication 頁面的 Amazon CloudWatch 指標區段中，您可以查看執行中應用程式的一些基本指標。

檢視指標

1. 在重新整理按鈕旁，從下拉式清單中選取 10 秒。
2. 當應用程式執行正常時，您可以看到運作時間指標持續增加。
3. Fullrestarts 指標應為零。如果增加，組態可能會發生問題。若要調查問題，請檢閱 Flink 儀表板上的例外狀況索引標籤。
4. 在運作狀態良好的應用程式中，失敗檢查點指標的數量應為零。

Note

此儀表板會顯示一組固定的指標，精細程度為 5 分鐘。您可以使用 CloudWatch 儀表板中的任何指標來建立自訂應用程式儀表板。

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 Data Generator 將資料發佈至輸入。

您現在可以使用 <https://console.aws.amazon.com/kinesis/> 中的資料檢視器來觀察 Managed Service for Apache Flink 上執行之應用程式的輸出，與先前已執行的操作類似。

檢視輸出

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis/>。
2. 確認區域與您用來執行本教學課程的區域相同。根據預設，它是 us-east-1US East (N. Virginia)。視需要變更 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，保持最新為開始位置，然後選擇取得記錄。您可能會看到「找不到此請求的記錄」錯誤。若是如此，請選擇重試取得記錄。發佈至串流顯示的最新記錄。
7. 選取資料欄中的值，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，請前往名為 的 Managed Service for Apache Flink 應用程式的主控台頁面MyApplication。

停止應用程式

1. 從動作下拉式清單中，選擇停止。
2. 應用程式詳細資訊中的狀態會從 轉換為 Running Stopping，然後在應用程式完全停止Ready時轉換為 。

Note

別忘了也要停止從 Python 指令碼或 Kinesis Data Generator 將資料傳送至輸入串流。

下一步驟

[清除 AWS 資源](#)

清除 AWS 資源

本節包含在入門 (Python) 教學課程中建立 AWS 的資源清除程序。

本主題包含下列章節。

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

請使用下列程序刪除應用程式。

刪除應用程式

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。

2. 在 Managed Service for Apache Flink 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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 選擇資料串流。
3. 選取您建立的兩個串流，ExampleInputStream以及 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 物件和儲存貯體

使用下列程序來刪除 S3 物件和儲存貯體。

從 S3 儲存貯體刪除物件

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
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成品，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. 開啟位於 <https://console.aws.amazon.com/s3/> 的 Amazon S3 主控台。
2. 選取您為成品建立的儲存貯體。
3. 選擇刪除並確認刪除。

Note

S3 儲存貯體必須為空白，才能將其刪除。

刪除您的 IAM 資源

使用下列程序刪除 IAM 資源。

刪除 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-east-1 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-east-1 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

使用下列程序刪除 CloudWatch 資源。

刪除 CloudWatch 資源

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

開始使用 (Scala)

Note

從 1.15 版開始，Flink 不含 Scala。應用程式現在可以使用任何 Scala 版本的 Java API。Flink 仍會在內部幾個關鍵元件中使用 Scala，但不會向使用者程式碼類別載入器公開 Scala。因此，您必須將 Scala 相依性新增至 JAR 封存檔。

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，您會使用 Kinesis 串流做為來源和接收器，為 Scala 建立 Managed Service for Apache Flink 應用程式。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [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](#)
- [建立並執行應用程式（主控台）](#)
- [建立並執行應用程式 \(CLI\)](#)
- [清除 AWS 資源](#)

建立相依資源

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 create-stream AWS CLI 命令。

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

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

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

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

其他資源

建立應用程式時，Managed Service for Apache Flink 會建立下列 Amazon CloudWatch 資源 (如果尚不存在該資源)：

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

將範例記錄寫入輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

Note

本節需要 [適用於 Python \(Boto\) 的 AWS SDK](#)。

 Note

本節中的 Python 指令碼會使用 AWS CLI。您必須 AWS CLI 將設定為使用您的帳戶登入資料和預設區域。若要設定您的 AWS CLI，請輸入下列項目：

```
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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行期應用程式的屬性，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](#) 建置工具來建置應用程式的 Scala 程式碼。若要安裝 SBT，請參閱[使用 cs 安裝程式安裝 sbt](#)。您還需要安裝 Java 開發套件 (JDK)。請參閱[完成練習的先決條件](#)。

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用 SBT 編譯和封裝程式碼：

```
sbt assembly
```

2. 如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Scala 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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 `getting-started-scala-1.0.jar` 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

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

建立並執行應用程式（主控台）

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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 1.19.1 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

請使用下列程序設定應用程式。

設定應用程式

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

群組 ID	金鑰	值
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

- 在監控下，確保監控指標層級設為應用程式。
- 針對 CloudWatch 記錄，選取啟用核取方塊。
- 選擇更新。

Note

當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 S3 儲存貯體許可

- 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。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**)。

JSON

```
{
```

```

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

```

```
    ],
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}
```

執行應用程式

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

停止應用程式

若要停止應用程式，請在 MyApplication 頁面上選擇停止。確認動作。

建立並執行應用程式 (CLI)

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

建立許可政策

Note

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

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 read 動作的許可，而另一條則是授與目的地串流上 write 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

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

JSON

```
{
  "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-2:123456789012:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
```

```

        "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ],
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": [
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:123456789012:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:123456789012:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:123456789012:stream/
ExampleOutputStream"
    }
]
}

```

如需建立許可政策的逐步指示，請參閱《IAM 使用者指南》中的[教學課程：建立和連接您的第一個客戶管理政策](#)。

建立 IAM 政策

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

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。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。

如需建立角色的逐步指示，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": "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"  
  }  
]  
}
```

使用下列請求執行 [CreateApplication](#) 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": "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 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用 [StopApplication](#) 動作來停止應用程式。

停止應用程式

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

```
{
  "ApplicationName": "s3_sink"
}
```

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

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

現在已停止應用程式。

新增 CloudWatch 記錄選項

您可以使用 AWS CLI 將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch Logs 的相關資訊，請參閱[設定應用程式記錄](#)。

更新環境屬性

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

更新應用程式的環境屬性

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

```
{
  "ApplicationName": "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

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 [UpdateApplication](#) CLI 動作。

Note

若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"  
    }  
  }  
}  
}
```

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

將 Apache Beam 與 Managed Service for Apache Flink 應用程式搭配使用

Note

沒有適用於 Flink 1.20 的相容 Apache Flink Runner。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[使用 CloudFormation](#)。

本主題包含下列章節：

- [使用 Managed Service for Apache Flink 的 Apache Flink Runner 限制](#)
- [使用 Managed Service for Apache Flink 的 Apache Beam 功能](#)
- [使用 Apache Beam 建立應用程式](#)

使用 Managed Service for Apache Flink 的 Apache Flink Runner 限制

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。

使用 Managed Service for Apache Flink 的 Apache Beam 功能

Managed Service for Apache Flink 與 Apache Flink 執行器支援相同的 Apache Beam 功能。如需 Apache Flink 執行器所支援功能的相關資訊，請參閱 [Beam 相容性矩陣](#)。

建議您在 Managed Service for Apache Flink 服務中測試 Apache Flink 應用程式，以確認我們是否支援您的應用程式所需的全部功能。

使用 Apache Beam 建立應用程式

在本練習中，您將使用 [Apache Beam](#) 建立可轉換資料的 Managed Service for Apache Flink 應用程式。Apache Beam 是用於處理串流資料的程式設計模型。如需將 Apache Beam 與 Managed Service for Apache Flink 搭配使用的相關資訊，請參閱[將 Apache Beam 與 Managed Service for Apache Flink 應用程式搭配使用](#)。

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 Managed Service for Apache Flink 中的 DataStream API 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將範例記錄寫入輸入串流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳 Apache Flink 串流 Java 程式碼](#)
- [建立並執行 Managed Service for Apache Flink 應用程式](#)
- [清除 AWS 資源](#)
- [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。

將範例記錄寫入輸入串流

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 [適用於 Python \(Boto\) 的 AWS SDK](#)。

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 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先安裝。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：

- 該應用程式使用 Apache Beam [ParDo](#)，透過調用稱為 `PingPongFn` 的自定義轉換函數來處理傳入的記錄。

調用 `PingPongFn` 函數的代碼如下：

```
.apply("Pong transform",  
    ParDo.of(new PingPongFn()))
```

- 使用 Apache Beam 的 Managed Service for Apache Flink 應用程式需要下列元件。如果您未在 `pom.xml` 中包含這些元件和版本，應用程式會從環境相依性載入不正確的版本，而且由於版本不符合，應用程式會在執行期損毀。

```
<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\n` 到輸出串流。

轉換函數的程式碼如下：

```
private static class PingPongFn extends DoFn<KinesisRecord, byte[]> {
    private static final Logger LOG = LoggerFactory.getLogger(PingPongFn.class);

    @ProcessElement
    public void processElement(ProcessContext c) {
        String content = new String(c.element().getDataAsBytes(),
            StandardCharsets.UTF_8);
        if (content.trim().equalsIgnoreCase("ping")) {
            LOG.info("Ponged!");
            c.output("pong\n".getBytes(StandardCharsets.UTF_8));
        } else {
            LOG.info("No action for: " + content);
            c.output(c.element().getDataAsBytes());
        }
    }
}
```

編譯應用程式程式碼

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 Managed Service for Apache Flink 中的 DataStream API](#)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 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在[建立相依資源](#)一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-**<username>** 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 basic-beam-app-1.0.jar 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

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

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

依照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. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 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

Apache Beam 目前與 Apache Flink 1.19 版或更新版本不相容。

- 從版本下拉式清單中選取 Apache Flink 1.15 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 : `kinesis-analytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 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`)。

JSON

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

```

        "Effect": "Allow",
        "Action": "logs:PutLogEvents",
        "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
        "Sid": "ListCloudwatchLogGroups",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogGroups"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:*"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

設定應用程式

1. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
 - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
 - 對於 Amazon S3 物件的路徑，請輸入 **basic-beam-app-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。

4. 輸入下列資料：

群組 ID	金鑰	值
BeamApplicationProperties	InputStreamName	ExampleInputStream
BeamApplicationProperties	OutputStreamName	ExampleOutputStream
BeamApplicationProperties	AwsRegion	us-west-2

5. 在監控下，確保監控指標層級設為應用程式。
6. 針對 CloudWatch 記錄，選取啟用核取方塊。
7. 選擇更新。

Note

當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：

- 日誌群組：/aws/kinesis-analytics/MyApplication
- 日誌串流：kinesis-analytics-log-stream

此日誌串流用於監控應用程式。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 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包含清除在輪轉時段教學課程中建立 AWS 之資源的程序。

本主題包含下列章節：

- [刪除 Managed Service for Apache Flink 應用程式](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 物件和儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除 Managed Service for Apache Flink 應用程式

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在 Managed Service for Apache Flink 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，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除您的 Amazon S3 物件和儲存貯體

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

刪除您的 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 政策。

5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

現在您已建立並執行使用 Apache Beam 轉換資料的基本 Managed Service for Apache Flink 應用程式，請參閱下列應用程式，取得更進階 Managed Service for Apache Flink 解決方案的範例。

- [Beam 用於 Managed Service for Apache Flink 串流研討會](#)：在此研討會中，我們將探索一個端對端範例，將批次和串流方面結合在一個統一的 Apache Beam 管道中。

訓練研討會、實驗室和解決方案實作

下列端對端範例示範進階 Managed Service for Apache Flink 解決方案。

主題

- [使用 Amazon Managed Service for Apache Flink 部署、操作和擴展應用程式](#)
- [在本機開發 Apache Flink 應用程式，然後再部署到 Managed Service for Apache Flink](#)
- [搭配 Managed Service for Apache Flink Studio 使用事件偵測](#)
- [使用 AWS Amazon Kinesis 的串流資料解決方案](#)
- [練習搭配 Apache Flink 和 Apache Kafka 使用 Clickstream 實驗室](#)
- [使用 Application Auto Scaling 設定自訂擴展](#)
- [檢視範例 Amazon CloudWatch 儀表板](#)
- [使用 範本串流 AWS Amazon MSK 的資料解決方案](#)
- [在 GitHub 上探索更多 Managed Service for Apache Flink 解決方案](#)

使用 Amazon Managed Service for Apache Flink 部署、操作和擴展應用程式

本研討會涵蓋 Java 中的 Apache Flink 應用程式開發、如何在 IDE 中執行和偵錯，以及如何在 Amazon Managed Service for Apache Flink 上封裝、部署和執行。您也將了解如何擴展、監控和疑難排解您的應用程式。

[Amazon Managed Service for Apache Flink 研討會](#)。

在本機開發 Apache Flink 應用程式，然後再部署到 Managed Service for Apache Flink

此研討會示範在本機啟動並開始開發 Apache Flink 應用程式的基礎知識，其長期目標是部署至 Managed Service for Apache Flink for Apache Flink。

[使用 Apache Flink 進行本機開發的入門指南](#)

搭配 Managed Service for Apache Flink Studio 使用事件偵測

本研討會介紹如何使用 Managed Service for Apache Flink Studio 進行事件偵測，並將其部署為 Managed Service for Apache Flink 應用程式

[使用 Managed Service for Apache Flink for Apache Flink 進行事件偵測](#)

使用 AWS Amazon Kinesis 的串流資料解決方案

Amazon Kinesis AWS 的串流資料解決方案會自動設定擷取、儲存、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。Managed Service for Apache Flink 選項提供端對端串流 ETL 範例，展示真實世界的應用程式如何根據模擬的紐約計程車資料執行分析作業。

每個解決方案 包含下列元件：

- 部署完整範例的 CloudFormation 套件。
- 用於顯示應用程式指標的 CloudWatch 儀表板。
- CloudWatch 會對最相關的應用程式指標發出警示。
- 所有必要的 IAM 角色和政策。

[Amazon Kinesis 的串流資料解決方案](#)

練習搭配 Apache Flink 和 Apache Kafka 使用 Clickstream 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。

[Clickstream 實驗室](#)

使用 Application Auto Scaling 設定自訂擴展

兩個範例說明如何使用 Application Auto Scaling 自動擴展 Managed Service for Apache Flink 應用程式。這可讓您設定自訂擴展政策和自訂擴展屬性。

- [Managed Service for Apache Flink 應用程式自動擴展](#)
- [排程擴展](#)

如需可執行自訂擴展的詳細資訊，請參閱[啟用 Amazon Managed Service for Apache Flink 的指標型和排程擴展](#)。

檢視範例 Amazon CloudWatch 儀表板

用於監控 Managed Service for Apache Flink 應用程式的 CloudWatch 範例儀表板。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。

[Managed Service for Apache Flink 指標儀表板](#)

使用 範本串流 AWS Amazon MSK 的資料解決方案

適用於 AWS Amazon MSK 的串流資料解決方案提供 CloudFormation 範本，其中資料會流經生產者、串流儲存體、消費者和目的地。

[AWS Amazon MSK 的串流資料解決方案](#)

在 GitHub 上探索更多 Managed Service for Apache Flink 解決方案

下列端對端範例示範進階 Managed Service for Apache Flink 解決方案，並可在 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](#)
- [基於 Apache Flink 和 Amazon Managed Service for Apache Flink 的串流 ETL](#)
- [對客戶意見反饋進行即時情緒分析](#)

使用 Managed Service for Apache Flink 的實用公用程式

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

最佳實務是 Flink 應用程式定期啟動儲存點/快照，以允許更無縫的故障復原。快照管理員可自動執行此任務並提供下列優點：

- 建立執行中 Managed Service for Apache Flink 應用程式的新快照
- 取得應用程式快照的計數
- 檢查計數是否超過所需的快照數
- 刪除早於所需數目的舊快照

如需範例，請參閱[快照管理員](#)。

基準測試

Managed Service for Apache Flink 基準測試公用程式可協助進行 Managed Service for Apache Flink 應用程式的容量規劃、整合測試和基準測試。

有關示例，請參閱[基準測試](#)

建立和使用 Managed Service for Apache Flink 應用程式的範例

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逐步指示，可協助您建立 Managed Service for Apache Flink 應用程式並測試結果。

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 Managed Service for Apache Flink 中的 DataStream API](#)

Note

這些範例假設您使用美國東部（維吉尼亞北部）區域 (us-east-1)。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

- [Managed Service for Apache Flink 的 Java 範例](#)
- [Managed Service for Apache Flink 的 Python 範例](#)
- [Managed Service for Apache Flink 的 Scala 範例](#)

Managed Service for Apache Flink 的 Java 範例

下列範例示範如何建立以 Java 撰寫的應用程式。

Note

大多數範例都是為了在本機、您的開發機器和您選擇的 IDE 以及 Amazon Managed Service for Apache Flink 上執行而設計。它們示範了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，以在兩個環境中執行應用程式，而不會進行任何變更。

改善序列化效能，定義自訂 TypeInfo

此範例說明如何在您的記錄或狀態物件上定義自訂 TypeInfo，以防止序列化回到效率較低的 Kryo 序列化。例如，當您的物件包含 List 或 Map 時，這是必要的。如需詳細資訊，請參閱 Apache Flink 文件中的[資料類型和序列化](#)。此範例也會示範如何測試物件的序列化是否回到效率較低的 Kryo 序列化。

程式碼範例：[CustomTypeInfo](#)

DataStream API 入門

此範例顯示一個簡單的應用程式，使用 DataStream API 從 Kinesis 資料串流讀取和寫入另一個 Kinesis 資料串流。此範例示範如何使用正確的相依性設定 檔案、建置 uber-JAR，然後剖析組態參數，讓您可以在本機、IDE 和 Amazon Managed Service for Apache Flink 上執行應用程式。

程式碼範例：[GettingStarted](#)

開始使用資料表 API 和 SQL

此範例顯示使用 Table API 和 SQL 的簡單應用程式。它示範如何將 DataStream API 與相同 Java 應用程式中的 Table API 或 SQL 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。

完整範例：[GettingStartedTable](#)

使用 S3Sink (DataStream API)

此範例示範如何使用 DataStream API 的 `FileSink` 將 JSON 檔案寫入 S3 儲存貯體。

程式碼範例：[S3Sink](#)

使用 Kinesis 來源、標準或 EFO 取用者和接收器 (DataStream API)

此範例示範如何使用標準取用者或 EFO 從 Kinesis 資料串流設定來源取用，以及如何將接收器設定為 Kinesis 資料串流。

程式碼範例：[KinesisConnectors](#)

使用 Amazon Data Firehose 接收器 (DataStream API)

此範例示範如何將資料傳送至 Amazon Data Firehose（先前稱為 Kinesis Data Firehose）。

程式碼範例：[KinesisFirehoseSink](#)

使用 Prometheus 接收器連接器

此範例示範如何使用 [Prometheus 接收器連接器](#) 將時間序列資料寫入 Prometheus。

程式碼範例：[PrometheusSink](#)

使用視窗彙總 (DataStream API)

此範例示範 DataStream API 中的四種視窗化彙總類型。

1. 根據處理時間的滑動視窗
2. 根據事件時間的滑動視窗
3. 根據處理時間的輪轉時段
4. 根據事件時間的輪轉時段

程式碼範例：[視窗調整](#)

使用自訂指標

此範例說明如何將自訂指標新增至 Flink 應用程式，並將其傳送至 CloudWatch 指標。

程式碼範例：[CustomMetrics](#)

使用 Kafka 組態提供者在執行時間擷取 mTLS 的自訂金鑰存放區和信任存放區

此範例說明如何使用 Kafka 組態提供者，為 Kafka 連接器設定具有 mTLS 身分驗證憑證的自訂金鑰存放區和信任存放區。此技術可讓您從 Amazon S3 載入必要的自訂憑證，並從應用程式啟動 AWS Secrets Manager 時載入秘密。

程式碼範例：[Kafka-mTLS-Keystore-ConfigProviders](#)

使用 Kafka 組態提供者在執行時間擷取 SASL/SCRAM 身分驗證的秘密

此範例說明如何使用 Kafka 組態提供者從 Amazon S3 擷取憑證 AWS Secrets Manager，並從 Amazon S3 下載信任存放區，以在 Kafka 連接器上設定 SASL/SCRAM 身分驗證。Amazon S3 此技術可讓您從 Amazon S3 載入必要的自訂憑證，並從應用程式啟動 AWS Secrets Manager 時載入秘密。

程式碼範例：[Kafka-SASL_SSL-ConfigProviders](#)

使用 Kafka 組態提供者，在執行時間透過資料表 API/SQL 擷取 mTLS 的自訂金鑰存放區和信任存放區

此範例說明如何使用資料表 API /SQL 中的 Kafka 組態提供者，為 Kafka 連接器設定具有 mTLS 身分驗證憑證的自訂金鑰存放區和信任存放區。此技術可讓您從 Amazon S3 載入必要的自訂憑證，並從應用程式啟動 AWS Secrets Manager 時載入秘密。

程式碼範例：[Kafka-mTLS-Keystore-Sql-ConfigProviders](#)

使用側邊輸出分割串流

此範例說明如何利用 Apache Flink 中的 [Side Outputs](#) 在指定的屬性上分割串流。此模式在嘗試在串流應用程式中實作無效字母佇列 (DLQ) 的概念時特別有用。

程式碼範例：[SideOutputs](#)

使用非同步 I/O 呼叫外部端點

此範例說明如何使用 [Apache Flink 非同步 I/O](#) 以非封鎖方式呼叫外部端點，並重試可復原的錯誤。

程式碼範例：[AsyncIO](#)

Managed Service for Apache Flink 的 Python 範例

下列範例示範如何建立以 Python 撰寫的應用程式。

Note

大多數範例都是為了在本機、您的開發機器和您選擇的 IDE 以及 Amazon Managed Service for Apache Flink 上執行而設計。它們示範了您可以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在兩個環境中執行應用程式，而不會變更。

專案相依性

大多數 PyFlink 範例需要一或多個做為 JAR 檔案的相依性，例如 Flink 連接器。在 Amazon Managed Service for Apache Flink 上部署時，這些相依性必須與應用程式一起封裝。

下列範例已包含工具，可讓您在本機執行應用程式以進行開發和測試，並正確封裝所需的相依性。此工具需要使用 Java JDK11 和 Apache Maven。如需特定指示，請參閱每個範例中包含的 README。

範例

開始使用 PyFlink

此範例示範使用內嵌在 Python 程式碼中的 SQL 的 PyFlink 應用程式的基本結構。此專案也為包含連接器等 JAR 相依性的任何 PyFlink 應用程式提供骨架。README 區段提供如何在本機執行 Python 應用程式以進行開發的詳細指引。此範例也說明如何在 PyFlink 應用程式中包含單一 JAR 相依性、此範例中的 Kinesis SQL 連接器。

程式碼範例：[GettingStarted](#)

新增 Python 相依性

此範例示範如何以最一般的方式將 Python 相依性新增至 PyFlink 應用程式。此方法適用於簡單的相依性，例如 Boto3，或包含 PyArrow 等 C 程式庫的複雜相依性。

程式碼範例：[PythonDependencies](#)

使用視窗彙總 (DataStream API)

此範例示範在 Python 應用程式中內嵌的 SQL 中的四種視窗化彙總類型。

1. 根據處理時間的滑動視窗
2. 根據事件時間的滑動視窗
3. 根據處理時間的輪轉時段
4. 根據事件時間的輪轉時段

程式碼範例：[視窗調整](#)

使用 S3 接收器

此範例示範如何使用內嵌在 Python 應用程式中的 SQL，將輸出寫入 Amazon S3 做為 JSON 檔案。您必須啟用 S3 接收器的檢查點，才能將檔案寫入和輪換至 Amazon S3。

程式碼範例：[S3Sink](#)

使用使用者定義函數 (UDF)

此範例示範如何定義使用者定義的函數、在 Python 中實作，以及將其用於在 Python 應用程式中執行的 SQL 程式碼。

程式碼範例：[UDF](#)

使用 Amazon Data Firehose 接收器

此範例示範如何使用 SQL 將資料傳送至 Amazon Data Firehose。

程式碼範例：[FirehoseSink](#)

Managed Service for Apache Flink 的 Scala 範例

下列範例示範如何搭配使用 Scala 與 Apache Flink 來建立應用程式。

設定多步驟應用程式

此範例示範如何在 Scala 中設定 Flink 應用程式。它示範如何設定 SBT 專案以包含相依性並建置 uber-JAR。

程式碼範例：[GettingStarted](#)

Amazon Managed Service for Apache Flink 中的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您將受益於資料中心和網路架構，該架構旨在滿足最安全敏感組織的需求。

安全性是 AWS 與您之間의 共同責任。[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 Cloud 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 [AWS 合規計劃](#) 的一部分。若要了解適用於 Managed Service for Apache Flink 的合規計劃，請參閱[合規計劃範圍內的AWS 服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件有助於您了解如何在使用 Managed Service for Apache Flink 時套用共同責任模式。以下主題說明如何設定 Managed Service for Apache Flink 以符合您的安全和合規目標。您也將了解如何使用其他 Amazon 服務，協助監控並保護 Managed Service for Apache Flink 資源。

主題

- [Amazon Managed Service for Apache Flink 中的資料保護](#)
- [Amazon Managed Service for Apache Flink 中的金鑰管理](#)
- [Amazon Managed Service for Apache Flink 的身分和存取管理](#)
- [Amazon Managed Service for Apache Flink 的合規驗證](#)
- [Amazon Managed Service for Apache Flink 中的彈性](#)
- [Managed Service for Apache Flink 中的基礎設施安全性](#)
- [Managed Service for Apache Flink 的安全最佳實務](#)

Amazon Managed Service for Apache Flink 中的資料保護

您可以使用提供的工具來保護您的資料 AWS。Amazon MSF 可以使用支援加密資料的服務，包括 Firehose 和 Amazon S3。

Managed Service for Apache Flink 中的資料加密

靜態加密

請注意以下有關使用 Amazon MSF 加密靜態資料：

- 您可以使用 [StartStreamEncryption](#) 加密傳入的 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 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 [搭配使用伺服器端加密與 KMS 受管金鑰 \(SSE-KMS\) 來保護資料](#)。
- Amazon MSF 可以從任何串流來源讀取，並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

Amazon MSF 會加密傳輸中的所有資料。已為所有 Amazon MSF 應用程式啟用傳輸中加密，且無法停用。

在下列情況下，Amazon MSF 會加密傳輸中的資料：

- 從 Kinesis Data Streams 傳輸到 Amazon MSF 的資料。
- 在 Amazon MSF 內部元件之間傳輸的資料。
- Amazon MSF 和 Firehose 之間傳輸中的資料。

金鑰管理

在 Amazon MSF 中，您可以使用服務受管或您自己的客戶受管金鑰來加密資料。如需詳細資訊，請參閱 [Amazon Managed Service for Apache Flink 中的金鑰管理](#)。

Amazon Managed Service for Apache Flink 中的金鑰管理

在 Amazon MSF 中，您可以選擇使用 [AWS 受管金鑰](#) 或您自己的 [客戶受管金鑰 CMKs](#) 來加密資料。AWS Key Management Service (AWS KMS) 中的 CMKs 是您建立、擁有和管理自己的加密金鑰。

在此頁面

- [Amazon MSF 中的透明加密](#)
- [Amazon MSF 中的客戶受管金鑰](#)

- [在 Amazon MSF 中使用客戶受管金鑰](#)
- [使用 管理 CMK AWS 管理主控台](#)
- [使用 APIs 管理 CMK](#)

Amazon MSF 中的透明加密

根據預設，Amazon MSF 會使用 AWS 擁有的金鑰 (AOKs) 來加密暫時性（執行中的應用程式儲存）和耐用（耐用的應用程式儲存）儲存中的資料。這表示根據預設，受 Flink [檢查點](#)或[快照](#)約束的所有資料都會加密。AOKs 是 Amazon MSF 的預設加密方法，不需要額外設定。若要加密傳輸中的資料，Amazon MSF 預設會使用 TLS 和 HTTP+SSL，而且不需要額外的設定或組態。

Amazon MSF 中的客戶受管金鑰

在 Amazon MSF 中，CMK 是一項功能，可讓您使用建立、擁有和管理金鑰來加密應用程式的資料 AWS KMS。

本節內容

- [使用 CMKs ?](#)
- [哪些項目未使用 CMKs 加密 ?](#)
- [支援的 KMS 金鑰類型](#)
- [KMS 金鑰許可](#)
- [KMS 加密內容和限制條件](#)
- [金鑰輪換政策](#)
- [最低權限金鑰政策陳述式](#)
- [AWS CloudTrail 日誌項目範例](#)

使用 CMKs ?

在 Amazon MSF 應用程式中，受限於 Flink 檢查點或快照的資料將使用您為該應用程式定義的 CMK 加密。CMK 會立即加密存放在執行中應用程式儲存或耐用應用程式儲存中的資料。[下列各節](#)說明為您的 Amazon MSF 應用程式設定 CMKs 的程序。

金鑰輪換政策

Amazon MSF 不會管理 CMKs 的金鑰輪換政策。您必須自行負責金鑰輪換。這是因為您建立和維護 CMKs。如需如何在 Amazon MSF 中使用金鑰輪換政策搭配 CMK 的詳細資訊，請參閱[金鑰輪換政策](#)。

哪些項目未使用 CMKs 加密？

來源和目的地

Amazon MSF 不會管理資料來源和接收器的加密。它由您的來源或接收器組態或應用程式連接器組態管理。

加密的追溯應用程式

Amazon MSF 中的 CMK 不支援追溯性地將 CMKs 套用至現有的歷史快照。

日誌加密

目前，Amazon MSF 不支援對應用程式碼 jar 產生的日誌使用 KMS CMK 進行日誌加密。您需要確保日誌不包含需要 CMK 加密的資料。

加密傳輸中的資料

您無法使用 CMK 加密傳輸中的資料。根據預設，Amazon MSF 會使用 TLS 或 HTTP 和 SSL 加密傳輸中的所有資料。

支援的 KMS 金鑰類型

Amazon MSF 中的 CMK 支援[對稱金鑰](#)。

KMS 金鑰許可

Amazon MSF 中的 CMK 需要執行下列 KMS 動作的許可。這些許可是驗證存取權、建立 CMK 加密執行中應用程式儲存體，以及將 CMK 加密應用程式狀態儲存在耐用的應用程式儲存體中的必要許可。

- [kms:DescribeKey](#)

准許將 KMS 金鑰別名解析為金鑰 ARN。

- [kms:Decrypt](#)

准許存取持久的應用程式狀態，並佈建執行中的應用程式儲存。

- [kms:GenerateDataKey](#)

准許存放持久的應用程式狀態。

- [kms:GenerateDataKeyWithoutPlaintext](#)

准許佈建執行中的應用程式儲存體。

- [kms:CreateGrant](#)

准許存取執行中的應用程式儲存體。

KMS 加密內容和限制條件

Amazon MSF 中的 CMK 會在存取金鑰以讀取或寫入加密資料時提供加密內容，也就是 `kms:EncryptionContext:aws:kinesisanalytics:arn`。除了加密內容之外，在讀取或寫入耐用的應用程式儲存體時，也會提供來源內容 [aws : SourceArn](#) 和 [aws : SourceAccount](#)。

建立佈建加密執行中應用程式儲存的授予時，Amazon MSF CMK 會建立限制類型 [EncryptionContextSubset](#) 的授予，以確保僅允許透過 [進行解密](#) 操作 "`kms:GrantOperations`": "`Decrypt`"。

金鑰輪換政策

Amazon MSF 不會管理 CMKs 的金鑰輪換政策。您需自行負責金鑰輪換，因為您建立和維護 CMKs。

在 KMS 中，您可以使用自動或手動金鑰輪換來為 CMKs 建立新的密碼編譯資料。如需有關如何輪換金鑰的資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [輪換 AWS KMS keys](#)。

當您在 Amazon MSF 中輪換 CMKs 的金鑰時，您必須確定運算子 (API 呼叫者) 具有先前金鑰和新金鑰的許可。

Note

應用程式可以從設定為使用 CMK 後使用 AOK 加密的快照開始。應用程式也可以從使用舊版 CMK 加密的快照開始。若要從快照啟動應用程式，運算子 (API 呼叫者) 必須同時擁有舊金鑰和新金鑰的許可。

在 Amazon MSF 中，我們建議您使用 CMK 加密來停止和重新啟動應用程式。這可確保新的輪換主金鑰會套用至執行中應用程式儲存和耐用應用程式儲存中的所有資料。如果您不停止並重新啟動應用程

式，新的金鑰材料只會套用至耐用的應用程式儲存。執行中的應用程式儲存會繼續使用先前的輪換金鑰材料加密。

如果您要變更改用於 CMK 的 AWS KMS key ARN，您應該在 Amazon MSF 中使用 [UpdateApplication](#)。這將確保您的 Flink 應用程式將在 UpdateApplication 套用 CMK 變更時重新啟動。

Note

當您提供別名或別名 ARN 時，Amazon MSF 會將別名解析為金鑰 ARN，並將金鑰 ARN 儲存為應用程式的已設定金鑰。

最低權限金鑰政策陳述式

如需金鑰政策陳述式的相關資訊，請參閱 [建立 KMS 金鑰政策](#) 和 [應用程式生命週期運算子 \(API 呼叫者\) 許可](#)。

AWS CloudTrail 日誌項目範例

當 Amazon MSF 在中使用 CMKs 時 AWS KMS，AWS CloudTrail 會自動記錄所有 AWS KMS API 呼叫和相關詳細資訊。這些日誌包含資訊，例如 AWS 服務提出請求、KMS 金鑰 ARN、執行的 API 動作，以及排除加密資料的時間戳記。這些日誌透過顯示哪些服務存取了您的金鑰以及何時存取您的金鑰，提供必要的稽核線索，以進行合規、安全監控和故障診斷。

範例 1：使用 Amazon MSF 中擔任的角色 AWS KMS 來解密 API 呼叫

下列 CloudTrail 日誌顯示 Amazon MSF 在 CMK 上執行測試 [kms:Decrypt](#) 操作。Amazon MSF 在使用 [CreateApplication](#) API 時，使用 **Operator** 角色提出此請求。下列日誌包含基本詳細資訊，例如目標 KMS 金鑰 ARN、關聯的 Amazon MSF 應用程式 (*MyCmkApplication*)，以及操作的時間戳記。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "REDACTED",
    "arn": "arn:aws:sts::123456789012:assumed-role/Operator/CmkTestingSession",
    "accountId": "123456789012",
    "accessKeyId": "REDACTED",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "REDACTED",
```

```

        "arn": "arn:aws:iam::123456789012:role/Operator",
        "accountId": "123456789012",
        "userName": "Operator"
    },
    "attributes": {
        "creationDate": "2025-08-07T13:29:28Z",
        "mfaAuthenticated": "false"
    }
},
"invokedBy": "kinesisanalytics.amazonaws.com"
},
"eventTime": "2025-08-07T13:45:45Z",
"eventSource": "kms.amazonaws.com",
"eventName": "Decrypt",
"awsRegion": "us-east-1",
"sourceIPAddress": "kinesisanalytics.amazonaws.com",
"userAgent": "kinesisanalytics.amazonaws.com",
"errorCode": "DryRunOperationException",
"errorMessage": "The request would have succeeded, but the DryRun option is set.",
"requestParameters": {
    "encryptionContext": {
        "aws:kinesisanalytics:arn": "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication"
    },
    "keyId": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "dryRun": true
},
"responseElements": null,
"additionalEventData": {
    "keyMaterialId": "REDACTED"
},
"requestID": "56764d19-1eb1-48f1-8044-594aa7dd05c4",
"eventID": "1371b402-f1dc-4c47-8f3a-1004e4803c5a",
"readOnly": true,
"resources": [
    {
        "accountId": "123456789012",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
],

```

```

    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
  }

```

範例 2：使用直接服務身分驗證 AWS KMS 在 Amazon MSF 中解密 API 呼叫

下列 CloudTrail 日誌顯示 Amazon MSF 在 CMK 上執行測試 [kms:Decrypt](#) 操作。Amazon MSF 透過直接 AWS service-to-service 身分驗證提出此請求，而不是擔任角色。下列日誌包含基本詳細資訊，例如目標 KMS 金鑰 ARN、關聯的 Amazon MSF 應用程式 (*MyCmkApplication*)，以及操作的共用事件 ID。

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "kinesisanalytics.amazonaws.com"
  },
  "eventTime": "2025-08-07T13:45:45Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "kinesisanalytics.amazonaws.com",
  "userAgent": "kinesisanalytics.amazonaws.com",
  "errorCode": "DryRunOperationException",
  "errorMessage": "The request would have succeeded, but the DryRun option is set.",
  "requestParameters": {
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "keyId": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "encryptionContext": {
      "aws:kinesisanalytics:arn": "arn:aws:kinesisanalytics:us-east-1:123456789012:application/MyCmkApplication"
    },
    "dryRun": true
  },
  "responseElements": null,
  "additionalEventData": {
    "keyMaterialId": "REDACTED"
  },
  "requestID": "5fe45ada-7519-4608-be2f-5a9b8ddd62b2",
  "eventID": "6206b08f-ce04-3011-9ec2-55951d357b2c",

```

```
"readOnly": true,
"resources": [
  {
    "accountId": "123456789012",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "Application-account-ID",
"sharedEventID": "acbe4a39-ced9-4f53-9f3c-21ef7e89dc37",
"eventCategory": "Management"
}
```

在 Amazon MSF 中使用客戶受管金鑰

在根據 CMK 政策建立、管理和操作 Amazon MSF 應用程式時，您需要考慮下列因素。

客戶自管金鑰

這是[金鑰政策和金鑰材料](#)。您需要建立一個金鑰，用於在執行應用程式儲存體和耐用的應用程式儲存體中加密您的應用程式狀態。

應用程式生命週期運算子 (API 呼叫者)

這是 **Operator** IAM 使用者或角色。Operator 可以是人工或自動化，例如將建立、部署和執行 Amazon MSF 應用程式的 CI/CD 管道。應用程式生命週期運算子可以是 IAM 角色或使用者。

Note

金鑰管理員和運算子可能是同一個人。在這種情況下，我們建議您一律使用不同的角色或使用者。

應用程式

這是您建立的 Amazon MSF 應用程式。應用程式執行 (IAM) 角色不需要變更即可使用 CMK。如需 Amazon MSF 中 IAM 的詳細資訊，請參閱 [Amazon Managed Service for Apache Flink 的身分和存取管理](#)。

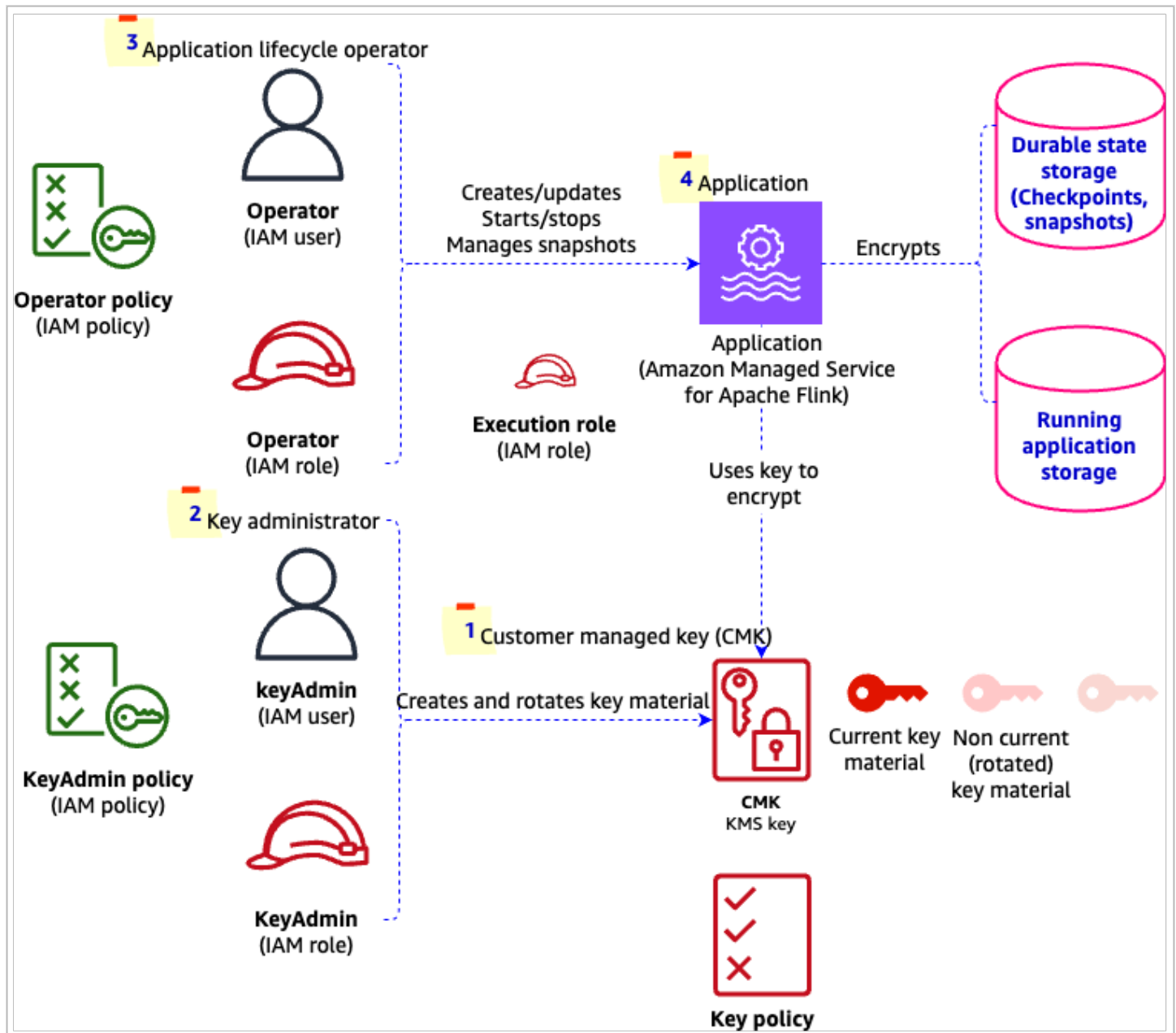
政策之間的相依性

指派給 CMK 的金鑰政策與定義應用程式生命週期運算子許可的 IAM 政策之間存在相互依存性。您可能想要依下列順序建立它們：

- 建立運算子 IAM 使用者或角色，而不用定義 CMK 許可的 IAM 政策。運算子使用 AOK 建立應用程式。
- 建立具有管理 KMS 金鑰許可的金鑰管理員。金鑰管理員會建立 CMK。金鑰政策會參考運算子和管理員角色 ARNs，以及應用程式 ARN。如需詳細資訊，請參閱[建立 KMS 金鑰政策](#)。
- 為 Operator 建立 IAM 政策，允許管理應用程式的 CMK。如需詳細資訊，請參閱[應用程式生命週期運算子 \(API 呼叫者\) 許可](#)。將新的 IAM 政策連接至運算子。運算子會更新啟用 CMK 的應用程式。如需詳細資訊，請參閱[更新現有應用程式以使用 CMK](#)。

如果應用程式不存在，請在沒有 CMK 的情況下建立應用程式。

下圖顯示如何在 Amazon MSF 中實作 CMK。



1. 客戶受管金鑰 (CMK)：包含金鑰政策和金鑰材料。
2. 金鑰管理員：IAM KeyAdmin 使用者或角色。
3. 應用程式生命週期運算子 (API 呼叫者)：運算子 IAM 使用者或角色。
4. 應用程式：已連接執行 (IAM) 角色。

使用 管理 CMK AWS 管理主控台

本主題說明如何使用 建立和更新 KMS CMKs AWS 管理主控台。若要遵循本主題中所述的程序，您必須具有管理 KMS 金鑰和 Amazon MSF 應用程式的許可。本主題中的程序使用寬鬆金鑰政策，僅供示範和測試之用。我們不建議針對生產工作負載使用這類寬鬆的金鑰政策。對於生產工作負載，您可以使用 主控台，但在實際情況下，會隔離角色、許可和工作流程。

開始之前，請先建立 KMS 金鑰。如需有關建立 KMS 金鑰的資訊，請參閱《AWS Key Management Service 開發人員指南》中的[建立 KMS 金鑰](#)。

建立和指派 KMS 金鑰

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在串流應用程式頁面上，選擇建立串流應用程式。
3. 對於 Apache Flink 版本，請務必選擇 Apache Flink 1.20。
4. 針對加密，選擇使用客戶受管金鑰。
5. 如果您沒有 KMS 金鑰，請選擇建立 AWS KMS key，然後建立 KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱《AWS Key Management Service 開發人員指南》中的[使用 AWS KMS 主控台](#)。
6. 如果您沒有 KMS 金鑰，請選擇建立 AWS KMS key，然後建立 KMS 金鑰。如需有關如何使用主控台建立金鑰的資訊，請參閱[建立對稱加密 KMS 金鑰](#)。
7. 在您要使用的選取器中選擇金鑰。請記住，僅允許狀態為已啟用的金鑰。

更新現有應用程式以使用 CMK

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在串流應用程式頁面上，選擇具有 Flink 1.20 版的應用程式。
3. 選擇設定。
4. 針對加密，選擇使用客戶受管金鑰。
5. 如果您沒有 KMS 金鑰，請選擇建立 AWS KMS key，然後建立 KMS 金鑰。如需有關如何使用主控台建立金鑰的資訊，請參閱[建立對稱加密 KMS 金鑰](#)。
6. 在您要使用的選取器中選擇金鑰。請記住，僅允許狀態為已啟用的金鑰。

從 CMK 切換到 AWS 擁有的金鑰

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
2. 在串流應用程式頁面上，選擇具有 Flink 1.20 版的應用程式。
3. 選擇設定。
4. 針對加密，選擇使用 AWS 擁有的金鑰。

使用 APIs 管理 CMK

本主題說明如何使用 Amazon MSF APIs 建立和更新 KMS CMKs。若要遵循本主題中所述的程序，您必須具有管理 KMS 金鑰和 Amazon MSF 應用程式的許可。本主題中的程序使用寬鬆金鑰政策，僅供示範和測試之用。我們不建議針對生產工作負載使用這類寬鬆的金鑰政策。在生產工作負載的真實案例中，會隔離角色、許可和工作流程。

在此頁面

- [建立和指派 KMS 金鑰](#)
- [更新現有應用程式以使用 CMK](#)
- [從 CMK 還原至 AWS 擁有的金鑰](#)

建立和指派 KMS 金鑰

開始之前，請先建立 KMS 金鑰。如需有關建立 KMS 金鑰的資訊，請參閱《AWS Key Management Service 開發人員指南》中的[建立 KMS 金鑰](#)。

本節內容

- [建立 KMS 金鑰政策](#)
- [應用程式生命週期運算子 \(API 呼叫者\) 許可](#)

建立 KMS 金鑰政策

若要在 Amazon MSF 中使用 CMK，您必須將下列服務主體新增至您的金鑰政策：

`kinesisanalytics.amazonaws.com`和

`infrastructure.kinesisanalytics.amazonaws.com`。Amazon MSF 使用這些服務主體進行驗證和資源存取。如果您未包含這些服務主體，Amazon MSF 會拒絕請求。

下列 KMS 金鑰政策可讓 Amazon MSF 將 CMK 用於應用程式 *MyCmkApplication*。此政策會將必要的許可授予**Operator**角色和 Amazon MSF 服務主體kinesisanalytics.amazonaws.com，以及 infrastructure.kinesisanalytics.amazonaws.com，以執行下列操作：

- 描述 CMK
- 加密應用程式資料
- 解密應用程式資料
- 建立金鑰的授予

下列範例使用 IAM 角色。您可以使用下列範例做為範本，建立 KMS 金鑰的金鑰政策，但請務必執行下列動作：

- arn:aws:iam::123456789012:role/Operator 將取代為 **Operator**角色。您必須先建立**Operator**角色或使用者，才能建立金鑰政策。否則會導致請求失敗。
- arn:aws:kinesisanalytics:us-east-1:123456789012:application/*MyCmkApplication* 將取代為您應用程式的 ARN。
- kinesisanalytics.us-east-1.amazonaws.com 將取代為對應區域的服務值。
- 將 123456789012 取代為您的 CMK 帳戶 idKey 政策。
- 新增其他政策陳述式，[以允許金鑰管理員管理 KMS 金鑰](#)。否則會導致失去管理金鑰的存取權。

下列金鑰政策陳述式很大，因為它們旨在明確顯示每個動作所需的條件。

```
{
  "Version": "2012-10-17",
  "Id": "MyMsfCmkApplicationKeyPolicy",
  "Statement": [
    {
      "Sid": "AllowOperatorToDescribeKey",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/Operator"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "Sid": "AllowOperatorToConfigureAppToUseKeyForApplicationState",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::123456789012:role/Operator"
    },
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyWithoutPlaintext"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:kinesisanalytics:arn":
          "arn:aws:kinesisanalytics:us-east-1:123456789012:application/MyCmkApplication",
        "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com"
      }
    }
  },
  {
    "Sid": "AllowOperatorToConfigureAppToCreateGrantForRunningState",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::123456789012:role/Operator"
    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:kinesisanalytics:arn":
          "arn:aws:kinesisanalytics:us-east-1:123456789012:application/MyCmkApplication",
        "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com",
        "kms:GrantConstraintType": "EncryptionContextSubset"
      },
      "ForAllValues:StringEquals": {
        "kms:GrantOperations": "Decrypt"
      }
    }
  },
},

```

```

{
  "Sid": "AllowMSFServiceToDescribeKey",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "kinesisanalytics.amazonaws.com",
      "infrastructure.kinesisanalytics.amazonaws.com"
    ]
  },
  "Action": "kms:DescribeKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceArn":
        "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication",
      "aws:SourceAccount": "123456789012"
    }
  }
},
{
  "Sid": "AllowMSFServiceToGenerateDataKeyForDurableState",
  "Effect": "Allow",
  "Principal": {
    "Service": "kinesisanalytics.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceArn":
        "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication",
      "kms:EncryptionContext:aws:kinesisanalytics:arn":
        "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication",
      "aws:SourceAccount": "123456789012"
    }
  }
},
{
  "Sid": "AllowMSFServiceToDecryptForDurableState",

```

```

    "Effect": "Allow",
    "Principal": {
      "Service": "kinesisanalytics.amazonaws.com"
    },
    "Action": [
      "kms:Decrypt"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:kinesisanalytics:arn":
          "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication"
      }
    }
  },
  {
    "Sid": "AllowMSFServiceToUseKeyForRunningState",
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "infrastructure.kinesisanalytics.amazonaws.com"
      ]
    },
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKeyWithoutPlaintext"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:kinesisanalytics:arn":
          "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication"
      }
    }
  },
  {
    "Sid": "AllowMSFServiceToCreateGrantForRunningState",
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "infrastructure.kinesisanalytics.amazonaws.com"
      ]
    }
  }
}

```

```

    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:kinesisanalytics:arn":
          "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication",
        "kms:GrantConstraintType": "EncryptionContextSubset"
      },
      "ForAllValues:StringEquals": {
        "kms:GrantOperations": "Decrypt"
      }
    }
  }
]
}

```

應用程式生命週期運算子 (API 呼叫者) 許可

下列 IAM 政策可確保應用程式生命週期運算子具有將 KMS 金鑰指派給應用程式 *MyCmkApplication* 的必要許可。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowMSFAPICalls",
      "Effect": "Allow",
      "Action": "kinesisanalytics:*",
      "Resource": "*"
    },
    {
      "Sid": "AllowPassingServiceExecutionRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/MyCmkApplicationRole"
    }
  ]
}

```

```

        "Sid": "AllowDescribeKey",
        "Effect": "Allow",
        "Action": [
            "kms:DescribeKey"
        ],
        "Resource": "arn:aws:kms:us-  
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com"
            }
        }
    },
    {
        "Sid": "AllowMyCmkApplicationKeyOperationsForDurableState",
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
        ],
        "Resource": "arn:aws:kms:us-  
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com",
                "kms:EncryptionContext:aws:kinesisanalytics:arn":  
"arn:aws:kinesisanalytics:us-  
east-1:123456789012:application/MyCmkApplication"
            }
        }
    },
    {
        "Sid": "AllowMyCmkApplicationKeyOperationsForRunningState",
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKeyWithoutPlaintext"
        ],
        "Resource": "arn:aws:kms:us-  
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com",
                "kms:EncryptionContext:aws:kinesisanalytics:arn":

```

```

        "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication"
    },
    {
        "Sid": "AllowMyCmkApplicationCreateGrantForRunningState",
        "Effect": "Allow",
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "Condition": {
            "ForAllValues:StringEquals": {
                "kms:GrantOperations": "Decrypt"
            },
            "StringEquals": {
                "kms:ViaService": "kinesisanalytics.us-east-1.amazonaws.com",
                "kms:EncryptionContext:aws:kinesisanalytics:arn":
                    "arn:aws:kinesisanalytics:us-
east-1:123456789012:application/MyCmkApplication",
                "kms:GrantConstraintType": "EncryptionContextSubset"
            }
        }
    }
]
}

```

更新現有應用程式以使用 CMK

在 Amazon MSF 中，您可以將 CMK 政策套用至使用 AWS 擁有的金鑰 (AOKs) 現有應用程式。

根據預設，Amazon MSF 會使用 AOKs 來加密暫時性（執行中的應用程式儲存）和耐用（耐用的應用程式儲存）儲存中的所有資料。這表示根據預設，受 Flink [檢查點](#) 或 [快照](#) 約束的所有資料都會使用 AOKs 加密。當您以 CMK 取代 AOK 時，新的檢查點和快照會以 CMK 加密。不過，歷史快照將保持使用 AOK 加密。

更新現有應用程式以使用 CMK

1. 使用下列組態建立 JSON 檔案。

請務必將 `CurrentApplicationVersionId` 的值取代為應用程式的目前版本編號。您可以使用 [DescribeApplication](#) 取得應用程式的目前版本編號。

在此 JSON 組態中，請記得將##值取代為實際值。

```
{
  "ApplicationName": "MyCmkApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationEncryptionConfigurationUpdate": {
      "KeyTypeUpdate": "CUSTOMER_MANAGED_KEY",
      "KeyIdUpdate": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

2. 儲存此檔案。例如，將其儲存為名稱 **enable-cmk.json**。
3. 執行 [update-application](#) AWS CLI 命令，如下列範例所示。在此命令中，提供您在先前步驟中建立的 JSON 組態檔案做為檔案引數。

```
aws kinesisanalyticstv2 update-application \
  --cli-input-json file://enable-cmk.json
```

只有在符合下列條件時，才會接受上述組態來更新應用程式以使用 CMK：

- API 呼叫者具有允許存取金鑰的政策陳述式。
- 金鑰政策具有允許 API 呼叫者存取金鑰的政策陳述式。
- 金鑰政策具有允許 Amazon MSF 服務主體 `kinesisanalytics.amazonaws.com` 存取金鑰的政策陳述式。

從 CMK 還原至 AWS 擁有的金鑰

從 CMK 還原至 AOK

1. 使用下列組態建立 JSON 檔案。

在此 JSON 組態中，請記得將##值取代為實際值。

```
{
  "ApplicationName": "MyCmkApplication",
```



```
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
  "ApplicationEncryptionConfigurationUpdate": {
    "KeyTypeUpdate": "AWS_OWNED_KEY"
  }
}
```

2. 儲存此檔案。例如，將其儲存為名稱 **disable-cmk.json**。
3. 執行 [update-application](#) AWS CLI 命令，如下列範例所示。在此命令中，提供您在先前步驟中建立的 JSON 組態檔案做為檔案引數。

```
aws kinesisanalyticstv2 update-application \
  --cli-input-json file://disable-cmk.json
```

Amazon Managed Service for Apache Flink 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可以控制誰能完成身分驗證 (登入) 和獲得授權 (具有許可)，而得以使用 Managed Service for Apache Flink 資源。IAM 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用](#)
- [Amazon Managed Service for Apache Flink 的身分型政策範例](#)
- [Amazon Managed Service for Apache Flink 的身分和存取權疑難排解](#)
- [預防跨服務混淆代理人](#)

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (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 — 負責設定使用者存取權並提交相關許可請求 (請參閱 [Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用](#))
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 [Amazon Managed Service for Apache Flink 的身分型政策範例](#))

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的[如何登入您的 AWS 帳戶](#)。

對於程式設計存取，AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的[API 請求的AWS 第 4 版簽署程序](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務和資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的[需要根使用者憑證的任務](#)。

聯合身分

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務使用臨時憑證存取。

聯合身分是您企業目錄、Web 身分提供者的使用者，或是 AWS 服務使用身分來源的登入資料 Directory Service 存取的使用者。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center ?](#)。

IAM 使用者和群組

IAM 使用者https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的[要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取](#)。

[IAM 群組](#)會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 使用者的使用案例](#)。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過[從使用者切換到 IAM 角色（主控台）](#)或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的快帳戶資源存取](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS，會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 [JSON 政策概觀](#)。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的[在受管政策與內嵌政策之間選擇](#)。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中[指定主體](#)。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- **許可界限** — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 實體許可界限](#)。
- **服務控制政策 (SCP)** — 為 AWS Organizations 中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- **資源控制政策 (RCP)** — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [資源控制政策 \(RCP\)](#)。
- **工作階段政策** — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的 [工作階段政策](#)。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的 [政策評估邏輯](#)。

Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用

在 Amazon MSF 中，您會在下列不同內容中使用 IAM：

- [應用程式許可](#)：控制應用程式對使用 IAM 身分驗證之外部資源的存取，例如 Amazon S3、Amazon Kinesis Data Streams 或 Amazon DynamoDB。
- [應用程式管理和生命週期控制許可](#)：控制 Amazon MSF API 動作的使用，例如 [CreateApplication](#)、[StartApplication](#) 和 [UpdateApplication](#)，以控制應用程式生命週期。如需您可以在 IAM 政策陳述式的 Action 元素中指定之所有 Amazon MSF API 動作的完整清單，請參閱《服務授權參考》中的 [Amazon Kinesis Analytics V2 定義的動作](#)。

主題

- [應用程式許可](#)
- [應用程式管理和生命週期控制許可](#)

- [Managed Service for Apache Flink 內的身分型政策](#)
- [Managed Service for Apache Flink 內的資源型政策](#)
- [Managed Service for Apache Flink 中的存取控制清單 \(ACL\)](#)
- [Managed Service for Apache Flink 的服務角色](#)
- [適用於 Managed Service for Apache Flink 的服務連結角色](#)

應用程式許可

您可以在應用程式組態中，使用指派給應用程式的 IAM 角色來控制 Amazon MSF 應用程式的 IAM 許可。此 IAM 角色決定應用程式存取其他服務的許可，例如使用 IAM 進行授權的 Amazon S3、Kinesis Data Streams 或 DynamoDB。

Warning

變更服務角色的許可可能會中斷 Amazon MSF 功能。請確定您未移除應用程式從 Amazon S3 儲存貯體下載應用程式碼的許可，並將日誌傳送至 Amazon CloudWatch。

不支援使用[資源型政策](#)將許可指派給應用程式。您無法在連接到要存取之資源的政策中，將 Amazon MSF 應用程式指定為主體。

主題

- [存取應用程式碼和應用程式日誌的許可](#)
- [預防跨服務混淆代理人](#)

存取應用程式碼和應用程式日誌的許可

Amazon MSF 也會使用應用程式 IAM 角色來存取 Amazon S3 儲存貯體中上傳的應用程式碼，並將應用程式日誌寫入 Amazon CloudWatch Logs。

當您使用 建立或更新應用程式時 AWS 管理主控台，請選擇使用應用程式組態中的必要政策建立/更新 IAM 角色 <role-name>，Amazon MSF 會自動建立和修改將必要許可指派給 Amazon S3 和 CloudWatch Logs 的 IAM 角色。

如果您手動建立 IAM 角色，或使用自動化工具建立和管理應用程式，則必須將下列許可新增至應用程式 IAM 角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/path-to-application-code"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/kinesis-
analytics/application-name:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],

```

```
"Resource": [
  "arn:aws:logs:us-east-1:123456789012:log-group:/aws/kinesis-
analytics/application-name:log-stream:kinesis-analytics-log-stream"
]
}
```

預防跨服務混淆代理人

當 Amazon MSF 應用程式呼叫不同的 AWS 服務，您可以提供更精細的存取許可。例如，如果 IAM 角色在多個應用程式之間重複使用，應用程式可能會存取不應存取的資源。這稱為[混淆代理人問題](#)。如需存取的資源如何限制存取特定 Amazon MSF 應用程式的相關資訊，請參閱[預防跨服務混淆代理人](#)。

應用程式管理和生命週期控制許可

管理應用程式及其生命週期的動作，例如 [CreateApplication](#)、[StartApplication](#) 和 [UpdateApplication](#)，是透過與執行動作之資源相關聯的身分型政策來控制，例如 IAM 使用者、IAM 群組或 AWS Lambda 呼叫 Amazon MSF API 等資源。

Note

基於回溯相容性原因，控制 Amazon MSF 應用程式生命週期的 API 和 SDK 稱為 Amazon Kinesis Analytics V2。

不支援使用連接至 Amazon MSF 應用程式的資源型政策來指派應用程式生命週期動作的許可。應用程式 IAM 角色不會用於控制對應用程式生命週期動作的存取。您不應將應用程式生命週期許可新增至應用程式角色。

下表列出您可以搭配 Amazon MSF 應用程式生命週期動作使用的 IAM 功能。

IAM 功能	Managed Service for Apache Flink 支援
身分型政策	是
資源型政策	否
政策動作	是

IAM 功能	Managed Service for Apache Flink 支援
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
跨服務主體許可	是
服務角色	否
服務連結角色	否

- 如需 Managed Service for Apache Flink 和其他 如何與大多數 IAM 功能 AWS 服務 搭配使用的高階檢視，請參閱 [《AWS 服務 IAM 使用者指南》中的 與 IAM 搭配使用](#)。
- 如需可在 IAM 許可政策中使用的服務特定資源、動作和條件內容金鑰的相關資訊，請參閱服務授權參考中的 [Amazon Kinesis Analytics V2 的動作、資源和條件金鑰](#)。

主題

- [應用程式生命週期政策動作](#)
- [應用程式生命週期政策資源](#)
- [應用程式生命週期政策條件索引鍵](#)
- [屬性型存取控制 \(ABAC\)用於 Managed Service for Apache Flink](#)
- [使用暫時登入資料](#)
- [跨服務主體許可](#)

應用程式生命週期政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

Amazon MSF 中的政策動作在動作之前使用 `kinesisanalytics:`字首。Amazon MSF APIs和 SDKs 使用 Amazon Kinesis Analytics V2字首。

若要在單一陳述式中指定多個動作，請用逗號分隔。下列範例顯示指定 Amazon MSF 政策動作的語法。

```
"Action" : [  
    "kinesisanalytics:action1",  
    "kinesisanalytics:action2"  
]
```

您也可以使用萬用字元 (*) 指定多個動作。例如，如需指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "kinesisanalytics:Describe*"
```

若要查看您可以在 IAM 政策陳述式的 Action元素中指定的所有 Amazon MSF API 動作的完整清單，請參閱 [Amazon Kinesis Analytics V2 定義的動作](#)。

若要檢視 Amazon MSF 身分型政策的範例，請參閱 [身分型政策範例](#)。

應用程式生命週期政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

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

會為每個應用程式定義 Amazon MSF 應用程式生命週期動作的許可。IAM 政策中的 Resource JSON 元素會定義適用許可的 Amazon MSF 應用程式。

您可以使用萬用字元指定應用程式 ARN 或一組應用程式，將許可指派給單一應用程式。下列範例顯示 Resource元素的語法。

```
"Resource" : "arn:partition:kinesisanalytics:Region:account:application/application-name"
```

您也可以指派許可，以使用萬用字元控制應用程式子集。例如，您可以指派許可來控制名稱開頭為特定字首的所有應用程式。

```
"Resource" : "arn:partition:kinesisanalytics:Region:account:application/application-name-prefix*"
```

應用程式生命週期政策條件索引鍵

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

您可以使用條件金鑰來控制 Amazon MSF 應用程式生命週期動作的許可。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](#)。

屬性型存取控制 (ABAC) 用於 Managed Service for Apache Flink

支援 ABAC (政策中的標籤)：是

您可以使用條件金鑰實作屬性型存取控制 (ABAC)，這是根據屬性定義許可的授權策略。在 AWS 中，這些屬性稱為標籤。您可以將標籤連接到 IAM 實體（使用者或角色）和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。然後，您可以設計 ABAC 政策，以便在委託人的標籤符合其嘗試存取之資源上的標籤時允許操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

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 ABAC 的詳細資訊，請參閱[使用 ABAC 授權根據屬性定義許可](#)。

- 若要檢視包含設定 ABAC 步驟的教學課程，請參閱 [IAM 教學課程：根據標籤定義存取 AWS 資源的許可](#)。

使用暫時登入資料

支援臨時憑證：是

Amazon MSF 應用程式生命週期動作支援臨時登入資料。

如果您 AWS 管理主控台 使用使用者名稱和密碼以外的任何方法登入，則會使用臨時登入資料。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登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱[從使用者切換到 IAM 角色（主控台）](#)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。我們建議您動態產生臨時憑證，而不是使用長期存取金鑰。如需詳細資訊，請參閱[IAM 中的暫時性安全憑證](#)。

跨服務主體許可

支援轉寄存取工作階段 (FAS)：是

Amazon MSF 應用程式生命週期動作支援跨服務主體許可。

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。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。轉送存取工作階段 (FAS) 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務或資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[轉發存取工作階段](#)。

Managed Service for Apache Flink 內的身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素參考](#)。

Managed Service for Apache Flink 內的身分型政策範例

如需檢視 Managed Service for Apache Flink 身分型政策範例，請參閱 [Amazon Managed Service for Apache Flink 的身分型政策範例](#)。

Managed Service for Apache Flink 內的資源型政策

Amazon Managed Service for Apache Flink 目前不支援資源型存取控制。

Managed Service for Apache Flink 中的存取控制清單 (ACL)

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Managed Service for Apache Flink 的服務角色

支援服務角色：是

服務角色是服務擔任的 [IAM 角色](#)，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可給 AWS 服務](#)。

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的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服务連結角色的詳細資訊，請參閱[可搭配 IAM 運作的 AWS 服務](#)。在資料表中尋找服務，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。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的[建立 IAM 政策 \(主控台\)](#)。

如需 Managed Service for Apache Flink 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的[Amazon Managed Service for Apache Flink 適用的動作、資源和條件索引鍵](#)。

主題

- [政策最佳實務](#)
- [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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的[AWS 受管政策](#)或[任務職能的AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱《IAM 使用者指南》中的[IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM JSON 政策元素：條件](#)。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access

Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM Access Analyzer 驗證政策](#)。

- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[透過 MFA 的安全 API 存取](#)。

如需 IAM 中最佳實務的相關資訊，請參閱《IAM 使用者指南》中的[IAM 安全最佳實務](#)。

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 帳戶帳戶中 Managed Service for Apache Flink 資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍然可以使用 Managed Service for Apache Flink 主控台，也請將 Managed Service for Apache Flink ConsoleAccess或ReadOnly AWS 受管政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的[新增許可到使用者](#)。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ]
    }
  ],
```



```

    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
  },
  {
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
      "iam:GetGroupPolicy",
      "iam:GetPolicyVersion",
      "iam:GetPolicy",
      "iam:ListAttachedGroupPolicies",
      "iam:ListGroupPolicies",
      "iam:ListPolicyVersions",
      "iam:ListPolicies",
      "iam:ListUsers"
    ],
    "Resource": "*"
  }
]
}

```

Amazon Managed Service for Apache Flink 的身分和存取權疑難排解

請使用以下資訊來協助診斷和修正使用 Managed Service for Apache Flink 和 IAM 時可能遇到的常見問題。

主題

- [我未獲授權，無法在 Managed Service for Apache Flink 中執行動作](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許 AWS 帳戶外的人員存取我的 Managed Service for Apache Flink 資源](#)

我未獲授權，無法在 Managed Service for Apache Flink 中執行動作

如果 AWS 管理主控台告訴您無權執行動作，則必須聯絡您的管理員尋求協助。您的管理員是為您提供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。

我未獲得執行 iam:PassRole 的授權

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 服務可讓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。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 Managed Service for Apache Flink 中執行動作時，會發生下列範例所示的錯誤。但是，該動作要求服務具備服務角色授與的許可。Mary 沒有將角色傳遞給服務的許可。

```
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 AWS 帳戶外的人員存取我的 Managed Service for Apache Flink 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Managed Service for Apache Flink 是否支援這些功能，請參閱 [Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用](#)。
- 若要了解如何提供您擁有 AWS 帳戶的資源存取權，請參閱《[IAM 使用者指南](#)》中的在[您擁有 AWS 帳戶的另一個中為 IAM 使用者提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《[IAM 使用者指南](#)》中的[將存取權提供給第三方 AWS 帳戶擁有](#)。

- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的[將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的[IAM 中的跨帳戶資源存取](#)。

預防跨服務混淆代理人

在中 AWS，當一個服務（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。

為了防止混淆代理人，AWS 提供工具，協助您使用可存取您帳戶中資源的服務主體來保護所有服務的資料。本節重點介紹特定於 Managed Service for Apache Flink 的防範跨服務混淆代理人；但是，您可以在《IAM 使用者指南》的[混淆代理人問題](#)一節，了解此主題的更多資訊。

在 Managed Service for Apache Flink 環境中，我們建議在角色信任政策中使用 [aws: SourceArn](#) 和 [aws:SourceAccount](#) 全域條件環境索引鍵，以將角色的存取權限制為僅由預期資源產生的請求。

如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 `aws:SourceAccount`。

`aws:SourceArn` 的值必須是 Managed Service for Apache Flink 所使用之資源的 ARN，其格式指定如下：`arn:aws:kinesisanalytics:region:account:resource`。

防範混淆代理人問題的建議方法是使用 `aws:SourceArn` 全域條件內容索引鍵以及資源的完整 ARN。

如果不知道資源的完整 ARN，或指定了多個資源，請使用 `aws:SourceArn` 索引鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如：`arn:aws:kinesisanalytics::111122223333:*`。

您提供給 Managed Service for Apache Flink 的角色政策，以及為您產生之角色的信任政策，都可以使用這些索引鍵。

請執行下列步驟以防範混淆代理人問題：

如要防範混淆代理人問題

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
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. 在編輯信任政策頁面上，將預設 JSON 政策取代為使用一個或兩個 `aws:SourceArn` 與 `aws:SourceAccount` 全域條件內容金鑰的政策。請參閱以下政策範例：
5. 選擇 更新政策。

JSON

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

Amazon Managed Service for Apache Flink 的合規驗證

在多個合規計畫中，第三方稽核人員會評估 Amazon Managed Service for Apache Flink 的安全性和 AWS 合規性。這些包括 SOC、PCI、HIPAA 等。

如需特定合規計畫範圍內 AWS 的服務清單，請參閱 [AWS 合規計畫](#)。

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 [在 中下載報告 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。若您使用 Managed Service for Apache Flink 必須遵循特定標準，如 HIPAA 或 PCI，AWS 會提供資源予以協助：

- [安全與合規快速入門指南](#) – 這些部署指南討論架構考量，並提供在其中部署以安全與合規為重心的基準環境的步驟 AWS。
- [Amazon Web Services 上的 HIPAA 安全與合規架構](#)。本白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。
- [AWS 合規資源](#) – 此工作手冊和指南的集合可能適用於您的產業和位置。
- [AWS Config](#) – AWS 此服務會評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub CSPM](#) – AWS 此服務提供 內安全狀態的全方位檢視 AWS，可協助您檢查是否符合安全產業標準和最佳實務。

FedRAMP

AWS FedRAMP 合規計劃包含 Managed Service for Apache Flink 作為 FedRAMP 授權的服務。如果您是聯邦或商業客戶，您可以使用 服務在 AWS GovCloud (US) 區域的授權界限中處理和存放敏感工作負載，並將資料儲存到高影響層級，以及將資料儲存到中等層級的美國東部（維吉尼亞北部）、美國東部（俄亥俄）、美國西部（加利佛尼亞北部）、美國西部（奧勒岡）區域。

您可以透過 AWS FedRAMP PMO、您的 AWS 銷售客戶經理請求存取 FedRAMP 安全套件，也可以透過 Artifact 中的 AWS [AWS Artifact](#) 下載。

如需詳細資訊，請參閱 [FedRAMP](#)。

Amazon Managed Service for Apache Flink 中的彈性

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置的。AWS 區域提供多個實體隔離和隔離的可用區域，這些區域以低延遲、高輸送量和高度備援的網路連接。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 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

除了 AWS 全球基礎設施之外，Managed Service for Apache Flink 還提供數種功能，以協助支援您的資料彈性和備份需求。

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 叢集透過 JobManager 在高可用性模式下使用 Zookeeper 跨多個可用區域執行。Managed Service for Apache Flink 使用 Amazon EKS 部署 Apache Flink。多個 Kubernetes Pod 用於跨可用區域的每個 AWS 區域 Amazon EKS。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[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[使用 CreateApplicationSnapshot](#) 動作的 SnapshotName 參數建立版本。

Managed Service for Apache Flink 會對儲存在檢查點和儲存點中的資料進行加密。

Managed Service for Apache Flink 中的基礎設施安全性

作為受管服務，Managed Service for Apache Flink 受到 [Amazon Web Services：安全程序概觀](#) 白皮書中所述 AWS 的全球網路安全程序的保護。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Managed Service for Apache Flink。對 Managed Service for Apache Flink 的所有 API 呼叫都會透過 Transport Layer Security (TLS) 保護，並透過 IAM 進行驗證。用戶端必須支援 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 [AWS Security Token Service](#) (AWS STS) 來產生暫時安全憑證來簽署請求。

Managed Service for Apache Flink 的安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon Managed Service for Apache Flink 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

實作最低權限存取

當您授與許可時，需要決定哪些使用者會取得哪些 Apache Flink 資源的哪些許可。您還需針對這些資源啟用允許執行的動作，因此，您只應授與執行任務所需的許可。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限存取是相當重要的一環。

使用 IAM 角色存取其他 Amazon 服務

您的 Managed Service for Apache Flink 應用程式必須具有有效的登入資料，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。

反之，您應使用 IAM 角色來管理暫時性憑證，讓應用程式存取其他資源。使用角色時，您不必使用長期憑證來存取其他資源。

如需詳細資訊，請參閱《IAM 使用者指南》中的以下主題：

- [IAM 角色](#)
- [常見的角色方案：使用者、應用程式和服務](#)

在相依資源中實作伺服器端加密

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 [資料保護](#)。

使用 CloudTrail 監控 API 呼叫

Managed Service for Apache Flink 已與整合 AWS CloudTrail，此服務提供使用者、角色或 Managed Service for Apache Flink 中 Amazon 服務所採取動作的記錄。

您可以利用 CloudTrail 所收集的資訊來判斷向 Managed Service for Apache Flink 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

如需詳細資訊，請參閱[the section called “使用 記錄 Managed Service for Apache Flink API 呼叫 AWS CloudTrail”](#)。

在 Amazon Managed Service for Apache Flink 中記錄和監控

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控資料，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 Managed Service for Apache Flink](#)
- [Managed Service for Apache Flink 中的監控](#)
- [在 Managed Service for Apache Flink 中設定應用程式記錄](#)
- [使用 CloudWatch Logs Insights 分析日誌](#)
- [Managed Service for Apache Flink 中的指標和維度](#)
- [將自訂訊息寫入 CloudWatch Logs](#)
- [使用 記錄 Managed Service for Apache Flink API 呼叫 AWS CloudTrail](#)

登入 Managed Service for Apache Flink

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日誌項目並將其轉寄至 CloudWatch Logs。雖然有些記錄正常且理想，但大量記錄可能會使服務多載，並導致 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訊息產生日誌輸出，請在 Flink 應用程式內使用額外的資料串流，並使用適當的接收器將資料傳送到 Amazon S3 或 CloudWatch。請勿將 Java 記錄系統用於此目的。此外，Managed Service for Apache Flink 的 Debug Monitoring Log Level 設定會產生大量流量，從而會造成背壓。您只能在主動調查應用程式問題時使用它。

使用 CloudWatch Logs Insights 查詢日誌

CloudWatch Logs Insights 是一項功能強大的服務，可大規模查詢日誌。客戶應利用其功能快速搜尋日誌，以識別並減輕操作事件期間的錯誤。

下列查詢會在所有任務管理員記錄中尋找例外狀況，並根據其發生的時間排序。

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
  @message like /(Error|Exception)/
| sort @timestamp desc
```

如需其他有用的查詢，請參閱[範例查詢](#)。

Managed Service for Apache Flink 中的監控

在生產環境中執行串流應用程式時，您著手持續且無限期地執行應用程式。它對實現所有元件的監控和適當報警至關重要，不只是對 Flink 應用程式。否則，您可能會較早就錯過新出現的問題，只在問題完全顯現並且更難以緩解之後才意識到操作事件。要監控的一般事項包括：

- 來源是否正在擷取資料？
- 是否已從來源讀取資料 (從來源角度看)？
- Flink 應用程式是否正在接收資料？
- Flink 應用程式能夠跟上還是落後？
- Flink 應用程式是否將資料保存到接收器中 (從應用程式角度看)？
- 接收器是否正在接收資料？

然後應該考慮針對 Flink 應用程式的更具體指標。此 [CloudWatch 儀表板](#) 提供了一個不錯的起點。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 Managed Service for Apache Flink 使用 CloudWatch 警示](#)。這些指標包括：

- `records_lag_max` 和 `millisbehindLatest`：如果應用程式正在從 Kinesis 或 Kafka 取用，這些指標會指出應用程式是否落後，需要進行擴展以跟上目前的負載。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`：這些指標可清楚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 綁定，否則可用於主動擴展。
- `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`：這些指標監控有多少資料存儲在狀態中以及需要多長時間執行檢查點。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。
- `numberOfFailedCheckpoint`：此指標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 Managed Service for Apache Flink 中設定應用程式記錄

透過將 Amazon CloudWatch 記錄選項新增至您的 Managed Service for Apache Flink 應用程式，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 Logs 串流。CloudWatch 記錄選項是應用程式設定和許可的集合，您的應用程式用它來設定將應用程式事件寫入 CloudWatch Logs 的方式。您可以使用 AWS 管理主控台 或 () 新增和設定 CloudWatch AWS Command Line Interface 記錄選項 AWS CLI。

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，Managed Service for Apache Flink 會為您建立 CloudWatch 日誌群組和日誌串流，並新增應用程式寫入日誌串流所需的許可。
- 使用 API 新增 CloudWatch 記錄選項時，還必須建立應用程式的日誌群組和日誌串流，並新增應用程式寫入日誌串流所需的許可。

使用主控台設定 CloudWatch 記錄

在主控台中為應用程式啟用 CloudWatch 記錄時，會為您建立 CloudWatch 日誌群組和日誌串流。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。

Managed Service for Apache Flink 會建立使用下列慣例命名的日誌群組，其中 *ApplicationName* 是應用程式的名稱。

```
/aws/kinesis-analytics/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級”](#)。

使用 CLI 設定 CloudWatch 記錄

若要使用 新增 CloudWatch 記錄選項 AWS CLI，請完成下列操作：

- 建立 CloudWatch 日誌群組和日誌串流。
- 當您使用 [CreateApplication](#) 動作建立應用程式時，新增記錄選項，或使用 [AddApplicationCloudWatchLoggingOption](#) 動作將記錄選項新增至現有的應用程式。
- 將許可新增至應用程式的政策中，以寫入日誌。

建立 CloudWatch 日誌群組和日誌串流

您可以使用 CloudWatch Logs 主控台或 API 來建立 CloudWatch 日誌群組和日誌串流。如需如何建立 CloudWatch 日誌群組和日誌串流的相關資訊，請參閱[使用日誌群組和日誌串流](#)。

使用應用程式 CloudWatch 記錄選項

使用下列 API 動作將 CloudWatch 日誌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日誌選項。如需如何使用 JSON 檔案作為 API 動作輸入的相關資訊，請參閱 [Managed Service for Apache Flink API 範例程式碼](#)。

建立應用程式時新增 CloudWatch 日誌選項

下列範例示範如何在建立應用程式時使用 `CreateApplication` 動作新增 CloudWatch 日誌選項。在此範例中，使用您自己的資訊取代 `##### CloudWatch ##### Amazon Resource Name (ARN)`。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:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  },
  "CloudWatchLoggingOptions": [{
    "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add to the new application>"
  }]
}
```

將 CloudWatch 日誌選項新增至現有應用程式

下列範例示範如何使用 `AddApplicationCloudWatchLoggingOption` 動作將 CloudWatch 日誌選項新增至現有的應用程式。在下列範例中，使用您自己的資訊取代每個 `#####`。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 日誌選項

下列範例示範如何使用 `UpdateApplication` 動作修改現有的 CloudWatch 日誌選項。在下列範例中，使用您自己的資訊取代每個#####。如需該動作的詳細資訊，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 日誌選項

下列範例示範如何使用 `DeleteApplicationCloudWatchLoggingOption` 動作刪除現有的 CloudWatch 日誌選項。在下列範例中，使用您自己的資訊取代每個#####。如需該動作的詳細資訊，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級”](#)。

建立應用程式時設定應用程式記錄層級

[CreateApplication](#) 動作的下列範例請求會將應用程式日誌層級設定為 INFO。

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My Application Description",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "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 日誌串流的許可

Managed Service for Apache Flink 需要將組態錯誤寫入 CloudWatch 的許可。您可以將這些許可新增至 Managed Service for Apache Flink 擔任的 AWS Identity and Access Management (IAM) 角色。

如需將 IAM 角色用於 Managed Service for Apache Flink 的詳細資訊，請參閱[Amazon Managed Service for Apache Flink 的身分和存取管理](#)。

信任政策

若要授與 Managed Service for Apache Flink 許可以擔任 IAM 角色，您可以將以下信任政策連接到服務執行角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

許可政策

若要向應用程式授與許可，以便從 Managed Service for Apache Flink 資源將日誌事件寫入 CloudWatch，您可以使用下列 IAM 許可政策。為日誌群組和串流提供正確的 Amazon Resource Name (ARN)。

JSON

```
{
  "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級

您可以使用應用程式監控指標層級和監控日誌層級來控制應用程式日誌訊息的產生。

應用程式的監控指標層級會控制日誌訊息的精細程度。監控指標層級的定義如下：

- 應用程式：指標的適用範圍是整個應用程式。
- 任務：指標的適用範圍是每個任務。如需任務的相關資訊，請參閱 [the section called “實作應用程式擴展”](#)。
- 運算子：指標的適用範圍是每個運算子。如需運算子的相關資訊，請參閱 [the section called “運算子”](#)。
- 平行處理層級：指標的適用範圍是應用程式平行處理層級。您只能使用 [UpdateApplication](#) API 的 [MonitoringConfigurationUpdate](#) 參數來設定此指標層級。您無法使用控制台來設定此指標層級。如需平行處理層級的相關資訊，請參閱 [the section called “實作應用程式擴展”](#)。

應用程式的監控日誌層級會控制應用程式日誌的詳細程度。監控日誌層級的定義如下：

- 錯誤：應用程式的潛在災難性事件。
- 警告：應用程式的可能有害的情況。
- 資訊：應用程式的資訊和暫時性故障事件。建議您使用此記錄層級。

- 偵錯：對於應用程式偵錯最有用的精細資訊事件。注意：此層級僅適用於暫時偵錯之目的。

套用記錄最佳實務

我們建議您的應用程式使用資訊記錄層級。我們建議您使用此層級以確保看到 Apache Flink 錯誤，這些錯誤記錄在資訊層級而非錯誤層級。

我們建議您只在調查應用程式問題時暫時使用偵錯層級。問題解決後，切換回資訊層級。使用偵錯記錄層級將顯著影響應用程式的效能。

過多的記錄也會大幅影響應用程式效能。例如，建議您不要為每筆處理的記錄寫入日誌項目。過多的記錄可能會導致嚴重的資料處理瓶頸，並可能導致從來源讀取資料時產生背壓。

執行記錄疑難排解

如果應用程式日誌未寫入日誌串流，請驗證下列項目：

- 確認應用程式的 IAM 角色和政策正確無誤。您的應用程式政策需要下列許可才能存取日誌串流：
 - logs:PutLogEvents
 - logs:DescribeLogGroups
 - logs:DescribeLogStreams

如需詳細資訊，請參閱 [the section called “新增寫入 CloudWatch 日誌串流的許可”](#)。

- 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。
- 監控 CloudWatch 指標 (例如 downtime)，以診斷其他應用程式問題。如需讀取 CloudWatch 指標的相關資訊，請參閱 [Managed Service for Apache Flink 中的指標和維度](#)。

使用 CloudWatch Logs Insights

在應用程式中啟用 CloudWatch 記錄後，您可以使用 CloudWatch Logs Insights 來分析應用程式日誌。如需詳細資訊，請參閱 [the section called “使用 CloudWatch Logs Insights 分析日誌”](#)。

使用 CloudWatch Logs Insights 分析日誌

如上一節所述，將 CloudWatch 記錄選項新增至應用程式後，您可以使用 CloudWatch Logs Insights 查詢您的日誌串流中是否有特定事件或錯誤。

CloudWatch Logs Insights 可讓您以互動方式搜尋和分析 CloudWatch Logs 中的日誌資料。

如需 CloudWatch Logs Insights 使用入門的相關資訊，請參閱[使用 CloudWatch Logs Insights 分析日誌資料](#)。

執行範例查詢

本節描述了如何執行 CloudWatch Logs Insights 查詢範例。

先決條件

- 在 CloudWatch Logs 中設定的現有日誌群組和日誌串流。
- 存放在 CloudWatch Logs 中的現有日誌。

如果您使用 服務 AWS CloudTrail，例如 Amazon Route 53 或 Amazon VPC，您可能已經設定這些服務的日誌以前往 CloudWatch Logs。如需將日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱[CloudWatch Logs 入門](#)。

CloudWatch Logs Insights 中的查詢會從日誌事件傳回一組欄位，或在日誌事件上執行的數學彙總或其他運算的結果。本節示範的查詢會傳回日誌事件清單。

執行 CloudWatch Logs Insights 範例查詢

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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。

選取日誌群組時，CloudWatch Logs Insights 會自動偵測日誌群組中資料內的欄位，並在右側窗格的 Discovered fields (已探索欄位) 中顯示。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 Logs Insights 查詢的詳細資訊，請參閱[執行和修改範例查詢](#)。

檢閱範例查詢

本節包含用於分析 Managed Service for Apache Flink 應用程式日誌的 CloudWatch Logs Insights 範例查詢。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

使用您應用程式的「地區」和「帳戶 ID」取代下列查詢範例中的地區 (*us-west-2*)、帳戶 ID (*012345678901*) 和應用程式名稱 (*YourApplication*)。

本主題包含下列章節：

- [分析操作：任務的分佈](#)
- [分析操作：平行處理的變化](#)
- [分析錯誤：存取遭拒](#)
- [分析錯誤：找不到來源或接收器](#)
- [分析錯誤：應用程式任務相關的失敗](#)

分析操作：任務的分佈

下列 CloudWatch Logs Insights 查詢會傳回 Apache Flink 作業管理員在任務管理員之間分配的任務數目。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[實作應用程式擴展](#)。

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

下列 CloudWatch Logs Insights 查詢會傳回指派給每個任務管理員的子任務。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](#)。

分析操作：平行處理的變化

下列 CloudWatch Logs Insights 查詢會傳回應用程式平行處理層級的變更 (例如由於自動擴展)。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 “使用自動擴展”](#)。

```
fields @timestamp, @parallelism  
| filter message like /property: parallelism.default, /  
| parse message "default, *" as @parallelism  
| sort @timestamp asc
```

分析錯誤：存取遭拒

下列 CloudWatch Logs Insights 查詢會傳回 Access Denied 日誌。

```
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 Logs Insights 查詢會傳回 ResourceNotFound 日誌。ResourceNotFound 如果找不到 Kinesis 來源或接收器，會記錄結果。

```
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：應用程式任務相關的失敗

下列 CloudWatch Logs Insights 查詢會傳回應用程式的任務相關失敗日誌。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
```

Managed Service for Apache Flink 中的指標和維度

當 Managed Service for Apache Flink 處理資料來源時，會向 Amazon CloudWatch 報告下列指標和維度。

Flink 2.2 指標變更

Flink 2.2 引入可能會影響監控和警示的指標變更。升級之前，請檢閱下列變更：

- 指標 `fullRestarts` 已移除。請改用 `numRestarts`。
- `uptime` 和 `downtime` 指標已棄用，並將在未來版本中移除。遷移至新的狀態特定指標。
- Kinesis Data Streams 連接器 `bytesRequestedPerFetch 6.0.0` 的指標已移除。

應用程式指標

指標	單位	Description	Level	使用須知
backPressuredTimeMsPerSecond*	毫秒	此任務或運算子每秒承受背壓的時間 (毫秒)。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。
busyTimeMsPerSecond*	毫秒	此任務或運算子每秒忙碌 (既不是閒置也沒有背壓) 的時間 (毫秒)。如果無法計算該值，則可以是 NaN。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。
cpuUtilization	百分比	跨任務管理員的 CPU 使用率整體百分比。例如，如果有 5 個任務管理員，Managed Service for 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 CPU 使用率。CPUUtilization 指標只考慮在容器內執行之 TaskManager

指標	單位	Description	Level	使用須知	
				JVM 處理序的 CPU 使用率。	
container CPUUtilization	百分比	Flink 應用程式叢集中跨任務管理員容器的 CPU 使用率整體百分比。例如，如果有 5 個任務管理員，則相應地有 5 個任務管理員容器，Managed Service for Apache Flink 會每 1 分鐘的報告間隔發佈 2* 5 個此指標樣本。	應用程式	它按每個容器計算如下： $\frac{\text{容器使用的總 CPU 時間 (秒)}}{\text{CPU 限制 (單位為 CPU/秒)}} * 100$ CPUUtilization 指標只考慮在容器內執行之 TaskManager JVM 處理序的 CPU 使用率。同一個容器內的 JVM 外部還有其他元件在執行。container CPUUtilization 指標為您提供了更完整的視角，包括容器中 CPU 耗盡的所有處理序以及由此引起的故障。	

指標	單位	Description	Level	使用須知	
container 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 5 個任務管理員，則相應地有 5 個任務管理員容器，Managed Service for Apache Flink 會每 1 分鐘的報告間隔發佈 2* 5 個此指標樣本。	應用程式	它按每個容器計算如下： 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) HeapMemoryUtilization 和 ManagedMemoryUtilizations 指標僅考慮特定的記憶體指標，例如 TaskManager JVM 的堆積記憶體使用情況或受管記憶體 (例如用於 RocksDB 狀態後端 等原生處理序的 JVM 外記憶體使用情況)。 containerMemoryUtilization 指標可為您提供更完整的視角，現包	

指標	單位	Description	Level	使用須知	
				括工作集記憶體，這是一個更佳的記憶體總量耗盡追蹤器。在其耗盡之後，它將導致任務管理員 Pod Out of Memory Error。	
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 5 個任務管理員，則相應地有 5 個任務管理員容器，Managed Service for Apache Flink 會每 1 分鐘的報告間隔發佈 2* 5 個此指標樣本。	應用程式	它按每個容器計算如下： 磁碟使用量 (位元組) * 100 / 容器的磁碟限制 (位元組) 對於容器，它代表在其上設定容器根磁碟區的檔案系統的使用率。	
currentInputWatermark	毫秒	此應用程式/運算子/任務/執行緒收到的最後一個浮水印	應用程式、運算子、任務、平行處理層級	僅針對具有兩個輸入的維度發出此記錄。這是最後接收到的浮水印的最小值。	

指標	單位	Description	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 【已棄用】	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。 Flink 2.2 中已棄用。請 failingTime 改用 cancellingTime、restartingTime 和/或。	

指標	單位	Description	Level	使用須知	
failingTime	毫秒	應用程式處於失敗狀態所花費的時間 (以毫秒為單位)。使用此指標來監控應用程式失敗和觸發警示。	應用程式、流程	可從 Flink 2.2 取得。取代部分已棄用downtime指標。	
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。HeapMemoryUtilization 只考慮特定記憶體指標，如 TaskManager JVM 的堆積記憶體使用情況。	
idleTimeMsPerSecond*	毫秒	此任務或運算子每秒閒置 (沒有要處理的資料) 的時間 (毫秒)。閒置時間不包括背壓時間，因此如果任務受到背壓，則不會閒置。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。	

指標	單位	Description	Level	使用須知	
lastCheck pointSize	位元組	最後一個檢查 點的大小總計	應用程式	您可以使用此指標判斷執行中應用程式的儲存體使用率。 如果此指標的值增加，可能表示應用程式發生了問題，例如記憶體流失或瓶頸。	
lastCheck pointDura tion	毫秒	完成最後一個 檢查點所花費 的時間	應用程式	此指標會測量完成最新檢查點所花費的時間。如果此指標的值增加，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。	

指標	單位	Description	Level	使用須知	
managedMemoryUsed*	位元組	目前使用中的受管記憶體數量。	應用程式、運算子、任務、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，也可用於應用程式。	

指標	單位	Description	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	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，也可用於應用程式。ManagedMemoryUtilizations 指標僅考慮特定的記憶體指標，例如受管記憶體 (用於 RocksDB 狀態後端 等原生處理序的 JVM 外記憶體使用情況)	

指標	單位	Description	Level	使用須知	
managedMemoryUtilization*	百分比	由 managedMemoryUsed/managedMemoryTotal 所衍生	應用程式、運算子、任務、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，也可用於應用程式。	
numberOfFailedCheckpoints	計數	檢查點失敗的次數。	應用程式	您可以使用此指標來監控應用程式運作狀態和進度。檢查點可能會因為應用程式問題 (例如輸送量或許可問題) 而失敗。	

指標	單位	Description	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	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：$m1/4$，其中 $m1$ 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	Description	Level	使用須知	
				特定任務接收的記錄總數。	

指標	單位	Description	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	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：$m1/4$，其中 $m1$ 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	Description	Level	使用須知	
				特定任務每秒接收的記錄總數。	

指標	單位	Description	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	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：$m1/4$，其中 $m1$ 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	Description	Level	使用須知	
				特定任務發出的記錄總數。	
numLateRecordsDropped*	計數	應用程式、運算子、任務、平行處理層級		* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：$m1/4$，其中 $m1$ 是一段期間 (秒/分鐘) 內的 SUM 統計資料 此運算子或任務因遲到而丟棄的記錄數。	

指標	單位	Description	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	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：$m1/4$，其中 $m1$ 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	Description	Level	使用須知	
				特定任務每秒發出的記錄總數。	
oldGenerationGCCount	計數	所有任務管理員中發生的垃圾回收操作總數。	應用程式		
oldGenerationGCTime	毫秒	執行垃圾回收操作所花費的總時間。	應用程式	您可以使用此指標來監控總計、平均和最大垃圾回收時間。	
threadsCount	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。	
cancellingTime	毫秒	應用程式處於取消狀態所花費的時間（以毫秒為單位）。使用此指標來監控應用程式取消操作。	應用程式、流程	可從 Flink 2.2 取得。取代部分已棄用downtime指標。	

指標	單位	Description	Level	使用須知	
restartin gTime	毫秒	應用程式處於重新啟動狀態所花費的時間（以毫秒為單位）。使用此指標來監控應用程式重新啟動行為。	應用程式、流程	可從 Flink 2.2 取得。取代部分已棄用downtime指標。	
runningTi me	毫秒	應用程式在未中斷的情況下執行的時間（以毫秒為單位）。取代已取代的uptime指標。	應用程式、流程	可從 Flink 2.2 取得。使用 直接取代已取代的uptime指標。	
uptime 【已棄用】	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。 Flink 2.2 中已棄用。請改用 runningTime 。	
jobmanage rFileDesc riptorsMa x	計數	JobManager 可用的檔案描述項數目上限。	應用程式、流程、主機	使用此指標來監控檔案描述項容量。	

指標	單位	Description	Level	使用須知	
jobmanagerFileDescriptorsOpen	計數	JobManager 目前開啟的檔案描述項數量。	應用程式、流程、主機	使用此指標來監控檔案描述項用量，並偵測潛在的資源耗盡。	
taskmanagerFileDescriptorsMax	計數	每個 TaskManager 可用的檔案描述項數量上限。	應用程式、流程、主機、tm_id	使用此指標來監控檔案描述項容量。	
taskmanagerFileDescriptorsOpen	計數	每個 TaskManager 的目前開啟檔案描述項數量。	應用程式、流程、主機、tm_id	使用此指標來監控檔案描述項用量，並偵測潛在的資源耗盡。	

指標	單位	Description	Level	使用須知
KPUs *	計數	應用程式使用的 KPUs 總數。	應用程式	*此指標每個計費期間（一小時）會收到一個範例。若要視覺化一段時間內的 KPUs 數量，請在至少一 (1) 小時內使用 MAX 或 AVG。 KPU 計數包含 orchestration KPU。如需詳細資訊，請參閱 Managed Service for Apache Flink 定價。

Flink 2.2 指標遷移指引

從 fullRestarts 遷移：指標已在 Flink fullRestarts 2.2 中移除。請改用 numRestarts 指標。numRestarts 指標提供同等功能，可以直接取代 CloudWatch 警示，無需調整閾值。

從執行時間遷移：在 Flink 2.2 中已棄用 uptime 指標，並將在未來版本中移除。請改用 runningTime 指標。runningTime 指標提供同等功能，可以直接取代 CloudWatch 警示，無需調整閾值。

從停機時間遷移：在 Flink 2.2 中已棄用 downtime 指標，並將在未來版本中移除。根據您想要監控的內容，使用下列一或多個指標：

- restartingTime：監控重新啟動應用程式所花費的時間
- cancellingTime：監控取消應用程式所花費的時間

- `failingTime` : 監控處於失敗狀態所花費的時間

Kinesis Data Streams 連接器指標

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	Description	Level	使用須知
<code>millisBehindLatest</code>	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 (用於串流)，平行處理層級 (用於 <code>ShardId</code>)	• 值為 0 表示記錄處理已跟上進度，此時沒有任何新記錄可供處理。可以使用串流名稱和碎片 ID 指定特定碎片的指標。 • 值 -1 表示服務尚未報告指標的值。

Note

指標 `bytesRequestedPerFetch` 已在 Flink AWS 連接器 6.0.0 版（唯一與 Flink 2.2 相容的連接器版本）中移除。Flink 2.2 中唯一可用的 Kinesis Data Streams 連接器指標是 `millisBehindLatest`。

Amazon MSK 連接器指標

AWS 除了下列項目之外，還會發出 Amazon MSK 的所有記錄：

指標	單位	Description	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)、平行處理層級 (針對 Partition Id)	
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	
committedoffsets	N/A	每個分割區最後一次成功提交到 Kafka 的位移。您可以依據主題名稱和分割區 ID 來指定特定分割區的指標。	應用程式 (針對主題)、平行處理層級 (針對 Partition Id)	
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	

指標	單位	Description	Level	使用須知
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	

Apache Zeppelin 指標

對於 Studio 筆記本，在應用程式層級 AWS 發出下列指

標：KPU、cpuUtilization、heapMemoryUtilization、oldGenerationGCTime、oldGenerationGCCount 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	Description	Prometheus 名稱
zeppelinCpuUtilization	百分比	Apache Zeppelin 伺服器中 CPU 使用率的整體百分比。	process_cpu_usage
zeppelinHeapMemoryUtilization	百分比	Apache Zeppelin 伺服器的堆積記憶體使用率整體百分比。	jvm_memory_used_bytes
zeppelinThreadCount	計數	Apache Zeppelin 伺服器使用的即時執行緒總數。	jvm_threads_live_threads
zeppelinWaitingJobs	計數	等待執行緒的已排入佇列的 Apache Zeppelin 作業數目。	jetty_threads_jobs
zeppelinServerUptime	秒鐘	伺服器啟動並執行的總時間。	process_uptime_seconds

檢視 CloudWatch 指標

您可以使用 Amazon CloudWatch 主控台或 AWS CLI 來檢視應用程式的 CloudWatch 指標。

使用 CloudWatch 主控台檢視指標

1. 在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 指標。
3. 在 Managed Service for Apache Flink 的依類別分類的 CloudWatch 指標窗格中，選擇指標類別。
4. 在上方窗格中，向下捲動以檢視完整指標清單。

使用 檢視指標 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 指標報告層級

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 CLI 來設定此指標層級。此指標層級在主控台中無法使用。

預設層級為應用程式。應用程式會報告目前層級和所有更高層級的指標。例如，如果報告層級設定為運算子，則應用程式會報告應用程式、任務和運算子指標。

您可以使用 [CreateApplication](#) 動作的 `MonitoringConfiguration` 參數或 [UpdateApplication](#) 動作的 `MonitoringConfigurationUpdate` 參數來設定 CloudWatch 指標報告層級。[UpdateApplication](#) 動作的下列範例請求會將 CloudWatch 指標報告層級設定為任務：

```
{
  "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 Amazon Managed Service for Apache Flink 使用自訂指標

Managed Service for Apache Flink 向 CloudWatch 公開了 19 個指標，包括資源使用率和輸送量的指標。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](#)
- [檢視建立映射類別的範例](#)
- [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 Metrics 不支援直方圖 Apache Flink 指標類型。CloudWatch 只顯示「計數」、「量計」和「計量」類型的 Apache Flink 指標。

- **範圍**：指標的範圍包含其識別碼和一組指示如何向 CloudWatch 報告指標的鍵值對。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 [MetricGroup](#) 物件，您可以用它來建立和註冊自訂指標。Managed Service for Apache Flink 會將使用群組索引鍵 KinesisAnalytics 建立的所有指標報告給 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。這些名稱必須包含根據 [Prometheus 命名規則](#) 的英數字元。
- 您在使用者範圍中定義的屬性 (KinesisAnalytics 指標群組除外) 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 [CreateApplication](#) 動作的 [MonitoringConfiguration](#) 參數或 [UpdateApplication](#) 動作的 [MonitoringConfigurationUpdate](#) 參數來設定應用程式的監控層級。

檢視建立映射類別的範例

下列程式碼範例示範如何建立並遞增自訂指標的映射類別，以及如何透過將映射類別新增至DataStream物件，在應用程式中實作映射類別。

記錄計數自訂指標

下列程式碼範例示範如何建立映射類別，以建立可計算資料串流中記錄數目的指標 (功能與 numRecordsIn 指標相同)：

```
private static class NoOpMapperFunction extends RichMapFunction<String, String> {
    private transient int valueToExpose = 0;
    private final String customMetricName;

    public NoOpMapperFunction(final String customMetricName) {
        this.customMetricName = customMetricName;
    }

    @Override
    public void open(Configuration config) {
        getRuntimeContext().getMetricGroup()
            .addGroup("KinesisAnalytics")
            .addGroup("Program", "RecordCountApplication")
            .addGroup("NoOpMapperFunction")
            .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
    }

    @Override
    public String map(String value) throws Exception {
        valueToExpose++;
        return value;
    }
}
```

在上述範例中，valueToExpose 變數會針對應用程式處理的每筆記錄遞增。

定義映射類別之後，您可以建立實作對應的應用程式內串流：

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

如需此應用程式的完整程式碼，請參閱[記錄計數自訂指標應用程式](#)。

單字計數自訂指標

下列程式碼範例示範如何建立映射類別，以建立可計算資料串流中字數的指標：

```
private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String, Integer>> {

    private transient Counter counter;

    @Override
    public void open(Configuration config) {
        this.counter = getRuntimeContext().getMetricGroup()
            .addGroup("KinesisAnalytics")
            .addGroup("Service", "WordCountApplication")
            .addGroup("Tokenizer")
            .counter("TotalWords");
    }

    @Override
    public void flatMap(String value, Collector<Tuple2<String, Integer>>out) {
        // normalize and split the line
        String[] tokens = value.toLowerCase().split("\\W+");

        // emit the pairs
        for (String token : tokens) {
            if (token.length() > 0) {
                counter.inc();
                out.collect(new Tuple2<>(token, 1));
            }
        }
    }
}
```

在上述範例中，counter 變數會針對應用程式處理的每個單字遞增。

定義映射類別之後，您可以建立實作對應的應用程式內串流：

```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
    Tokenizer()).keyBy(0).sum(1);

// Serialize the tuple to string format, and publish the output to kinesis sink
```

```
wordCountStream.map(tuple -> tuple.toString()).addSink(createSinkFromStaticConfig());
```

如需此應用程式的完整程式碼，請參閱[單字計數自訂指標應用程式](#)。

檢視自訂指標

應用程式的自訂指標會顯示在 CloudWatch 指標主控台 AWS/KinesisAnalytics 儀表板的應用程式指標群組下。

搭配 Amazon Managed Service for Apache Flink 使用 CloudWatch 警示

使用 Amazon CloudWatch 指標警示，您可在自己指定的時段內監看 CloudWatch 指標。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。某個動作將通知傳送至 Amazon Simple Notification Service (Amazon SNS) 主題的範例。

如需 CloudWatch 警示的詳細資訊，請參閱[使用 Amazon CloudWatch 警示](#)。

檢閱建議的警示

本節包含用於監控 Managed Service for Apache Flink 應用程式的建議警示。

下表說明了建議的警示，其中包含下列欄位：

- 指標表達式：根據閾值測試的指標或指標表示式。
- 統計值：用來檢查指標的統計值 — 例如平均值。
- 閾值：使用此警示會要求您決定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	Description
downtime > 0	Average	0	A downtime greater than zero indicates that the application has failed. If the value is larger than 0, the application is not processing any data.

指標表達式	統計數字	Threshold	Description
			Recommended for all applications. The ## metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see 應用程式正在重新啟動 .

指標表達式	統計數字	Threshold	Description
RATE (numberOfFailedCheckpoints) > 0	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring RATE(numberOfFailedCheckpoints) to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress in processing the input

指標表達式	統計數字	Threshold	Description
Operator. numRecordsOutPerSecond threshold	Average	The minimum number of records emitted from the application during normal conditions.	data. For troubleshooting, see 檢查點逾時 . Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see 輸送量太慢 .

指標表達式	統計數字	Threshold	Description
<code>records_lag_max_millisbehindLatest > threshold</code>	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the <code>records_lag_max</code> metric for a Kafka source, or the <code>millisbehindLatest</code> for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see 輸送量太慢 .

指標表達式	統計數字	Threshold	Description
<code>lastCheckpointDuration > threshold</code>	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>RATE(lastCheckpointSize)</code> and <code>RATE(lastCheckpointDuration)</code> . If the <code>lastCheckpointDuration</code> continuously increases, rising above this threshold can indicate that the application isn't making expected

指標表達式	統計數字	Threshold	Description
			progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see 無限制狀態成長 .

指標表達式	統計數字	Threshold	Description
<code>lastCheckpointSize > threshold</code>	Maximum	The maximum expected checkpoint size during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>RATE(lastCheckpointSize)</code> and <code>RATE(lastCheckpointDuration)</code> . If the <code>lastCheckpointSize</code> continuously increases, rising above this threshold can indicate that the application is accumulating state

指標表達式	統計數字	Threshold	Description
			data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troubleshooting, see 無限制狀態成長 .
heapMemoryUtilization > threshold	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected heapMemoryUtilization size during normal conditions, with a recommended value of 90 percent.	You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources . You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources , see 實作應用程式擴展 .

指標表達式	統計數字	Threshold	Description
<code>cpuUtilization > threshold</code>	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected <code>cpuUtilization</code> size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources. You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see 實作應用程式擴展 .
<code>threadsCount > threshold</code>	Maximum	The maximum expected <code>threadsCount</code> size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

指標表達式	統計數字	Threshold	Description
<code>(oldGarbageCollectionTime * 100) / 60_000 over 1 min period') > threshold</code>	Maximum	The maximum expected oldGarbageCollectionTime duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold , but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
<code>RATE(oldGarbageCollectionCount) > threshold</code>	Maximum	The maximum expected oldGarbageCollectionCount under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
<code>Operator.currentOutputWatermark - Operator.currentInputWatermark > threshold</code>	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.

將自訂訊息寫入 CloudWatch Logs

您可以將自訂訊息寫入 Managed Service for Apache Flink 應用程式的 CloudWatch 日誌。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 SLF4J 寫入 CloudWatch 日誌](#)

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 SLF4J 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.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.7</version>
  <scope>runtime</scope>
</dependency>
```

2. 包括來自程式庫的物件：

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

3. 具現化 Logger 物件，傳入你的應用程式類別：

```
private static final Logger log =
  LoggerFactory.getLogger(YourApplicationClass.class);
```

4. 使用 log.info 寫入日誌。大量訊息會寫入應用程式日誌。若要讓您的自訂訊息更易於篩選，請使用 INFO 應用程式日誌層級。

```
log.info("This message will be written to the application's CloudWatch log");
```

應用程式會將記錄寫入日誌，並顯示類似如下的訊息：

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

使用 記錄 Managed Service for Apache Flink API 呼叫 AWS CloudTrail

Managed Service for Apache Flink 已與 整合 AWS CloudTrail，此服務提供使用者、角色或 Managed Service for Apache Flink 中 AWS 服務所採取動作的記錄。CloudTrail 會將 Managed Service for Apache Flink 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 Managed Service for Apache Flink 主控台的呼叫，以及對 Managed Service for Apache Flink API 操作的程式碼呼叫。如果您建立追蹤，就可以將 CloudTrail 事件持續交付到 Amazon S3 儲存貯體，包括 Managed Service for Apache Flink 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台的事件歷史記錄檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Managed Service for Apache Flink 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「[AWS CloudTrail 使用者指南](#)」。

CloudTrail 中的 Managed Service for Apache Flink 資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 Managed Service for Apache Flink 中發生時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html>。

若要持續記錄您 AWS 帳戶中的事件，包括 Managed Service for Apache Flink 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台中建立線索時，線索會套用至所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務和整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [從多個區域接收 CloudTrail 日誌檔案](#)，以及[從多個帳戶接收 CloudTrail 日誌檔案](#)

CloudTrail 會記錄所有 Managed Service for Apache Flink 動作，這些動作也會記載在 [Managed Service for Apache Flink API 參考](#)中。例如，對 [CreateApplication](#) 以及 [UpdateApplication](#) 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

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：

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 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](#)。

了解 Managed Service for Apache Flink 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例為展示了 [AddApplicationCloudWatchLoggingOption](#) 和 [DescribeApplication](#) 的 CloudTrail 日誌項目。

```
{
  "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"
  }
]
```

在 Amazon Managed Service for Apache Flink 中調校效能

本主題說明可監控和改善 Managed Service for Apache Flink 應用程式效能的技術。

主題

- [對效能問題進行故障診斷](#)
- [使用效能最佳實務](#)
- [監控效能](#)

對效能問題進行故障診斷

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。

了解資料路徑

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：確保應用程式互動的外部資源已正確佈建，以達到應用程式將體驗的輸送量。
- 狀態資料：確保應用程式不會太頻繁地與狀態存放區互動。

您可以最佳化應用程式正在使用的序列化程式。預設的 Kryo 序列化程式可以處理任何可序列化類型，但是如果應用程式只將資料存儲在 POJO 類型中，則可以使用更高性能的序列化程式。如需 Apache Flink 序列化程式的相關資訊，請參閱 Apache Flink 文件中的[資料類型和序列化](#)。

- 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。

效能疑難排解解決方案

本節包含效能問題的潛在解決方案。

主題

- [CloudWatch 監控層級](#)

- [應用程式 CPU 指標](#)
- [應用程式平行處理](#)
- [應用程式日誌記錄](#)
- [運算子平行處理](#)
- [應用程式邏輯](#)
- [應用程式記憶體](#)

CloudWatch 監控層級

確認未將 CloudWatch 監控層級設定得太詳細。

Debug 監控日誌層級設定會產生大量的流量，這會造成背壓。您只能在主動調查應用程式問題時使用它。

如果您的應用程式具有較高的 Parallelism 設定值，使用 Parallelism 監控指標層級也會產生大量流量，進而導致背壓。只有在您的應用程式 Parallelism 不足或調查應用程式問題時，才使用此指標層級。

如需詳細資訊，請參閱[控制應用程式監控層級](#)。

應用程式 CPU 指標

檢查應用程式的 CPU 指標。如果此指標高於 75%，您可以啟用自動擴展，允許應用程式為自己配置更多資源。

如果啟用了自動擴展，則當 CPU 使用率超過 75% 並持續 15 分鐘時，應用程式會配置更多資源。如需擴展的詳細資訊，請參閱下面的[適當管理擴展](#)一節和[實作應用程式擴展](#)。

Note

應用程式只會根據 CPU 使用率自動擴展。應用程式不會自動擴展以回應其他系統指標，例如 heapMemoryUtilization。如果應用程式在其他指標上具有較高的使用量，請手動增加應用程式的平行處理層級。

應用程式平行處理

增加應用程式的平行處理層級。您可以使用 [UpdateApplication](#) 動作的 ParallelismConfigurationUpdate 參數來更新應用程式的平行處理層級。

應用程式的最大 KPU 預設為 64 個，可透過請求提高限制來增加。

務必基於每個運算子的工作負載指派運算子的平行處理層級，而不僅僅是單獨增加應用程式平行處理層級。請參閱下文的[運算子平行處理](#)。

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 CloudWatch Logs Insights 分析日誌”](#)。

運算子平行處理

確認應用程式的工作負載在工作者處理序之間平均分配。

如需調整應用程式運算子工作負載的相關資訊，請參閱[運算子擴展](#)。

應用程式邏輯

檢查您的應用程式邏輯是否有效率低或性能不佳的操作，例如存取外部相依性 (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](#)。

應用程式記憶體

檢查應用程式是否有資源洩漏。如果應用程式未正確處置執行緒或記憶體，您可能會看到 `millisBehindLatest`、`CheckpointSize` 和 `CheckpointDuration` 指標急遽增加或逐漸增加。此情況還可能導致任務管理員或作業管理員失敗。

使用效能最佳實務

本節說明針對效能設計應用程式時的特殊考量。

適當管理擴展

本節包含管理應用程式層級和運算子層級擴展的相關資訊。

本節包含下列主題：

- [適當管理應用程式擴展](#)
- [適當管理運算子擴展](#)

適當管理應用程式擴展

您可以使用自動擴展來處理應用程式活動中的意外尖峰。如果符合下列條件，應用程式的 KPU 會自動增加：

- 已為應用程式啟用自動擴展。
- CPU 使用率在 15 分鐘內保持在 75% 以上。

如果已啟用自動擴展，但 CPU 使用率未維持在此閾值，則應用程式將不會縱向擴展 KPU。如果您遇到不符合此閾值的 CPU 使用率尖峰，或是不同使用量指標 (例如 `heapMemoryUtilization`) 中出現尖峰的情況，請手動增加擴展以允許應用程式處理活動尖峰。

Note

如果應用程式透過自動擴展自動新增了更多資源，則會在閒置一段時間後釋出新資源。縮減資源會暫時影響效能。

如需擴展的詳細資訊，請參閱[實作應用程式擴展](#)。

適當管理運算子擴展

您可以透過驗證應用程式的工作負載在工作者處理序之間平均分配，以及應用程式中的運算子擁有穩定且高效能狀態所需的系統資源，藉此改善應用程式的效能。

您可以使用 `parallelism` 設定為應用程式程式碼中的每個運算子設定平行處理層級。如果您未為運算子設定平行處理層級，它就會使用應用程式層級的平行處理設定。使用應用程式層級平行處理設定的運算子可能會使用應用程式可用的所有系統資源，這會導致應用程式不穩定。

為了準確確定每個運算子的平行處理層級，請考慮該運算子與應用程式中其他運算子相比的相對資源需求。對比耗費更少資源的運算子，為耗費更多資源的運算子設定更高的運算子平行處理設定。

應用程式的運算子平行處理層級總數是應用程式中所有運算子的平行處理層級之總和。您可以決定應用程式的運算子平行處理層級總數與應用程式可用的任務空位總數之間的最佳比率，以調整應用程式的運算子平行處理層級總數。運算子平行處理層級與任務空位的典型穩定比率為 4:1，也就是說，應用程式

每四個可用的運算子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 [使用執行期屬性](#) 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 Managed Service for Apache Flink 應用程式資源](#)。

若要控制整個應用程式工作者處理序的工作負載分配，請使用 Parallelism 設定和 KeyBy 分割方法。如需詳細資訊，請參閱 [Apache Flink 文件](#) 中的下列主題：

- [平行執行](#)
- [DataStream 轉換](#)

監控外部相依性資源使用量

如果目的地（例如 Kinesis Streams、Firehose、DynamoDB 或 OpenSearch Service）發生效能瓶頸，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故障，請檢查目的地服務的 CloudWatch 日誌查看故障。

在本機執行 Apache Flink 應用程式

若要疑難排解記憶體問題，您可以在本機 Flink 安裝中執行應用程式。這可讓您存取堆疊追蹤和堆積傾印等偵錯工具，在 Managed Service for Apache Flink 中執行應用程式時，這些工具不可用。

如需有關建立本機 Flink 安裝的資訊，請參閱 Apache Flink 文件中的[獨立](#)。

監控效能

本節說明監控應用程式效能的工具。

使用 CloudWatch 指標監控效能

您可以使用 CloudWatch 指標監控應用程式的資源使用量、輸送量、檢查點和停機時間。如需將 CloudWatch 指標與 Managed Service for Apache Flink 應用程式搭配使用的相關資訊，請參閱[Managed Service for Apache Flink 中的指標和維度](#)。

使用 CloudWatch 日誌和警示監控效能

您可以使用 CloudWatch Logs 監控可能導致效能問題的錯誤情況。

當 Apache Flink 作業狀態從 RUNNING 狀態變更為 FAILED 狀態時，日誌項目中會顯示錯誤情況。

您可以使用 CloudWatch 警示來建立效能問題的通知，例如資源使用或檢查點指標超出安全閾值，或未預期的應用程式狀態變更。

如需為 Managed Service for Apache Flink 應用程式建立 CloudWatch 警示的相關資訊，請參閱[搭配 Amazon Managed Service for Apache Flink 使用 CloudWatch 警示](#)。

Managed Service for Apache Flink 和 Studio 筆記本配額

Note

Apache Flink 社群超過三年不支援 Apache Flink 1.6、1.8 和 1.11 版。我們現在計劃結束對 Amazon Managed Service for Apache Flink 中這些版本的支援。從 2024 年 11 月 5 日起，您將無法為這些 Flink 版本建立新的應用程式。此時您可以繼續執行現有的應用程式。

對於除中國區域和 以外的所有區域 AWS GovCloud (US) Regions，從 2025 年 2 月 5 日起，您將無法再在 Amazon Managed Service for Apache Flink 中使用這些版本的 Apache Flink 建立、啟動或執行應用程式。

對於中國區域和 AWS GovCloud (US) Regions，從 2025 年 3 月 19 日起，您將無法再在 Amazon Managed Service for Apache Flink 中使用這些版本的 Apache Flink 建立、啟動或執行應用程式。

您可以使用 Managed Service for Apache Flink 中的就地版本升級功能，以狀態升級應用程式。如需詳細資訊，請參閱[針對 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：

- 您帳戶中每個區域最多可以建立 100 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 支援 中心](#)。

如需支援 Managed Service for Apache Flink 的區域清單，請參閱 [Managed Service for Apache Flink 區域和端點](#)。

- 依預設，Kinesis 處理單元 (KPU) 的數目限制為 64。如需如何請求提高此配額的指示，請參閱 [Service Quotas](#) 中的請求提高配額。請確定您已指定需要套用新 KPU 限制的應用程式前綴。

使用 Managed Service for Apache Flink，您的 AWS 帳戶會支付已配置資源的費用，而不是應用程式使用的資源。我們會根據您執行串流處理應用程式所使用的 KPU 數目上限，以小時費率計費。單一 KPU 可為您提供 1 個 vCPU 和 4 GiB 的記憶體。此服務也會針對每個 KPU 提供 50 GiB 的執行中應用程式儲存體。

- 每個應用程式最多可以建立 1,000 個 Managed Service for Apache Flink 快照。如需詳細資訊，請參閱[使用快照管理應用程式備份](#)。
- 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 Managed Service for Apache Flink 的維護任務

Managed Service for Apache Flink 會使用作業系統和容器映像安全性更新定期修補您的應用程式，以維持合規性並達成 AWS 安全目標。Managed Service for Apache Flink 應用程式的維護時段為 8 小時的時段，在此期間 Managed Service for Apache Flink 會對應用程式執行應用程式維護活動。維護可能會在服務團隊 AWS 區域 排定的不同日期開始。如需維護時段的資訊，請參閱下一節中的表格。

在維護程序中，您的 Managed Service for Apache Flink 應用程式將會重新啟動。這會導致應用程式維護時段的停機時間為 10 到 30 秒。實際停機時間取決於應用程式狀態、大小和快照/檢查點的延遲。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”](#)。您可以了解 Managed Service for Apache Flink 是否已使用 ListApplicationOperations API 對您的應用程式執行維護動作。如需詳細資訊，請參閱[識別您的應用程式何時發生維護](#)。

中的維護時段 AWS 區域

AWS 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)	03:00–11:00 UTC
美國東部 (維吉尼亞北部)	03:00–11:00 UTC
美國東部 (俄亥俄)	03:00–11:00 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)
亞太地區 (香港)	13:00–21:00 UTC
亞太地區 (孟買)	16:30–00:30 UTC
亞太地區 (海德拉巴)	16:30–00:30 UTC
亞太地區 (首爾)	下午 1 時至 9 時 (UTC)
亞太地區 (新加坡)	14:00–22:00 UTC

AWS 區域	維護時段
亞太地區 (悉尼)	12:00–20:00 UTC
亞太地區 (雅加達)	15:00–23:00 UTC
亞太地區 (東京)	下午 1 時至下午 9 時 (UTC)
加拿大 (中部)	03:00–11:00 UTC
中國 (北京)	下午 1 時至下午 9 時 (UTC)
中國 (寧夏)	下午 1 時至下午 9 時 (UTC)
歐洲 (法蘭克福)	上午 6 時至下午 2 時 (UTC)
歐洲 (蘇黎世)	20:00–04:00 UTC
歐洲 (愛爾蘭)	22:00–06:00 UTC
歐洲 (倫敦)	22:00–06:00 UTC
歐洲 (斯德哥爾摩)	23:00–07:00 UTC
歐洲 (米蘭)	21:00–05:00 UTC
歐洲 (西班牙)	下午 9 時至次日上午 5 時 (UTC)
非洲 (開普敦)	20:00–04:00 UTC
歐洲 (愛爾蘭)	22:00–06:00 UTC
歐洲 (倫敦)	下午 11 時至次日上午 7 時 (UTC)
Europe (Paris)	下午 11 時至次日上午 7 時 (UTC)
歐洲 (斯德哥爾摩)	23:00–07:00 UTC
Middle East (Bahrain)	下午 1 時至下午 9 時 (UTC)
中東 (阿拉伯聯合大公國)	18:00–02:00 UTC

AWS 區域	維護時段
南美洲 (聖保羅)	19:00–03:00 UTC
以色列 (特拉維夫)	下午 8 時至次日凌晨 4 時 (UTC)

選擇維護時段

Managed Service for Apache Flink 透過電子郵件和 AWS Health 通知通知您即將規劃的維護事件。在 Managed Service for Apache Flink 中，您可以透過使用 `UpdateApplicationMaintenanceConfiguration` API 並更新您的維護時段組態，來變更維護開始的時間。如需詳細資訊，請參閱 [UpdateApplicationMaintenanceConfiguration](#)。Managed Service for Apache Flink 會在下次排程應用程式的維護時，使用更新的維護組態。如果您在服務已排程維護之後叫用此操作，服務會在下一次排程應用程式的維護時套用組態更新。

Note

為了盡可能提供最高的安全性狀態，Managed Service for Apache Flink 不支援在特定日期選擇退出維護、暫停維護或執行維護的任何例外狀況。

識別您的應用程式何時進行維護

您可以使用 `ListApplicationOperations` API 來尋找 Managed Service for Apache Flink 是否已對您的應用程式執行維護動作。

以下是的範例請求 `ListApplicationOperations`，可協助您篩選應用程式維護的清單：

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

實現 Managed Service for Apache Flink 應用程式的生產準備

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。我們看到應用程式看起來運作狀態良好，但操作事件大幅擴大了應用程式的負載。這可以完全獨立於應用程式本身。如果資料來源或資料接收器無法使用幾個小時，則 Flink 應用程式無法進行進度。當此問題已修正時，會有已累積的未處理資料待處理項目，這可能會完全耗盡可用的資源。然後，負載可以放大之前未出現的錯誤或效能問題。

因此，您必須為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。然後啟動應用程式，並讓它從一開始就取用資料。例如，TRIM_HORIZON 在 Kinesis 資料串流的情況下，使用的開始位置。

定義最大平行處理

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。

在第一次建立狀態時，會設定最大平行處理。

根據預設，最大平行處理設定為：

- 128：如果平行處理層級 ≤ 128
- $\text{MIN}(\text{nextPowerOfTwo}(\text{parallelism} + (\text{parallelism} / 2)), 2^{15})$ ：如果平行處理層級 > 128

如果您打算擴展應用程式 > 128 個平行處理，您應該明確定義最大平行處理。

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。除非另有指定，否則所有運算子都會繼承應用程式的最大平行處理。

如需詳細資訊，請參閱 Apache Flink 文件中的 [設定平行處理上限](#)。

為所有運算子設定 UUID

在 Flink 將儲存點映射回個別運算子的操作中，會使用 UUID。為每個運算子設定特定的 UUID 可以為要還原的儲存點程序提供穩定映射。

```
.map(...).uid("my-map-function")
```

如需詳細資訊，請參閱 [生產就緒性檢查清單](#)。

維護 Managed Service for Apache Flink 應用程式的最佳實務

本節包含開發穩定、高效能 Managed Service for Apache Flink 應用程式的資訊和建議。

主題

- [最小化 uber JAR 的大小](#)
- [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](#)
- [記錄](#)
- [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](#)
- [為所有運算子設定 UUID](#)
- [將 ServiceResourceTransformer 新增至 Maven 著色外掛程式](#)

最小化 uber JAR 的大小

Java/Scala 應用程式必須封裝在 uber（超級/重度）JAR 中，並包含執行時間尚未提供的所有其他必要相依性。不過，uber JAR 的大小會影響應用程式的啟動和重新啟動時間，並可能導致 JAR 超過 512 MB 的限制。

若要最佳化部署時間，您的 uber JAR 不應包含下列項目：

- 執行時間提供的任何相依性，如下列範例所示。它們應該在 POM 檔案或 Gradle 組態 `compileOnly` 中具有 `provided` 範圍。
- 任何僅用於測試的相依性，例如 JUnit 或 Mockito。它們應該在 POM 檔案或 Gradle 組態 `testImplementation` 中具有 `test` 範圍。

- 您的應用程式未實際使用的任何相依性。
- 您的應用程式所需的任何靜態資料或中繼資料。應用程式應在執行時間載入靜態資料，例如從資料存放區或從 Amazon S3 載入。
- 如需上述組態設定的詳細資訊，請參閱此 [POM 範例檔案](#)。

提供的相依性

Managed Service for Apache Flink 執行時間提供許多相依性。這些相依性不應包含在重型 JAR 中，且必須在 POM 檔案中具有 `provided` 範圍，或在 `maven-shade-plugin` 組態中明確排除。在執行時間會忽略包含在重型 JAR 中的任何相依性，但在部署期間增加 JAR 增加額外負荷的大小。

執行時間在執行時間版本 1.18、1.19 和 1.20 中提供的相依性：

- `org.apache.flink:flink-core`
- `org.apache.flink:flink-java`
- `org.apache.flink:flink-streaming-java`
- `org.apache.flink:flink-scala_2.12`
- `org.apache.flink:flink-table-runtime`
- `org.apache.flink:flink-table-planner-loader`
- `org.apache.flink:flink-json`
- `org.apache.flink:flink-connector-base`
- `org.apache.flink:flink-connector-files`
- `org.apache.flink:flink-clients`
- `org.apache.flink:flink-runtime-web`
- `org.apache.flink:flink-metrics-code`
- `org.apache.flink:flink-table-api-java`
- `org.apache.flink:flink-table-api-bridge-base`
- `org.apache.flink:flink-table-api-java-bridge`
- `org.apache.logging.log4j:log4j-slf4j-impl`
- `org.apache.logging.log4j:log4j-api`
- `org.apache.logging.log4j:log4j-core`
- `org.apache.logging.log4j:log4j-1.2-api`

此外，執行期提供程式庫，用於擷取 Managed Service for Apache Flink 中的應用程式執行期屬性 `com.amazonaws:aws-kinesisanalytics-runtime:1.2.0`。

執行時間提供的所有相依性都必須使用下列建議，才不會將其包含在 uber JAR 中：

- 在 Maven (`pom.xml`) 和 SBT (`build.sbt`) 中，使用 `provided` 範圍。
- 在 Gradle (`build.gradle`) 中，使用 `compileOnly` 組態。

由於 Apache Flink 的父系優先類別載入，在執行時間會忽略意外包含在 uber JAR 中的任何提供的相依性。如需詳細資訊，請參閱 Apache Flink 文件中的 [parent-first-patterns](#)。

連接器

除了 `FileSystem` 連接器之外，執行時間中未包含的大多數連接器都必須包含在具有預設範圍 (`()`) 的 POM 檔案中 `compile`。

其他建議

一般而言，提供給 Managed Service for Apache Flink 的 Apache Flink uber JAR 應包含執行應用程式所需的最低程式碼。包含包含來源類別、測試資料集或引導狀態的相依性不應包含在此 jar 中。如果需要在執行時間提取靜態資源，請將此問題分成 Amazon S3 等資源。此範例包括狀態引導或推論模型。

花一些時間考慮您的深層相依性樹狀結構，並移除非執行時間相依性。

雖然 Managed Service for Apache Flink 支援 512MB jar 大小，但這應該視為規則的例外狀況。Apache Flink 目前透過其預設組態支援約 104MB 的 jar 大小，這應該是所需 jar 的目標大小上限。

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：

- 建議您為應用程式保持啟用檢查點。檢查點可在排程維護期間為您的應用程式提供容錯能力，以及因服務問題、應用程式相依性故障和其他問題而導致的非預期故障。如需排程維護的相關資訊，請參閱 [管理 Managed Service for Apache Flink 的維護任務](#)。
- 在應用程式開發或疑難排解期間，將 `ApplicationSnapshotConfiguration::SnapshotsEnabled` 設定為 `false`。每次應用程式停止時都會建立快照，如果應用程式處於運作狀態不佳或效能不佳，這可能會造成問題。當應用程式進入生產環境且狀態穩定之後，將 `SnapshotsEnabled` 設定為 `true`。

Note

我們建議您將應用程式設定為每天建立快照數次，以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。經常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系統資源。

如需監控應用程式停機時間的相關資訊，請參閱 [Managed Service for Apache Flink 中的指標和維度](#)。

如需實作容錯能力的詳細資訊，請參閱 [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，Managed Service for Apache Flink 會在應用程式 JARs 中使用不支援的 Kinesis 連接器版本時，自動防止應用程式啟動或更新。升級至 Managed Service for Apache Flink 1.15 版或更新版本時，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。Managed Service for Apache Flink 不支援所有其他版本，因為它們可能會導致與 Stop with Savepoint 功能的一致性問題或失敗，以防止清除停止/更新操作。若要進一步了解 Amazon Managed Service for Apache Flink 版本中的連接器相容性，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。如果來源和目的地是其他服務 AWS，請使用 [CloudWatch](#) 監控這些服務。
- 對於具有非常高的平行處理層級的應用程式，請檢查該高平行處理層級是否已套用到應用程式中的所有運算子。根據預設，Apache Flink 會對應用程式圖形中的所有運算子套用相同的應用程式平行處理層級。這可能會導致來源或接收器的佈建問題，或導致運算子資料處理出現瓶頸。您可以使用 [setParallelism](#) 來變更程式碼中每個運算子的平行處理層級設定。
- 了解應用程式中運算子平行處理層級設定的意義。如果您變更運算子的平行處理層級，則當運算子的平行處理層級與目前設定不相容時，可能無法從建立的快照還原應用程式。如需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 [實作應用程式擴展](#)。

設定每個運算子的平行處理層級

根據預設，所有運算子都會在應用程式層級設定平行處理層級。您可以使用 DataStream API 和 `.setParallelism(x)` 覆寫單一運算子的平行處理層級。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，請考慮將運算子平行處理傳遞為應用程式組態參數來變更它，而不變更程式碼，例如重新碎片來源串流。

記錄

您可以使用 CloudWatch Logs 來監控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：

- 為應用程式啟用 CloudWatch 記錄，以便對任何執行期問題進行偵錯。
- 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，並可能導致資料處理的背壓。
- 建立 CloudWatch 警示，以在應用程式未正常執行時通知您。如需詳細資訊，請參閱[搭配 Amazon Managed Service for Apache Flink 使用 CloudWatch 警示](#)

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。
- 考慮遮蔽您的應用程式 JAR 檔案及其包含的相依性。如果應用程式與 Apache Flink 執行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。如需遮蔽應用程式 JAR 檔案的詳細資訊，請參閱 [Apache Maven Shade 外掛程式](#)。

管理憑證

您不應將任何長期憑證封裝到生產 (或任何其他) 應用程式中。長期憑證可能簽入版本控制系統，很容易丟失。反之，您可以將角色與 Managed Service for Apache Flink 應用程式建立關聯，並將許可授予該角色。然後，執行中的 Flink 應用程式可以從環境中選取具有相應許可的臨時登入資料。如果未與 IAM 原生整合的服務需要身分驗證，例如需要使用者名稱和密碼進行身分驗證的資料庫，您應該考慮將秘密儲存在 [AWS Secrets Manager](#) 中。

許多 AWS 原生服務支援身分驗證：

- Kinesis Data Streams：[ProcessTaxiStream.java](#)
- Amazon MSK：<https://github.com/aws/aws-msk-iam-auth/#using-the-amazon-msk-library-for-iam-authentication>
- Amazon Elasticsearch Service：[AmazonElasticsearchSink.java](#)
- 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 Data Stream 讀取時，串流的平行處理 (Kafka 的分割區數量和 Kinesis 的碎片數量) 與應用程式的平行處理之間可能不相符。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測試，並在 `events-per-second` 乘以 時獲得最佳效能 `number-of-keys`，低於 25,000,000。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`。有一些實用的內建策略：

- `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。

從 [Flink 文件](#) 中：

來源函數的每個平行子任務通常會獨立生成其浮水印。這些浮水印會定義該特定平行來源的事件時間。

當浮水印流經串流傳輸程序時，它們會將所到達之運算子處的事件時間提前。每當運算子提前其事件時間時，都會為其後續運算子產生新的浮水印。

某些運算子會取用多個輸入串流；例如聯集，或跟隨 `keyBy (...)` 或 `partition (...)` 函數的運算子。這類運算子的目前事件時間是其輸入串流事件時間的最小值。隨著其輸入串流更新其事件時間，運算子的事件時間也會更新。

這意味著，如果來源子任務從閒置碎片中取用，則下游運算子不會從該子任務中收到新的浮水印，因此對使用時間範圍的所有下游運算子進行的處理將停止。為了避免這種情況，客戶可以將 `withIdleness` 選項新增到浮水印策略中。使用此選項時，運算子會在運算運算子的事件時間時，從閒置上游子任務中排除浮水印。因此，閒置子任務不會再封鎖下游運算子中事件時間的進展。

視您使用的碎片指派器而定，某些工作者可能不會獲指派任何 Kinesis 碎片。在這種情況下，即使所有 Kinesis 碎片持續交付事件資料，這些工作者也會呈現閒置來源行為。您可以 `uniformShardAssigner` 搭配來源運算子使用來降低此風險。這可確保只要工作者數目小於或等於作用中碎片數目，所有來源子任務都有要處理的碎片。

不過，如果沒有子任務正在讀取任何事件，而串流中沒有事件，則內建浮水印策略的閒置選項不會延長事件時間。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。由於事件時間在讀取最後一個事件之後不會提前，因此最後一個時段（包含最後一個事件）不會關閉。

摘要

- 如果碎片閒置，此 `withIdleness` 設定不會產生新的浮水印。它會從下游運算子的最低浮水印計算中排除閒置子任務傳送的最後一個浮水印。
- 使用內建浮水印策略時，最後一個開啟的時段不會關閉（除非會傳送推進浮水印的新事件，但會建立一個保持開啟的新時段）。
- 即使 Kinesis 串流設定了時間，如果一個碎片的耗用速度比其他碎片快（例如，在應用程式初始化期間，或使用所有現有碎片的並行耗用 `TRIM_HORIZON` 時，忽略其父/子關係），仍可能發生延遲到達事件。
- 浮水印策略 `withIdleness` 的設定似乎會中斷閒置碎片的 Kinesis 來源特定設定 (`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.spliterator(), 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 kineses put-record --stream-name hp-16 --partition-key 1 --data MQ==
$ aws kineses put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kineses 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 kineses put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kineses 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 kineses 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.internal.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，因為預先建置的浮水印策略時間絕不會超過從串流讀取的最後一個事件時間。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。在此情況下，最後一個浮水印是 2022-03-23T10:21:27.170Z，但工作階段視窗關閉時，需要浮水印 10 秒和 1 毫秒之後。

如果從浮水印策略中移除 `withIdleness` 選項，則工作階段視窗將不會關閉，因為視窗運算子的「全域浮水印」無法繼續。

當 Flink 應用程式啟動時（或如果有資料扭曲），某些碎片的使用速度可能會比其他碎片快。這可能會導致部分浮水印過早從子任務發出（子任務可能會根據一個碎片的內容發出浮水印，而不會從訂閱的其他碎片耗用）。緩解的方式是不同的浮水印策略，可新增安全緩衝區 (`forBoundedOutOfOrderness(Duration.ofSeconds(30))`) 或明確允許延遲到達事件 (`allowedLateness(Time.minutes(5))`)。

為所有運算子設定 UUID

當 Managed Service for Apache Flink 使用快照為應用程式啟動 Flink 作業時，Flink 作業可能會因為某些問題而無法啟動。其中一個問題是運算子 ID 不符。Flink 預期 Flink 作業圖表運算子具有明確且一致的運算子 ID。如果未明確設定，Flink 會為運算子產生 ID。這是因為，Flink 使用這些運算子 ID 來唯一識別作業圖表中的運算子，並使用它們將每個運算子的狀態儲存在儲存點中。

當 Flink 在作業圖表的運算子 ID 與儲存點中定義的運算子 ID 之間找不到 1:1 對應時，就會發生運算子 ID 不符問題。當未設定明確一致運算子 IDs 且 Flink 產生可能與每個任務圖表建立不一致的運算子 IDs 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，我們建議客戶為 Flink 程式碼中的所有運算子設定 UUID。如需詳細資訊，請參閱生產就緒性下的[為所有運算子設定 UUID](#)主題。

將 ServiceResourceTransformer 新增至 Maven 著色外掛程式

Flink 使用 Java 的[服務提供者介面 \(SPI\)](#) 來載入連接器和格式等元件。使用 SPI 的多個 Flink 相依性[可能會在 uber-jar 和非預期的應用程式行為中造成衝突](#)。我們建議您新增 Maven 著色外掛程式的 [ServiceResourceTransformer](#)，如 pom.xml 中所定義。

```
<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>
```

Apache Flink 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。

以下是為 Managed Service for Apache Flink 進行的 StateFun Python 範例調整：

Apache Flink 應用程式範本

客戶可以編譯只調用有狀態函數執行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。

Apache Flink 設定

Managed Service for Apache Flink 是 Apache Flink 框架的實作。Managed Service for Apache Flink 使用本節所述的預設值。其中一些值可由 Managed Service for Apache Flink 應用程式在程式碼中設定，其他則無法變更。

使用本節中的連結，進一步了解 Apache flink 設定和可修改的設定。

本主題包含下列章節：

- [Apache Flink 組態](#)
- [狀態後端](#)
- [檢查點](#)
- [儲存點](#)
- [堆積大小](#)
- [緩衝區消脹](#)
- [可修改的 Flink 組態屬性](#)
- [程式設計 Flink 組態屬性](#)
- [檢視設定的 Flink 屬性](#)

Apache Flink 組態

Managed Service for Apache Flink 提供預設的 Flink 組態，其中包含大多數屬性的 Apache Flink 建議值，少數一些基於常用應用程式設定檔。如需 Flink 組態的詳細資訊，請參閱[組態](#)。服務提供的預設組態適用於大多數應用程式。不過，若要調整 Flink 組態屬性，以改善具有高平行處理、高記憶體和狀態用量的特定應用程式效能，或在 Apache Flink 中啟用新的偵錯功能，您可以透過請求支援案例來變更特定屬性。如需詳細資訊，請參閱[AWS 支援中心](#)。您可以使用 [Apache Flink 儀表板](#) 檢查應用程式的目前組態。

狀態後端

Managed Service for Apache Flink 將暫時性資料儲存在狀態後端。Managed Service for Apache Flink 使用 RocksDBStateBackend。呼叫 `setStateBackend` 來設定不同的後端沒有任何效果。

我們在狀態後端啟用以下功能：

- 增量狀態後端快照
- 非同步狀態後端快照
- 檢查點本機復原

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 [CheckpointConfiguration](#) 進行變更。您必須將 `CheckpointConfiguration.ConfigurationType` 將 CUSTOM Managed Service for Apache Flink 設定為 `CUSTOM`，才能使用修改後的檢查點值。

設定	可以修改嗎？	方法	預設值
CheckpointingEnabled	可修改	建立應用程式 更新應用程式 CloudFormation	True
CheckpointInterval	可修改	建立應用程式 更新應用程式 CloudFormation	60000
MinPauseBetweenCheckpoints	可修改	建立應用程式 更新應用程式 CloudFormation	5000
未對齊的檢查點	可修改	支援案例	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

設定	可以修改嗎？	方法	預設值
檢查點逾時	不可修改	N/A	60 分鐘
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。您可以透過將應用程式 [FlinkRunConfiguration](#) 的 `AllowNonRestoredState` 參數設定為 `true`，以允許操作成功。這將允許恢復操作跳過無法對應至新程式的狀態。

如需詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [允許非還原的狀態](#)。

堆積大小

Managed Service for Apache Flink 會為每個 KPU 分配 3 GiB 的 JVM 堆積，並為原生程式碼配置保留 1 GiB。如需增加應用程式容量的相關資訊，請參閱 [the section called “實作應用程式擴展”](#)。

如需 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。如果您想要修改此清單以外的其他 Flink 組態屬性，請在您的案例中指定確切的屬性。

重新啟動策略

從 Flink 1.19 及更新版本，預設會使用exponential-delay重新啟動策略。根據預設，所有先前的版本都會使用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.type:

state.backend.fs.memory-threshold:

state.backend.incremental:

檢查點

execution.checkpointing.unaligned:

execution.checkpointing.interval-during-backlog:

RocksDB 原生指標

RocksDB 原生指標不會運送到 CloudWatch。啟用後，您可以從 Flink 儀表板或使用自訂工具從 Flink REST API 存取這些指標。

Managed Service for Apache Flink 可讓客戶使用 [CreateApplicationPresignedUrl](#) API，以唯讀模式存取最新的 Flink [REST API](#) (或您正在使用的受支援版本)。此 API 由 Flink 自己的儀表板使用，但也可以由自訂監控工具使用。

```
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:`

RocksDB 選項

`state.backend.rocksdB.compaction.style:`

`state.backend.rocksdB.memory.partitioned-index-filters:`

`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.memory.network.fraction:`

`taskmanager.memory.network.max:`

`taskmanager.network.request-backoff.initial:`

`taskmanager.network.request-backoff.max:`

`taskmanager.network.memory.buffer-debloat.enabled:`

`taskmanager.network.memory.buffer-debloat.period:`

`taskmanager.network.memory.buffer-debloat.samples:`

`taskmanager.network.memory.buffer-debloat.threshold-percentages:`

記憶體組態

`taskmanager.memory.jvm-metaspace.size:`

`taskmanager.memory.jvm-overhead.fraction:`

`taskmanager.memory.jvm-overhead.max:`

`taskmanager.memory.managed.consumer-weights:`

`taskmanager.memory.managed.fraction:`

`taskmanager.memory.network.fraction:`

`taskmanager.memory.network.max:`

`taskmanager.memory.segment-size:`

`taskmanager.memory.task.off-heap.size:`

RPC / Akka

`akka.ask.timeout:`

`akka.client.timeout:`

`akka.framesize:`

`akka.lookup.timeout:`

`akka.tcp.timeout:`

用戶端

`client.timeout:`

進階叢集選項

`cluster.intercept-user-system-exit:`

`cluster.processes.halt-on-fatal-error:`

檔案系統組態

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

進階容錯能力選項

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

記憶體組態

`jobmanager.memory.heap.size:`

指標

`metrics.latency.interval:`

REST 端點和用戶端的進階選項

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

進階 SSL 安全選項

`security.ssl.internal.handshake-timeout:`

進階排程選項

`slot.request.timeout:`

Flink Web UI 的進階選項

`web.timeout:`

程式設計 Flink 組態屬性

以下是您可以直接在應用程式程式碼中修改的 Flink 組態屬性。從 MSF 2.2 開始，如果您嘗試修改應用程式程式碼中此頁面未列出的屬性，則會擲回例外狀況。

Note

這些屬性與透過 AWS 支援案例修改的設定不同。如需 TaskManager 記憶體、狀態後端、RocksDB 調校和重新啟動策略等基礎設施層級設定，請參閱 [可修改的 Flink 組態屬性](#)。

管道組態

`execution.runtime-mode`

`pipeline.auto-generate-uids`

`pipeline.auto-watermark-interval`

`pipeline.cached-files`

`pipeline.classpaths`

`pipeline.closure-cleaner-level`

`pipeline.force-avro`

`pipeline.force-kryo`

`pipeline.generic-types`

`pipeline.global-job-parameters`

`pipeline.jars`

`pipeline.jobvertex-parallelism-overrides`

`pipeline.max-parallelism`

`pipeline.name`

`pipeline.object-reuse`

`pipeline.operator-chaining.chain-operators-with-different-max-parallelism`

`pipeline.operator-chaining.enabled`

`pipeline.serialization-config`

`pipeline.vertex-description-mode`

`pipeline.vertex-name-include-index-prefix`

`pipeline.watermark-alignment.allow-unaligned-source-splits`

Python API

`python.execution-mode`

`python.operator-chaining.enabled`

`python.job-options`

`python.internal.archives-key-map`

`python.internal.files-key-map`

`python.internal.requirements-file-key`

資料表 API/SQL

`table.exec.async-lookup.buffer-capacity`

`table.exec.async-lookup.key-ordered-enabled`

`table.exec.async-lookup.output-mode`

`table.exec.async-lookup.timeout`

`table.exec.async-ml-predict.max-concurrent-operations`

`table.exec.async-ml-predict.output-mode`

`table.exec.async-ml-predict.timeout`

`table.exec.async-scalar.max-concurrent-operations`

`table.exec.async-scalar.max-attempts`

`table.exec.async-scalar.retry-delay`

`table.exec.async-scalar.retry-strategy`

`table.exec.async-scalar.timeout`

`table.exec.async-state.enabled`

`table.exec.async-table.max-concurrent-operations`

`table.exec.async-table.max-retries`

`table.exec.async-table.retry-delay`

`table.exec.async-table.retry-strategy`

`table.exec.async-table.timeout`

`table.exec.async-vector-search.max-concurrent-operations`

`table.exec.async-vector-search.output-mode`

`table.exec.async-vector-search.timeout`

`table.exec.deduplicate.insert-update-after-sensitive-enabled`

`table.exec.deduplicate.mini-batch.compact-changes-enabled`

`table.exec.delta-join.cache-enabled`

`table.exec.disabled-operators`

`table.exec.iceberg.use-v2-sink`

`table.exec.interval-join.min-cleanup-interval`

`table.exec.legacy-cast-behaviour`

`table.exec.local-hash-agg.adaptive.distinct-value-rate-threshold`

`table.exec.local-hash-agg.adaptive.enabled`

`table.exec.local-hash-agg.adaptive.sampling-threshold`

`table.exec.mini-batch.allow-latency`

`table.exec.mini-batch.enabled`

`table.exec.mini-batch.size`

`table.exec.operator-fusion-codegen.enabled`

`table.exec.simplify-operator-name-enabled`

`table.exec.sink.keyed-shuffle`

`table.exec.sink.nested-constraint-enforcer`

`table.exec.sink.not-null-enforcer`

`table.exec.sink.rowtime-inserter`

`table.exec.sink.type-length-enforcer`

`table.exec.sink.upsert-materialize`

`table.exec.sink.upsert-materialize-strategy.adaptive.threshold.high`

`table.exec.sink.upsert-materialize-strategy.adaptive.threshold.low`

`table.exec.sink.upsert-materialize-strategy.type`

`table.exec.sort.async-merge-enabled`

`table.exec.sort.default-limit`

`table.exec.sort.max-num-file-handles`

`table.exec.source.cdc-events-duplicate`

`table.exec.source.idle-timeout`

`table.exec.spill-compression.block-size`

`table.exec.spill-compression.enabled`

table.exec.state.ttl

table.exec.uid.format

table.exec.uid.generation

table.exec.unbounded-over.version

table.exec.window-agg.buffer-size-limit

table.optimizer.agg-phase-strategy

table.optimizer.adaptive-broadcast-join.strategy

table.optimizer.bushy-join-reorder-threshold

table.optimizer.delta-join.strategy

table.optimizer.distinct-agg.split.bucket-num

table.optimizer.distinct-agg.split.enabled

table.optimizer.dynamic-filtering.enabled

table.optimizer.incremental-agg-enabled

table.optimizer.join-reorder-enabled

table.optimizer.multi-join.enabled

table.optimizer.multiple-input-enabled

table.optimizer.non-deterministic-update.strategy

table.optimizer.reuse-optimize-block-with-digest-enabled

table.optimizer.reuse-sink-enabled

table.optimizer.reuse-source-enabled

table.optimizer.reuse-sub-plan-enabled

table.optimizer.runtime-filter.enabled

table.optimizer.skewed-join-optimization.skewed-factor

`table.optimizer.skewed-join-optimization.skewed-threshold`

`table.optimizer.skewed-join-optimization.strategy`

`table.optimizer.source.report-statistics-enabled`

`table.optimizer.union-all-as-breakpoint-enabled`

`table.builtin-catalog-name`

`table.builtin-database-name`

`table.catalog-modification.listeners`

`table.column-expansion-strategy`

`table.display.max-column-width`

`table.dml-sync`

`table.dynamic-table-options.enabled`

`table.generated-code.max-length`

`table.legacy-nested-row-nullability`

`table.local-time-zone`

`table.plan.compile.catalog-objects`

`table.plan.force-recompile`

`table.plan.restore.catalog-objects`

`table.rtas-ctas.atomicity-enabled`

`table.sql-dialect`

檢視設定的 Flink 屬性

您可以透過 Apache Flink 儀表板檢視您自己設定或請求透過[支援案例](#)修改的 Apache Flink 屬性，並遵循下列步驟進行操作：

1. 前往 Flink 儀表板
2. 在左側導覽窗格中選擇作業管理員。
3. 選擇組態以檢視 Flink 屬性的清單。

設定 Managed Service for Apache Flink 以存取 Amazon VPC 中的資源

您可以設定 Managed Service for Apache Flink 應用程式連線到您帳戶中虛擬私有雲端 (VPC) 中的私有子網路。使用 Amazon Virtual Private Cloud (Amazon VPC) 為資料庫、快取執行個體或內部服務等資源建立私有網路。將應用程式連線到 VPC 以在執行期間存取私有資源。

本主題包含下列章節：

- [Amazon VPC 概念](#)
- [VPC 應用程式許可](#)
- [VPC 連線的 Managed Service for Apache Flink 應用程式的網際網路和服務存取](#)
- [使用 Managed Service for Apache Flink VPC API](#)
- [範例：使用 VPC 存取 Amazon MSK 叢集中的資料](#)

Amazon VPC 概念

Amazon VPC 是 Amazon EC2 的網路層。如果您是 Amazon EC2 的新手，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的[什麼是 Amazon EC2？](#)，以取得簡要概觀。

以下是 VPC 的重要概念：

- 虛擬私有雲端 (VPC) 是您 AWS 帳戶的專用虛擬網路。
- 子網是您的 VPC 中的 IP 地址範圍。
- 「路由表」包含一組名為路由的規則，用來判斷網路流量的方向。
- 網際網路閘道是一種水平擴展、備援且高可用性的 VPC 元件，允許 VPC 中執行個體與網際網路之間的通訊。因此不會對網路流量強加可用性風險或頻寬限制。
- VPC 端點可讓您將 VPC 私下連線至支援的 AWS 服務和採用 PrivateLink 技術的 VPC 端點服務，而不需要網際網路閘道、NAT 裝置、VPN 連接或 Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址，即可與服務中的資源通訊。VPC 與另一個服務之間的流量都會保持在 Amazon 網路的範圍內。

如需 Amazon VPC 服務的詳細資訊，請參閱 [Amazon Virtual Private Cloud 使用者指南](#)。

Managed Service for Apache Flink 會在應用程式的 VPC 組態中提供的其中一個子網路中建立 [彈性網路介面 \(ENI\)](#)。VPC 子網路中建立的 ENI 數目可能會有所不同，取決於應用程式平行處理層級及其每個 KPU 的平行處理層級。如需應用程式擴展的詳細資訊，請參閱 [實作應用程式擴展](#)。

Note

SQL 應用程式不支援 VPC 組態。

Note

Managed Service for Apache Flink 服務管理具有 VPC 組態之應用程式的檢查點和快照狀態。

VPC 應用程式許可

本節說明您的應用程式在使用 VPC 時需要的許可政策。如需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](#)。

下列許可政策會授與您的應用程式與 VPC 互動的必要許可。若要使用此許可政策，請將其新增至應用程式的執行角色。

新增存取 Amazon VPC 的許可政策

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    }
  ],
}
```

```
{
  "Sid": "ENIReadWritePermissions",
  "Effect": "Allow",
  "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DescribeNetworkInterfaces",
    "ec2>DeleteNetworkInterface"
  ],
  "Resource": "*"
}
```

Note

當您使用主控台 (例如 CloudWatch Logs 或 Amazon VPC) 指定應用程式資源時，主控台會修改您的應用程式執行角色，以授與存取這些資源的許可。只有在不使用主控台的情況下建立應用程式時，才需要手動修改應用程式的執行角色。

VPC 連線的 Managed Service for Apache Flink 應用程式的網際網路和服務存取

當您將 Managed Service for Apache Flink 應用程式連線至您帳戶的 VPC 時，除非 VPC 提供存取權，否則該應用程式無法存取網際網路。如果應用程式需要網際網路存取，必須符合下列條件：

- Managed Service for Apache Flink 應用程式只能使用私有子網路進行設定。
- VPC 必須在公有子網路中包含 NAT 閘道或執行個體。
- 從私有子網路到公有子網路 NAT 閘道之間必須存在路由以傳輸傳出流量。

Note

多項服務提供 [VPC 端點](#)。您可以使用 VPC 端點從 VPC 內部連線至 Amazon 服務，而不需要存取網際網路。

子網路是公有還是私有，取決於其路由表。每個路由表都有一個預設路由，決定具有公用目的地之封包的下一個躍點。

- 對於私有子網路：預設路由指向 NAT 閘道 (nat-...) 或 NAT 執行個體 (eni-...)。
- 對於公有子網路：預設路由指向網際網路閘道 (igw-...)。

使用一個公有子網路 (使用 NAT) 和一或多個私有子網路設定 VPC 後，請執行下列動作以識別您的私有和公有子網路：

- 在 VPC 主控台的導覽窗格中，選擇子網路。
- 選取子網路，然後選擇路由表標籤。驗證預設路由：
 - 公有子網路：目的地：0.0.0.0/0，目標：igw-...
 - 私有子網路：目的地：0.0.0.0/0，目標：nat-... 或 eni-...

在 Managed Service for Apache Flink 應用程式與私有子網路之間建立關聯：

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/flink> 開啟 Amazon MSF 主控台。
- 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，選擇要與應用程式建立關聯的 VPC。選擇希望應用程式用來存取 VPC 資源的與 VPC 相關聯的子網路和安全群組。
- 選擇更新。

相關資訊

[建立含公有子網路和私有子網路的 VPC](#)

[NAT 閘道基本概念](#)

使用 Managed Service for Apache Flink VPC API

您可以使用下列 Managed Service for Apache Flink API 作業來管理應用程式的 VPC。如需如何使用 Managed Service for Apache Flink API 的相關資訊，請參閱[API 範例程式碼](#)。

建立應用程式

使用 [CreateApplication](#) 動作可在建立期間將 VPC 組態新增至應用程式。

CreateApplication 動作的下列請求程式碼範例包括建立應用程式時的 VPC 組態：

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration": {
      "ParallelismConfiguration": {
        "ConfigurationType": "CUSTOM",
        "Parallelism": 2,
        "ParallelismPerKPU": 1,
        "AutoScalingEnabled": true
      }
    }
  },
  "VpcConfigurations": [
    {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
    }
  ]
}
```

AddApplicationVpcConfiguration

使用 [AddApplicationVpcConfiguration](#) 動作可在建立 VPC 組態之後將其新增至應用程式。

AddApplicationVpcConfiguration 動作的下列範例請求程式碼會將 VPC 組態新增至現有的應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfiguration": {
    "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
    "SubnetIds": [ "subnet-0123456789abcdef0" ]
  }
}
```

DeleteApplicationVpcConfiguration

使用 [DeleteApplicationVpcConfiguration](#) 動作可從應用程式中移除 VPC 組態。

AddApplicationVpcConfiguration 動作的下列範例請求程式碼可將現有的 VPC 組態從應用程式中移除：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

更新應用程式

使用 [UpdateApplication](#) 動作可一次更新應用程式的所有 VPC 組態。

UpdateApplication 動作的下列範例請求程式碼可更新應用程式的所有 VPC 組態：

```
{
  "ApplicationConfigurationUpdate": {
    "VpcConfigurationUpdates": [
      {
        "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
        "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
        "VpcConfigurationId": "2.1"
      }
    ]
  }
}
```

```
    ]  
  },  
  "ApplicationName": "MyApplication",  
  "CurrentApplicationVersionId": 9  
}
```

範例：使用 VPC 存取 Amazon MSK 叢集中的資料

如需如何在 VPC 中存取 Amazon MSK 叢集資料的完整教學課程，請參閱[MSK 複寫](#)。

Managed Service for Apache Flink 故障診斷

下列主題可協助您針對 Amazon Managed Service for Apache Flink 可能遇到的問題進行疑難排解。

選擇適當的主題來檢閱解決方案。

主題

- [開發疑難排解](#)
- [執行期疑難排解](#)

開發疑難排解

本節包含診斷和修正 Managed Service for Apache Flink 應用程式開發問題的相關資訊。

主題

- [系統復原最佳實務](#)
- [Hudi 組態最佳實務](#)
- [Apache Flink 火焰圖](#)
- [EFO 連接器 1.15.2 的登入資料提供者問題](#)
- [應用程式使用不支援的 Kinesis 連接器](#)
- [編譯錯誤：「無法解析專案的相依性」](#)
- [無效的選擇：「kinesisanalyticsv2」](#)
- [UpdateApplication 動作不會重新載入應用程式程式碼](#)
- [S3 StreamingFileSink FileNotFoundExceptions](#)
- [使用儲存點停止時的 FlinkKafkaConsumer 問題](#)
- [FLINK 1.15 非同步接收器死鎖](#)
- [在重新分片期間，Amazon Kinesis 資料串流來源處理順序不正確](#)
- [即時向量內嵌藍圖常見問答集和疑難排解](#)

系統復原最佳實務

透過 Amazon Managed Service for Apache Flink 中的自動系統復原和操作可見性功能，您可以識別和解決應用程式的問題。

系統轉返

如果您的應用程式更新或擴展操作因客戶錯誤而失敗，例如程式碼錯誤或許可問題，如果您已選擇使用此功能，Amazon Managed Service for Apache Flink 會自動嘗試回復至先前的執行版本。如需詳細資訊，請參閱[為您的 Managed Service for Apache Flink 應用程式啟用系統復原](#)。如果此自動轉返失敗，或您尚未選擇加入或退出，您的應用程式將進入 READY 狀態。若要更新您的應用程式，請完成下列步驟：

手動轉返

如果應用程式未進行且長時間處於暫時性狀態，或者應用程式成功轉換為 Running，但您看到下游問題，例如在成功更新的 Flink 應用程式中處理錯誤，您可以使用 RollbackApplication API 手動將其轉返。

1. 呼叫 RollbackApplication - 這將還原至先前的執行版本，並還原先前的狀態。
2. 使用 DescribeApplicationOperation API 監控轉返操作。
3. 如果轉返失敗，請使用先前的系統轉返步驟。

操作可見性

ListApplicationOperations API 會顯示應用程式上所有客戶和系統操作的歷史記錄。

1. 從清單中取得失敗操作的 operationId。
2. 呼叫 DescribeApplicationOperation 並檢查狀態和 statusDescription。
3. 如果操作失敗，描述會指向調查的潛在錯誤。

常見的錯誤碼錯誤：使用復原功能來還原至上次運作的版本。解決錯誤並重試更新。

許可問題：使用 DescribeApplicationOperation 查看所需的許可。更新應用程式許可並重試。

Amazon Managed Service for Apache Flink 服務問題：檢查 AWS Health 儀板表 或開啟支援案例。

Hudi 組態最佳實務

若要在 Managed Service for Apache Flink 上執行 Hudi 連接器，我們建議進行下列組態變更。

停用 `hoodie.embed.timeline.server`

Flink 上的 Hudi 連接器會在 Flink 任務管理員 (JM) 上設定內嵌時間軸 (TM) 伺服器，以快取中繼資料，以在任務平行處理很高時改善效能。我們建議您在 Managed Service for Apache Flink 上停用此內嵌伺服器，因為我們停用 JM 和 TM 之間的非 Flink 通訊。

如果啟用此伺服器，Hudi 寫入會先嘗試連接到 JM 上的內嵌伺服器，然後回到從 Amazon S3 讀取中繼資料。這表示 Hudi 發生連線逾時，延遲 Hudi 寫入，並對 Managed Service for Apache Flink 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停用火焰圖，請建立一個案例，以請求將其為您的應用程式 ARN 停用。如需詳細資訊，請參閱 [AWS 支援中心](#)。

EFO 連接器 1.15.2 的登入資料提供者問題

Kinesis Data Streams EFO 連接器截止 1.15.2 版本都存在一個[已知問題](#)，其中的 `FlinkKinesisConsumer` 不遵守 `Credential Provider` 組態。由於該問題，有效組態被忽略，導致使用 `AUTO` 憑證提供者。使用 EFO 連接器跨帳戶存取 Kinesis 時，這可能會導致問題。

若要解決此錯誤，請使用 EFO 連接器 1.15.3 版或更新版本。

應用程式使用不支援的 Kinesis 連接器

Managed Service for Apache Flink 1.15 版或更新版本會在[應用程式使用不支援的 Kinesis Connector 版本 \(1.15.2 版之前\)](#) 封裝至應用程式 JAR 或封存檔 (ZIP) 時，自動拒絕應用程式啟動或更新。JARs

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：「無法解析專案的相依性」

若要編譯 Managed Service for Apache Flink 範例應用程式，必須先下載並編譯 Apache Flink Kinesis 連接器，並將其新增至本機 Maven 儲存庫。如果連接器尚未新增至儲存庫，則會出現如下編譯錯誤：

```
Could not resolve dependencies for project your project name: Failure to  
find org.apache.flink:flink-connector-kinesis_2.11:jar:1.8.2 in https://  
repo.maven.apache.org/maven2 was cached in the local repository, resolution will not be  
reattempted until the update interval of central has elapsed or updates are forced
```

若要解決此錯誤，必須從 <https://flink.apache.org/downloads.html> 為連接器下載 Apache Flink 來源程式碼 1.8.2 版本。如需如何下載、編譯和安裝 Apache Flink 來源程式碼的相關指示，請參閱[the section called “將 Apache Flink Kinesis Streams 連接器與先前的 Apache Flink 版本搭配使用”](#)。

無效的選擇：「kinesisanalyticsv2」

若要使用 Managed Service for Apache Flink API v2，需要最新版本的 AWS Command Line Interface (AWS CLI)。

如需有關升級的資訊 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[安裝 AWS Command Line Interface](#)。

UpdateApplication 動作不會重新載入應用程式程式碼

如果未指定 S3 物件版本，[UpdateApplication](#) 動作將不會以相同的檔案名稱重新載入應用程式程式碼。若要使用相同的檔案名稱重新載入應用程式程式碼，請在 S3 儲存貯體上啟用版本控制，然後使用

ObjectVersionUpdate 參數指定新的物件版本。如需在 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://amzn-s3-demo-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 會將記錄 StreamingFileSink 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 Managed Service for Apache Flink 應用程式的快照過時 — 只有停止或更新應用程式時所拍攝的最新系統快照可用來啟動使用 Amazon S3 StreamingFileSink 的 Managed Service for Apache Flink 應用程式。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分段檔案的過期參考，而該檔案已由後續檢查點或儲存點重新命名並遞交。
- 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，使用 Amazon S3 `StreamingFileSink` 的 Managed Service for Apache Flink 應用程式應始終啟用系統快照並設定 `RESTORE_FROM_LATEST_SNAPSHOT`。
- 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。
 - 當已停止應用程式太長時間，且 [S3 儲存貯體 MultiPartUpload](#) 生命週期政策移除了應用程式儲存點所參考的進行中部分檔案時，就可能會發生這種情況。若要避免此類失敗，請確保 S3 儲存貯體 MPU 生命週期政策針對您的使用案例涵蓋了足夠長的期間。
 - 當進行中的分段檔案已手動移除或由系統的其他元件移除時，也會發生這種情況。若要避免此類失敗，請確定進行中的分段檔案不會被其他參與者或元件移除。
- 在儲存點之後觸發自動檢查點的競爭情形 — 這會影響 Managed Service for Apache Flink 1.13 及以下版本。Managed Service for Apache Flink 1.15 版已修正此問題。將您的應用程式遷移至最新版本的 Managed Service for Apache Flink，以防止重複。我們還建議您從 `StreamingFileSink` 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。

使用儲存點停止時的 FlinkKafkaConsumer 問題

使用舊版 `FlinkKafkaConsumer` 時，如果您啟用了系統快照，應用程式可能會卡在 `UPDATING`、`STOPPING` 或 `SCALING` 狀態。此[問題](#)沒有已發佈的修正程式可用，因此建議您升級至新的 [KafkaSource](#) 以緩解此問題。

如果您正在啟用快照的情況下使用 `FlinkKafkaConsumer`，Flink 作業可能會使用儲存點 API 請求處理停止，`FlinkKafkaConsumer` 可能會失敗並報告執行期錯誤 `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

實作 `AsyncSink` 介面的 Apache Flink AWS 連接器存在[已知問題](#)。這會影響使用具有下列連接器的 Flink 1.15 的應用程式：

- 對於 Java 應用程式：
 - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`
 - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
 - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
 - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`
- 快速 SQL/資料表 API/Python 應用程式：
 - kinesis – `org.apache.flink:flink-sql-connector-kinesis`
 - kinesis – `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
 - firehose – `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
 - dynamodb – `org.apache.flink:flink-sql-connector-dynamodb`

受影響的應用程式會出現下列徵狀：

- FLINK 作業處於 RUNNING 狀態，但未在處理資料；
- 沒有作業重新啟動；
- 檢查點逾時。

問題是由 AWS SDK 中的[錯誤](#)所造成，導致在使用非同步 HTTP 用戶端時不會向發起人顯示特定錯誤。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 版開始，已在 AWS SDK 中修正此問題。

以下是如何更新受影響連接器以在應用程式中使用新版本 AWS SDK 的指示：

主題

- [更新 Java 應用程式](#)
- [更新 Python 應用程式](#)

更新 Java 應用程式

請依照下列程序更新 Java 應用程式：

flink-connector-kinesis

如果應用程式使用 `flink-connector-kinesis`：

Kinesis 連接器使用著色將一些相依性，包括 AWS SDK 封裝到連接器 jar 中。若要更新 AWS SDK 版本，請使用下列程序取代這些著色類別：

Maven

1. 新增 Kinesis 連接器和必要的 AWS SDK 模組做為專案相依性。
2. 設定 `maven-shade-plugin`：
 - a. 新增篩選條件，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，將更新的 AWS SDK 類別移至 Kinesis 連接器預期的套件。

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>
</project>
```

```

    ...
</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. 新增 Kinesis 連接器和必要的 AWS SDK 模組做為專案相依性。
2. 調整 shadowJar 組態：
 - a. 複製 Kinesis 連接器 jar 的內容時，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，應在專案建置組態中強制執行 SDK 版本。

Maven

將 AWS SDK 物料清單 (BOM) 新增至 pom.xml 檔案的相依性管理區段，以強制執行專案的 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 AWS SDK 物料清單 (BOM) 上新增平台相依性，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-kinesis-streams

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-kinesis-firehose

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：

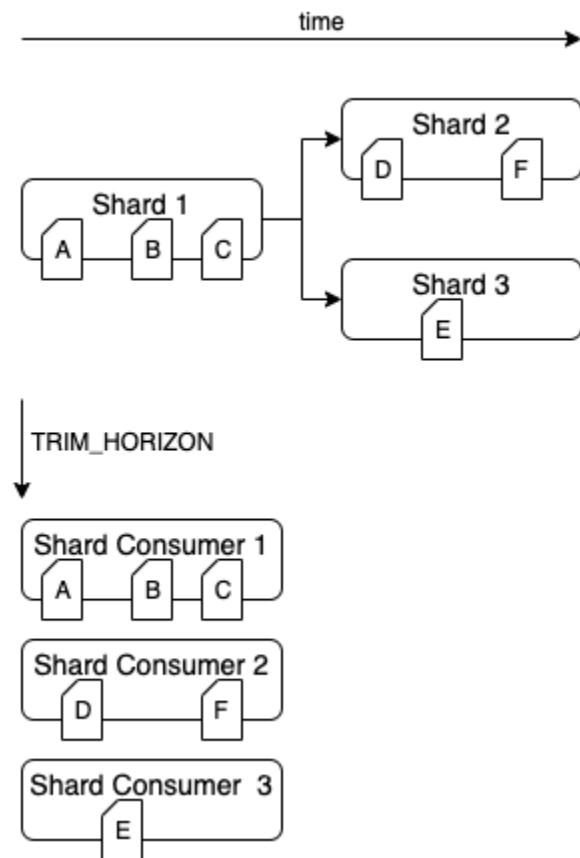
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -  
Daws.sdk.version=2.20.144
```

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 資料串流來源處理順序不正確

目前的 FlinkKinesisConsumer 實作無法在 Kinesis 碎片之間提供強有力的排序保證。這可能會導致 Kinesis 串流重新分片期間出現不按順序處理的情況，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。透過確保 Flink 應用程式可以容忍處理延遲，您可以最大程度地減少錯誤處理的影響和資料遺失的風險。

即時向量內嵌藍圖常見問答集和疑難排解

檢閱下列常見問答集和疑難排解章節，以疑難排解即時向量內嵌藍圖問題。如需即時向量內嵌藍圖的詳細資訊，請參閱[即時向量內嵌藍圖](#)。

如需一般 Managed Service for Apache Flink 應用程式故障診斷，請參閱 <https://docs.aws.amazon.com/managed-flink/latest/java/troubleshooting-runtime.html>。

主題

- [即時向量內嵌藍圖 - 常見問答集](#)
- [即時向量內嵌藍圖 - 疑難排解](#)

即時向量內嵌藍圖 - 常見問答集

檢閱下列有關即時向量內嵌藍圖的常見問答集。如需即時向量內嵌藍圖的詳細資訊，請參閱[即時向量內嵌藍圖](#)。

常見問答集

- [此藍圖會建立哪些 AWS 資源？](#)
- [AWS CloudFormation 堆疊部署完成後，我的動作是什麼？](#)
- [來源 Amazon MSK 主題 \(Amazon MSK\) 中的資料結構應該是什麼？](#)
- [我可以指定要內嵌的訊息部分嗎？](#)
- [我可以從多個 Amazon MSK 主題讀取資料嗎？](#)
- [我可以使用 regex 來設定 Amazon MSK 主題名稱嗎？](#)
- [可以從 Amazon MSK 主題讀取的訊息大小上限是多少？](#)
- [支援哪種類型的 OpenSearch？](#)
- [為什麼我需要在 OpenSearch Serverless collection 中使用向量搜尋集合、向量索引和新增向量欄位？](#)
- [我應該將什麼設定為向量欄位的維度？](#)
- [在設定的 OpenSearch 索引中，輸出看起來像什麼？](#)
- [我可以指定中繼資料欄位以新增至儲存在 OpenSearch 索引中的文件嗎？](#)
- [我應該預期 OpenSearch 索引中重複的項目嗎？](#)
- [我可以將資料傳送至多個 OpenSearch 索引嗎？](#)

- [我可以在單一 中部署多個即時向量內嵌應用程式 AWS 帳戶嗎？](#)
- [多個即時向量內嵌應用程式是否可以使用相同的資料來源或接收器？](#)
- [應用程式是否支援跨帳戶連線？](#)
- [應用程式是否支援跨區域連線？](#)
- [我的 Amazon MSK 叢集和 OpenSearch 集合是否可以位於不同的 VPCs或子網路中？](#)
- [應用程式支援哪些內嵌模型？](#)
- [我可以根據工作負載微調應用程式的效能嗎？](#)
- [支援哪些 Amazon MSK 身分驗證類型？](#)
- [什麼是 sink.os.bulkFlushIntervalMillis 以及如何設定它？](#)
- [當我部署 Managed Service for Apache Flink 應用程式時，會從 Amazon MSK 主題的哪個時間點開始讀取訊息？](#)
- [如何使用 source.msk.starting.offset？](#)
- [支援哪些區塊策略？](#)
- [如何讀取向量資料存放區中的記錄？](#)
- [哪裡可以找到原始程式碼的新更新？](#)
- [我可以變更 AWS CloudFormation 範本並更新 Managed Service for Apache Flink 應用程式嗎？](#)
- [會代表我 AWS 監控和維護應用程式嗎？](#)
- [這個應用程式是否會將我的資料移動到我的 之外 AWS 帳戶？](#)

此藍圖會建立哪些 AWS 資源？

若要尋找您帳戶中部署的資源，請導覽至 AWS CloudFormation 主控台，並識別以您為 Managed Service for Apache Flink 應用程式提供的名稱開頭的堆疊名稱。選擇資源索引標籤，以檢查在堆疊中建立的資源。以下是堆疊建立的關鍵資源：

- 即時向量內嵌 Managed Service for Apache Flink 應用程式
- Amazon S3 儲存貯體，用於保留即時向量內嵌應用程式的原始程式碼
- 用於儲存日誌的 CloudWatch 日誌群組和日誌串流
- 用於擷取和建立資源的 Lambda 函數
- Lambdas、Managed Service for Apache Flink 應用程式以及存取 Amazon Bedrock 和 Amazon OpenSearch Service 的 IAM 角色和政策
- Amazon OpenSearch Service 的資料存取政策

- 用於存取 Amazon Bedrock 和 Amazon OpenSearch Service 的 VPC 端點

AWS CloudFormation 堆疊部署完成後，我的動作是什麼？

AWS CloudFormation 堆疊部署完成後，請存取 Managed Service for Apache Flink 主控台，並尋找您的藍圖 Managed Service for Apache Flink 應用程式。選擇設定索引標籤，並確認所有執行期屬性都已正確設定。它們可能會溢位到下一頁。當您對設定有信心時，請選擇執行。應用程式將開始從您的主題擷取訊息。

若要檢查是否有新版本，請參閱 <https://github.com/aws-labs/real-time-vectorization-of-streaming-data/releases>。

來源 Amazon MSK 主題 (Amazon MSK) 中的資料結構應該是什麼？

我們目前支援結構化和非結構化來源資料。

- 非結構化資料在 STRING 中由 `source.msk.data.type` 表示。資料會依原樣從傳入訊息讀取。
- 我們目前支援結構化 JSON 資料，在 JSON 中由 `source.msk.data.type` 表示。資料必須一律採用 JSON 格式。如果應用程式收到格式不正確的 JSON，應用程式將會失敗。
- 使用 JSON 做為來源資料類型時，請確定所有來源主題中的每則訊息都是有效的 JSON。如果您訂閱的一或多個主題不包含具有此設定的 JSON 物件，應用程式將會失敗。如果一個或多個主題混合了結構化和非結構化資料，建議您在 Managed Service for Apache Flink 應用程式中將來源資料設定為非結構化。

我可以指定要內嵌的訊息部分嗎？

- 對於 `source.msk.data.type` 為的非結構化輸入資料 STRING，應用程式將一律嵌入整個訊息，並將整個訊息存放在設定的 OpenSearch 索引中。
- 對於 `source.msk.data.type` 為的結構化輸入資料 JSON，您可以設定 `embed.input.config.json.fieldsToEmbed` 來指定應該在 JSON 物件中選取哪個欄位以進行內嵌。這僅適用於最上層 JSON 欄位，不適用於巢狀 JSONs 和包含 JSON 陣列的訊息。使用 `*` 內嵌整個 JSON。

我可以從多個 Amazon MSK 主題讀取資料嗎？

可以，您可以使用此應用程式從多個 Amazon MSK 主題讀取資料。所有主題的資料都必須是相同類型 (STRING 或 JSON)，否則可能會導致應用程式失敗。來自所有主題的資料一律存放在單一 OpenSearch 索引中。

我可以使用 regex 來設定 Amazon MSK 主題名稱嗎？

source.msk.topic.names 不支援 regex 的清單。我們支援以逗號分隔的主題名稱清單或 regex .* 來包含所有主題。

可以從 Amazon MSK 主題讀取的訊息大小上限是多少？

可處理的訊息大小上限受限於目前設定為 25,000,000 的 Amazon Bedrock InvokeModel 內文限制。如需詳細資訊，請參閱 [InvokeModel](#)。

支援哪種類型的 OpenSearch？

我們同時支援 OpenSearch 網域和集合。如果您使用的是 OpenSearch 集合，請務必使用向量集合並建立向量索引以用於此應用程式。這可讓您使用 OpenSearch 向量資料庫功能來查詢資料。若要進一步了解，請參閱[Amazon OpenSearch Service 的向量資料庫功能說明](#)。

為什麼我需要在 OpenSearch Serverless collection 中使用向量搜尋集合、向量索引和新增向量欄位？

OpenSearch Serverless 中的向量搜尋集合類型提供可擴展且高效能的相似性搜尋功能。它簡化了建置現代機器學習 (ML) 增強型搜尋體驗和生成式人工智慧 (AI) 應用程式。如需詳細資訊，請參閱[使用向量搜尋集合](#)。

我應該將什麼設定為向量欄位的維度？

根據您要使用的內嵌模型，設定向量欄位的維度。請參閱下表，並從個別文件中確認這些值。

向量欄位維度

Amazon Bedrock 向量內嵌模型名稱	模型提供的輸出維度支援
Amazon Titan 文字內嵌 V1	1,536
Amazon Titan 文本嵌入 V2	1,024 (預設)、384、256
Amazon Titan Multimodal Embeddings G1	1,024 (預設)、384、256
Cohere Embed 英文	1,024
Cohere Embed 多語言	1,024

在設定的 OpenSearch 索引中，輸出看起來像什麼？

OpenSearch 索引中的每個文件都包含下列欄位：

- `original_data`：用於產生內嵌的資料。對於 `STRING` 類型，它是整個訊息。對於 `JSON` 物件，它是用於內嵌的 `JSON` 物件。它可以是訊息中的整個 `JSON`，或 `JSON` 中的指定欄位。例如，如果已選取要從傳入訊息內嵌的名稱，則輸出會如下所示：

```
"original_data": "{\"name\":\"John Doe\"}"
```

- `embedded_data`：Amazon Bedrock 產生的內嵌向量浮點數陣列
- 日期：文件存放在 OpenSearch 中的 UTC 時間戳記

我可以指定中繼資料欄位以新增至儲存在 OpenSearch 索引中的文件嗎？

否，目前我們不支援將其他欄位新增至儲存在 OpenSearch 索引中的最終文件。

我應該預期 OpenSearch 索引中重複的項目嗎？

根據您設定應用程式的方式，您可能會在索引中看到重複的訊息。其中一個常見原因是應用程式重新啟動。根據預設，應用程式會設定為從來源主題中最早的訊息開始讀取。當您變更組態時，應用程式會重新啟動，並再次處理主題中的所有訊息。若要避免重新處理，請參閱有關如何使用 `source.msk.starting.offset` 的文件，並正確設定應用程式的起始位移。

我可以將資料傳送至多個 OpenSearch 索引嗎？

否，應用程式支援將資料儲存到單一 OpenSearch 索引。若要將向量化輸出設定為多個索引，您必須部署個別的 Managed Service for Apache Flink 應用程式。

我可以在單一 中部署多個即時向量內嵌應用程式 AWS 帳戶嗎？

可以，AWS 帳戶 如果每個應用程式都有唯一的名稱，您可以在單一 中部署多個即時向量內嵌 Managed Service for Apache Flink 應用程式。

多個即時向量內嵌應用程式是否可以使用相同的資料來源或接收器？

可以，您可以建立多個即時向量內嵌 Managed Service for Apache Flink 應用程式（從相同主題讀取資料），或將資料儲存在相同索引中。

應用程式是否支援跨帳戶連線？

否，若要讓應用程式成功執行，Amazon MSK 叢集和 OpenSearch 集合必須位於您嘗試設定 Managed Service for Apache Flink 應用程式的相同 AWS 帳戶 位置。

應用程式是否支援跨區域連線？

否，應用程式只允許您在 Managed Service for Apache Flink 應用程式的相同區域中，使用 Amazon MSK 叢集和 OpenSearch 集合部署 Managed Service for Apache Flink 應用程式。

我的 Amazon MSK 叢集和 OpenSearch 集合是否可以位於不同的 VPCs 或子網路中？

是，我們支援不同 VPCs 和子網路中的 Amazon MSK 叢集和 OpenSearch 集合，只要它們位於相同位置即可 AWS 帳戶。請參閱（一般 MSF 疑難排解）以確保您的設定正確。

應用程式支援哪些內嵌模型？

目前，應用程式支援 Bedrock 支援的所有模型。其中包含：

- Amazon Titan Embeddings G1 - Text
- Amazon Titan 文本嵌入 V2
- Amazon Titan Multimodal Embeddings G1
- Cohere Embed 英文
- Cohere Embed 多語言

我可以根據工作負載微調應用程式的效能嗎？

是。應用程式輸送量取決於多種因素，所有因素都可以由客戶控制：

1. AWS MSF KPIs：以預設平行處理係數 2 和每個 KPI 1 的平行處理來部署應用程式，並開啟自動擴展。不過，我們建議您根據您的工作負載設定 Managed Service for Apache Flink 應用程式的擴展。如需詳細資訊，請參閱[檢閱 Managed Service for Apache Flink 應用程式資源](#)。
2. Amazon Bedrock：根據選取的 Amazon Bedrock 隨需模型，可能會套用不同的配額。檢閱 Bedrock 中的服務配額，以查看服務能夠處理的工作負載。如需詳細資訊，請參閱[Amazon Bedrock 的配額](#)。
3. Amazon OpenSearch Service：此外，在某些情況下，您可能會注意到 OpenSearch 是管道中的瓶頸。如需擴展資訊，請參閱 OpenSearch 擴展[規模調整 Amazon OpenSearch Service 網域](#)。

支援哪些 Amazon MSK 身分驗證類型？

我們僅支援 IAM MSK 身分驗證類型。

什麼是 `sink.os.bulkFlushIntervalMillis` 以及如何設定它？

將資料傳送至 Amazon OpenSearch Service 時，大量排清間隔是執行大量請求的間隔，無論動作數目或請求大小為何。預設值設定為 1 毫秒。

設定排清間隔有助於確保資料及時編製索引，如果設定過低，也可能會導致額外負荷增加。考慮您的使用案例，以及在選擇排清間隔時及時編製索引的重要性。

當我部署 Managed Service for Apache Flink 應用程式時，會從 Amazon MSK 主題的哪個時間點開始讀取訊息？

應用程式將以應用程式執行時間 `source.msk.starting.offset` 組態中組態設定所指定的位移，開始從 Amazon MSK 主題讀取訊息。如果 `source.msk.starting.offset` 未明確設定，應用程式的預設行為是從主題中最早的可用訊息開始讀取。

如何使用 `source.msk.starting.offset`？

根據所需的行為，將明確設為 `source.msk.starting.offset` 下列其中一個值：

- EARLIEST：預設設定，從分割區中最舊的位移讀取。這是一個不錯的選擇，特別是在以下情況：
 - 您已建立新的 Amazon MSK 主題和取用者應用程式。
 - 您需要重播資料，才能建置或重建狀態。這與實作事件來源模式或初始化需要完整檢視資料歷史記錄的新服務相關。
- LATEST：Managed Service for Apache Flink 應用程式會從分割區結尾讀取訊息。如果您只關心正在產生的新訊息，而且不需要處理歷史資料，建議您使用此選項。在此設定中，消費者會忽略現有的訊息，並只讀取上游生產者發佈的新訊息。
- COMMITTED：Managed Service for Apache Flink 應用程式將從耗用群組的遞交偏移開始耗用訊息。如果遞交的位移不存在，則會使用 EARLIEST 重設策略。

支援哪些區塊策略？

我們使用 [langchain](#) 程式庫來區塊輸入。只有在輸入的長度大於選擇的時，才會套用區塊。`maxSegmentSizeInChars` 我們支援以下五種區塊類型：

- SPLIT_BY_CHARACTER：將盡可能多的字元放入每個區塊，其中每個區塊的長度不大於 `maxSegmentSizeInChars`。不在乎空格，因此可以截斷單字。
- SPLIT_BY_WORD：將尋找要區塊的空格字元。沒有任何單字被截斷。

- `SPLIT_BY_SENTENCE`：句子界限是使用 Apache OpenNLP 程式庫搭配英文句子模型進行偵測。
- `SPLIT_BY_LINE`：會尋找要區塊的新行字元。
- `SPLIT_BY_PARAGRAPH`：將尋找要區塊的連續新行字元。

分割策略會根據上述順序回復，而較大的區塊策略如 `SPLIT_BY_PARAGRAPH` 會回復為 `SPLIT_BY_CHARACTER`。例如，使用時 `SPLIT_BY_LINE`，如果某行太長，則該行將依句子子區塊化，其中每個區塊將盡可能放入任意數量的句子中。如果有任何句子太長，則會在單字層級將其區塊化。如果單字太長，則會依字元分割。

如何讀取向量資料存放區中的記錄？

1. 當 `source.msk.data.type` 為 時 `STRING`

- `original_data`：來自 Amazon MSK 訊息的整個原始字串。
- `embedded_data`：`chunk_data` 如果不是空的（套用區塊），則從 建立內嵌向量，如果未套用區塊，`original_data` 則從 建立內嵌向量。
- `chunk_data`：只有在原始資料被區塊化時才會出現。包含用於在 中建立內嵌的原始訊息區塊 `embedded_data`。

2. 當 `source.msk.data.type` 為 時 `JSON`

- `original_data`：套用 JSON 金鑰篩選後，來自 Amazon MSK 訊息的整個原始 JSON。
- `embedded_data`：`chunk_data` 如果不是空的（套用區塊），則從 建立內嵌向量，如果未套用區塊，`original_data` 則從 建立內嵌向量。
- `chunk_key`：只有在原始資料被區塊化時才會出現。包含區塊來自的 JSON 金鑰 `original_data`。例如，它看起來像 `jsonKey1.nestedJsonKeyA` 範例中的巢狀金鑰或中繼資料 `original_data`。
- `chunk_data`：只有在原始資料被區塊化時才存在。包含用於在 中建立內嵌的原始訊息區塊 `embedded_data`。

可以，您可以使用此應用程式從多個 Amazon MSK 主題讀取資料。來自所有主題的資料必須屬於相同類型 (`STRING` 或 `JSON`)，否則可能會導致應用程式失敗。來自所有主題的資料一律存放在單一 OpenSearch 索引中。

哪裡可以找到原始程式碼的新更新？

前往 <https://github.com/aws-labs/real-time-vectorization-of-streaming-data/releases>。

我可以變更 AWS CloudFormation 範本並更新 Managed Service for Apache Flink 應用程式嗎？

否，變更 AWS CloudFormation 範本並不會更新 Managed Service for Apache Flink 應用程式。中的任何新變更都 AWS CloudFormation 意味著需要部署新的堆疊。

會代表我 AWS 監控和維護應用程式嗎？

否，AWS 不會代表您監控、擴展、更新或修補此應用程式。

這個應用程式是否會將我的資料移動到我的 之外 AWS 帳戶？

Managed Service for Apache Flink 應用程式讀取和儲存的所有資料都會保留在您的 中 AWS 帳戶，絕不會離開您的帳戶。

即時向量內嵌藍圖 - 疑難排解

檢閱下列有關即時向量內嵌藍圖的疑難排解主題。如需即時向量內嵌藍圖的詳細資訊，請參閱[即時向量內嵌藍圖](#)。

故障診斷主題

- [My CloudFormation 堆疊部署失敗或已復原。我可以做什麼來修正此問題？](#)
- [我不希望我的應用程式開始從 Amazon MSK 主題開頭讀取訊息。我要怎麼做？](#)
- [如何知道我的 Managed Service for Apache Flink 應用程式是否存在問題，以及如何對其進行偵錯？](#)
- [我應該為 Managed Service for Apache Flink 應用程式監控哪些關鍵指標？](#)

My CloudFormation 堆疊部署失敗或已復原。我可以做什麼來修正此問題？

- 前往您的 CFN 堆疊，並尋找堆疊失敗的原因。它可能與缺少許可、AWS 資源名稱衝突以及其他原因有關。修正部署失敗的根本原因。如需詳細資訊，請參閱 [CloudWatch 疑難排解指南](#)。
- **【選用】** 每個 VPC 每個服務只能有一個 VPC 端點。如果您部署了多個即時向量內嵌藍圖，以寫入相同 VPC 中的 Amazon OpenSearch Service 集合，則它們可能會共用 VPC 端點。這些可能已經存在於您的 VPC 帳戶中，或者第一個即時向量內嵌藍圖堆疊將為 Amazon Bedrock 和 Amazon OpenSearch Service 建立 VPC 端點，以供您帳戶中部署的所有其他堆疊使用。如果堆疊失敗，請檢查該堆疊是否已為 Amazon Bedrock 和 Amazon OpenSearch Service 建立 VPC 端點，如果您的帳戶中的其他位置未使用它們，請將其刪除。如需刪除 VPC 端點的步驟，請參閱如何安全地刪除應用程式的文件。
- 您的帳戶中可能有使用 VPC 端點的其他服務或應用程式。刪除它可能會對其他服務造成網路中斷。請小心刪除這些端點。

我不希望我的應用程式開始從 Amazon MSK 主題開頭讀取訊息。我要怎麼做？

您必須根據所需的行為，明確 `source.msk.starting.offset` 地將設定為下列其中一個值：

- 最早位移：分割區中最舊的位移。
- 最新偏移：消費者將從分割區結尾讀取訊息。
- 已遞交偏移：從取用者在分割區中處理的最後一個訊息中讀取。

如何知道我的 Managed Service for Apache Flink 應用程式是否存在問題，以及如何對其進行偵錯？

使用 [Managed Service for Apache Flink 疑難排解指南](#)，為您的應用程式偵錯 Managed Service for Apache Flink 相關問題。

我應該為 Managed Service for Apache Flink 應用程式監控哪些關鍵指標？

- 適用於一般 Managed Service for Apache Flink 應用程式的所有指標都可協助您監控應用程式。如需詳細資訊，請參閱 [Managed Service for Apache Flink 中的指標和維度](#)。
- 若要監控 Amazon Bedrock 指標，請參閱 [Amazon Bedrock 的 Amazon CloudWatch 指標](#)。
- 我們已新增兩個新的指標來監控產生內嵌的效能。在 CloudWatch 中的 `EmbeddingGeneration` 操作名稱下尋找它們。這兩個指標為：
 - `BedrockTitanEmbeddingTokenCount`：Amazon Bedrock 的單一請求中存在的字符數量。
 - `BedrockEmbeddingGenerationLatencyMs`：報告從 Amazon Bedrock 傳送和接收回應以產生內嵌所需的時間，以毫秒為單位。
- 對於 Amazon OpenSearch Service 無伺服器集合，您可以使用等指標 `IngestionDataRateIngestionDocumentErrors` 和其他指標。如需詳細資訊，請參閱 [使用 Amazon CloudWatch 監控 OpenSearch Serverless](#)。
- 如需 OpenSearch 佈建指標，請參閱 [使用 Amazon CloudWatch 監控 OpenSearch 叢集指標](#)。

執行期疑難排解

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](#)
- [應用程式問題](#)

- [應用程式正在重新啟動](#)
- [輸送量太慢](#)
- [無限制狀態成長](#)
- [I/O 綁定運算子](#)
- [Kinesis 資料串流的上游或來源限流](#)
- [檢查點](#)
- [檢查點逾時](#)
- [Apache Beam 應用程式檢查點失敗](#)
- [背壓](#)
- [資料扭曲](#)
- [狀態扭曲](#)
- [與不同區域中的資源整合](#)

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 指標的閾值，以指出應用程式中的錯誤或瓶頸狀況。如需建議的 CloudWatch 警示的相關資訊，請參閱[搭配 Amazon Managed Service for Apache Flink 使用 CloudWatch 警示](#)。

應用程式問題

本節包含針對 Managed Service for Apache Flink 應用程式可能會遇到的錯誤情況的解決方案。

主題

- [應用程式卡在暫時狀態](#)
- [快照建立失敗](#)
- [無法存取 VPC 中的資源](#)
- [寫入 Amazon S3 儲存貯體時遺失資料](#)
- [應用程式處於執行中狀態，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.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts](#)

應用程式卡在暫時狀態

如果應用程式已保持暫時狀態 (STARTING、UPDATING、STOPPING 或 AUTOSCALING) 一段時間，您可以使用 [StopApplication](#) 動作停止應用程式，並將 Force 參數設定為 true。您無法強制停止 DELETING 狀態的應用程式。或者，如果應用程式處於 UPDATING 或 AUTOSCALING 狀態，您可以將其復原至先前執行的版本。復原應用程式時，它會從上次成功的快照載入狀態資料。如果應用程式沒有快照，Managed Service for Apache Flink 會拒絕復原請求。如需復原應用程式的詳細資訊，請參閱 [RollbackApplication](#) 動作。

Note

強制停止應用程式可能會導致資料遺失或重複。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。
- 應用程式的程式碼太大：確認您的應用程式 JAR 檔案小於 512 MB。不支援大於 512 MB 的 JAR 檔案。
- 應用程式快照建立失敗：Managed Service for Apache Flink 會在 [UpdateApplication](#) 或 [StopApplication](#) 請求期間擷取應用程式的快照。然後，服務會使用此快照狀態，並使用更新的應用程式組態還原應用程式，以提供恰好一次的處理語義。如果自動建立快照失敗，請參閱下文的[快照建立失敗](#)。
- 從快照還原失敗：如果您移除或變更應用程式更新中的運算子，並嘗試從快照還原，則如果快照包含遺失運算子的狀態資料，還原預設將會失敗。此外，應用程式將卡在 STOPPED 或 UPDATING 狀態。若要變更此行為並讓還原成功，請將應用程式 [FlinkRunConfiguration](#) 的 `AllowNonRestoredState` 參數變更為 true。這將允許恢復操作跳過無法對應至新程式的狀態資料。
- 應用程式初始化所花費的時間較長：Managed Service for Apache Flink 在等待 Flink 作業啟動時，會使用 5 分鐘的內部逾時 (軟體設定)。如果作業無法在此逾時內啟動，您將會看到如下所示的 CloudWatch 日誌：

```
Flink job did not start within a total timeout of 5 minutes for application: %s under account: %s
```


如果遇到上述錯誤，表示在 Flink 作業 main 方法下定義的操作需要 5 分鐘以上的時間，從而導致建立 Flink 作業的操作在 Managed Service for Apache Flink 結束時逾時。我們建議您檢查 Flink JobManager 日誌以及應用程式的程式碼，查看 main 方法中是否預期存在此延遲。如果沒有，則需要採取措施來解決該問題，以便在 5 分鐘內完成。

您可以使用 [ListApplications](#) 或 [DescribeApplication](#) 動作來檢查應用程式狀態。

快照建立失敗

在下列情況下，Managed Service for Apache Flink 服務無法拍攝快照：

- 應用程式超過快照限制。快照的限制為 1,000。如需詳細資訊，請參閱[使用快照管理應用程式備份](#)。
- 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。

無法存取 VPC 中的資源

如果應用程式使用在 Amazon VPC 上執行的 VPC，請執行以下操作以確認應用程式是否可以存取其資源：

- 檢查 CloudWatch 日誌中是否有下列錯誤。此錯誤表示應用程式無法存取 VPC 中的資源：

```
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.
```

如果您看到此錯誤，請確認您的路由表設定正確，並且連接器具有正確的連線設定。

如需設定和分析 CloudWatch 日誌的相關資訊，請參閱[在 Amazon Managed Service for Apache Flink 中記錄和監控](#)。

寫入 Amazon S3 儲存貯體時遺失資料

使用 Apache Flink 1.6.2 版將輸出寫入 Amazon S3 儲存貯體時，可能會發生一些資料遺失。我們建議您在直接使用 Amazon S3 進行輸出時，使用 Apache Flink 最新支援版本。若要使用 Apache Flink 1.6.2 寫入 Amazon S3 儲存貯體，建議使用 Firehose。如需搭配使用 Firehose 與 Managed Service for Apache Flink 的詳細資訊，請參閱[Firehose 接收器](#)。

應用程式處於執行中狀態，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。如果應用程式進入 RUNNING 狀態，但未在將資料寫入接收器，您可以透過將 Amazon CloudWatch 日誌串流新增至應用程式來解決該問題。如需詳細資訊，請參閱[使用應用程式 CloudWatch 記錄選項](#)。日誌串流包含可用於對應用程式問題進行疑難排解的訊息。

快照、應用程式更新或應用程式停止錯誤：InvalidApplicationConfigurationException

在快照操作期間或在建立快照的操作 (例如更新或停止應用程式) 期間，可能會發生如下錯誤：

```
An error occurred (InvalidApplicationConfigurationException) when calling the
UpdateApplication operation:
```

```
Failed to take snapshot for the application xxxx at this moment. The application is
currently experiencing downtime.
```

```
Please check the application's CloudWatch metrics or CloudWatch logs for any possible
errors and retry the request.
```

```
You can also retry the request after disabling the snapshots in the Managed Service for
Apache Flink console or by updating
the ApplicationSnapshotConfiguration through the AWS SDK
```

應用程式無法建立快照時，會發生此錯誤。

如果在快照操作或建立快照的操作期間遇到此錯誤，請執行下列動作：

- 為應用程式停用快照。您可以在 Managed Service for Apache Flink 主控台中執行此操作，也可以使用 [UpdateApplication](#) 動作的 `SnapshotsEnabledUpdate` 參數。
- 調查無法建立快照的原因。如需詳細資訊，請參閱[應用程式卡在暫時狀態](#)。

- 當應用程式恢復正常狀態時，再重新啟用快照。

java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

SSL 信任存放區的位置已在先前的部署中更新。為 `ssl.truststore.location` 參數改用下列值：

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

應用程式正在重新啟動

如果您的應用程式運作狀況不正常，其 Apache Flink 作業就會持續失敗並重新啟動。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

這種情況可能有下列徵狀：

- FullRestarts 指標不為零。此指標代表自您啟動應用程式後，應用程式作業已重新啟動的次數。
- Downtime 指標不為零。此指標代表應用程式處於 FAILING 或 RESTARTING 狀態的毫秒數。
- 應用程式日誌包含狀態變更 (變更為 RESTARTING 或 FAILED)。您可以使用下列 CloudWatch Logs Insights 查詢，來查詢您的應用程式日誌中是否有這些狀態變更：[分析錯誤：應用程式任務相關的失敗](#)。

原因和解決方案

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：

- 運算子擲回例外狀況：如果應用程式中運算子中的任何例外狀況未處理，應用程式會容錯移轉（透過解譯運算子無法處理失敗）。應用程式會從最新的檢查點重新啟動，以維護「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碼中的任何可重試的例外狀況。

您可以查詢應用程式日誌中是否包含從 RUNNING 到 FAILED 的應用程式狀態變更，以調查此情況的原因。如需詳細資訊，請參閱[the section called “分析錯誤：應用程式任務相關的失敗”](#)。

- 未正確佈建 Kinesis 資料串流：如果您應用程式的來源或目的地是 Kinesis 資料串流，請檢查串流的[指標](#)是否有 ReadProvisionedThroughputExceeded 或 WriteProvisionedThroughputExceeded 錯誤。

如果看到這些錯誤，您可以增加串流的碎片數目，以增加 Kinesis 串流的可用輸送量。如需詳細資訊，請參閱[如何變更 Kinesis Data Streams 中的開放碎片數目？](#)

- 其他來源或接收器未正確佈建或不可用：確認應用程式是否正確佈建來源和接收器。檢查應用程式中使用的任何來源或接收器（例如 AWS 其他服務，或外部來源或目的地）是否已妥善佈建、未遇到讀取或寫入限流，或是否定期無法使用。

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，出現 `DaemonException`：如果您使用的是 1.11 之前的 Apache Flink 版本，此錯誤會出現在應用程式日誌中。您可能需要升級至更新版本的 Apache Flink，以便使用 0.14 或更新版本的 KPL。
- 應用程式失敗，出現 `TimeoutException`、`FlinkException` 或 `RemoteTransportException`：如果任務管理員損毀，這些錯誤可能會出現在應用程式日誌中。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 CloudWatch 指標以了解 CPU 或記憶體用量是否出現異常尖峰。
- 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[對效能問題進行故障診斷](#)。
- 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，出現「找不到 JaxbAnnotationModule」錯誤：如果應用程式使用 Apache Beam，但沒有正確的相依性或相依性版本，則會發生此錯誤。使用 Apache Beam 的 Managed Service for Apache Flink 應用程式必須使用以下版本的相依性：

```
<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[使用 CloudFormation](#)。

- 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，則叢集的取用者延遲指標會持續增加。如需詳細資訊，請參閱 [Amazon MSK 開發人員指南](#) 中的 [取用者延遲監控](#)。
- 如果應用程式的資料來源是其他服務或來源，請檢查任何可用的取用者延遲指標或可用資料。

原因和解決方案

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[應用程式正在重新啟動](#)。
- 如果輸送量延遲一致，請檢查應用程式是否已進行效能最佳化。如需最佳化應用程式效能的相關資訊，請參閱[對效能問題進行故障診斷](#)。
- 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[實作應用程式擴展](#)。
- 如果應用程式從不同區域的 Kafka 叢集讀取，並且儘管取用者延遲很高，FlinkKafkaConsumer 或 KafkaSource 大多是閒置狀態 (高 idleTimeMsPerSecond 或低 CPUUtilization)，則可以增加 `receive.buffer.byte` 的值，例如 2097152。如需詳細資訊，請參閱[自訂 MSK 組態](#)中的「高延遲環境」一節。

如需應用程式來源中輸送量緩慢或取用者延遲增加的疑難排解步驟，請參閱[對效能問題進行故障診斷](#)。

無限制狀態成長

如果應用程式未正確處置過期的狀態資訊，這些資訊會持續累積並導致應用程式效能或穩定性問題。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

這種情況可能有下列徵狀：

- `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Time-To-Live\(TTL\)](#)。

- 您正在執行的應用程式相依於 Apache Beam 2.25.0 版或更高版本。您可以透過用關鍵實驗和 `use_deprecated_read` 值擴充 [BeamApplicationProperties](#)，選擇退出讀取轉換的新版本。如需詳細資訊，請參閱 [Apache Beam 文件](#)。

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。除非使用允許增量彙總的 [AggregateFunctions](#)，否則 Flink 需要將整個視窗的事件保持在狀態。

此外，當使用進程函數來實作自訂運算子時，應用程式需要從業務邏輯不再需要的狀態中移除資料。在這種情況下，[狀態存活期](#)可用於根據處理時間自動老化資料。Managed Service for Apache Flink 正在使用增量檢查點，因此狀態 TTL 基於 [RocksDB 壓縮](#)。在壓縮操作發生之後，您只能觀察到狀態大小的實際縮減 (由檢查點大小表示)。特別是對於低於 200 MB 的檢查點大小，由於狀態到期，您不太可能觀察到任何檢查點大小縮小。儲存點則基於不包含舊資料之狀態的全新副本，因此您可以在 Managed Service for Apache Flink 中觸發快照，以強制移除過期狀態。

出於偵錯目的，停用增量檢查點以更快速地驗證檢查點大小是否確實減小或穩定 (並避免 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 綁定運算子的應用程式，也可以增加 Apache Flink 應用程式的 [ParallelismPerKPU](#) 設定。此設定描述應用程式每 Kinesis 處理單元 (KPU) 可執行的平行子任務數目。藉由將值從預設值 1 增加到 4，應用程式利用相同的資源 (以相同的成本)，但可擴展至 4 倍的平行處理層級。這非常適用於 I/O 綁定的應用程式，但會給不是 I/O 綁定的應用程式造成額外開銷。

Kinesis 資料串流的上游或來源限流

徵狀：應用程式遇到來自其上游來源 Kinesis 資料串流的 `LimitExceededExceptions`。

潛在原因：Apache Flink 程式庫 Kinesis 連接器的預設設定設定為從 Kinesis 資料串流來源讀取，且每次 `GetRecords` 呼叫擷取的最大記錄數目具有非常積極的預設設定。Apache Flink 預設設定為每

次GetRecords呼叫擷取 10,000 個記錄（此呼叫預設為每 200 毫秒），但每個碎片的限制只有 1,000 個記錄。

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。

您可以透過檢查 CloudWatch ReadProvisionedThroughputExceeded 指標並查看此指標大於零的延長或持續期間來確認這一點。

您也可以透過觀察持續的LimitExceededException錯誤，在 Amazon Managed Service for Apache Flink 應用程式的 CloudWatch 日誌中看到此情況。

解決方法：您可以執行下列兩項操作之一來解決這種情況：

- 降低每次GetRecords呼叫擷取的記錄數量的預設限制
- 在 Amazon Managed Service for Apache Flink 應用程式中啟用自適應讀取。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 中的檢查點運算子任務，有 5 個主要階段：

- 等待 [開始延遲]：Flink 使用插入串流的檢查點障礙，因此在此階段的時間是運算子等待檢查點障礙到達它的時間。
- 對齊 [對齊持續時間]：在此階段，子任務已到達一個障礙，但它正在等待來自其他輸入串流的障礙。
- 同步檢查點 [同步持續時間]：在此階段，子任務會實際拍攝運算子狀態快照，並阻止該子任務上的所有其他活動。
- 非同步檢查點 [非同步持續時間]：此階段的主要操作是子任務將狀態上傳到 Amazon S3。在此階段，子任務不再被阻止，可以處理記錄。

- 確認：這通常是一個短暫的階段，只是子任務發送確認給 JobManager 並執行任何遞交訊息 (例如，使用 Kafka 接收器)。

上述每個階段 (除了「確認」) 都對應到 Flink WebUI 中可用檢查點的持續時間指標，這可以幫助隔離長檢查點的原因。

要查看檢查點上每個可用指標的確切定義，請轉到[歷史記錄](#)標籤。

調查

調查長檢查點的持續時間時，最重要的是要確定檢查點的瓶頸，也就是說，什麼運算子和子任務正在採用最長檢查點，該子任務的哪個階段正在花費較長的時間。這可以使用作業檢查點任務下的 Flink WebUI 來確定。Flink 的 Web 介面提供了可協助調查檢查點問題的資料和資訊。如需完整明細，請參閱[監控檢查點](#)。

首先要注意的是作業圖表中每個運算子的端對端持續時間，以確定哪個運算子需要較長時間才能到達檢查點，需要進一步調查。根據 Flink 文件，持續時間的定義如下：

從觸發時間戳記到最近確認為止的持續時間 (如果尚未收到確認，則為 n/a)。完整檢查點的端對端持續時間由確認檢查點的最後一個子任務決定。此時間通常大於單個子任務對狀態實際執行檢查點需要的時間。

檢查點的其他持續時間還提供了有關花費時間的更精細資訊。

如果同步持續時間很高，則表示快照過程中發生了問題。在這個階段，為實作 snapshotState 介面的類別呼叫 snapshotState()；這可以是使用者程式碼，所以執行緒傾印對於調查這一點會有幫助。

非同步持續時間長表明將狀態上傳到 Amazon S3 花費了大量時間。如果狀態很大，或者有許多狀態檔案正在上傳，就會發生這種情況。如果是這種情況，則值得調查應用程式如何使用狀態，並確保在可能的情況下使用 Flink 本機資料結構 ([使用具有索引鍵的狀態](#))。Managed Service for Apache Flink 會以最小化 Amazon S3 呼叫數目的方式來設定 Flink，以確保不會變得太長。下面是某個運算子的檢查點統計資料範例。它表明，與之前的運算子檢查點統計資料相比，此運算子的非同步持續時間相對較長。

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	
I D	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。以下是作業圖表範例，其中第二個 KeyedProcess 運算子處於忙碌狀態。



您可以使用 Flink 火焰圖或 TaskManager 執行緒傾印來調查是什麼需要這麼長時間。一旦確定了瓶頸，就可以使用火焰圖或執行緒傾印進一步調查。

執行緒傾印

執行緒傾印是比火焰圖層級略低的另一種偵錯工具。執行緒傾印會在某個時間點輸出所有執行緒的執行狀態。Flink 接受 JVM 執行緒傾印，這是 Flink 處理序中所有執行緒的執行狀態。執行緒狀態由執行緒的堆疊追蹤以及一些附加資訊來表示。火焰圖實際上是使用快速連續採取的多個堆疊追蹤所建置。該圖形是由這些追蹤構成的可視化呈現，可讓您輕鬆地識別常見程式碼路徑。

```
"KeyedProcess (1/3)#0" prio=5 Id=1423 RUNNABLE
```

```

    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:154)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>>19)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(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(AbstractStreamTaskNet
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcess
    ...

```

以上是從 Flink UI 為單個執行緒取得的執行緒傾印的片段。第一行包含有關此執行緒的一些一般資訊，包括：

- 執行緒名稱 KeyedProcess (1/3)#0
- 執行緒優先順序 prio=5
- 唯一的執行緒 ID Id=1423
- 執行緒狀態 RUNNABLE

執行緒名稱通常會提供執行緒一般用途的資訊。運算子執行緒可以通過其名稱來識別，因為運算子執行緒與運算子具有相同的名稱，並且會指出其相關子任務，例如，KeyedProcess (1/3)#0 執行緒來自 KeyedProcess 運算子，並且來自第 1 個子任務 (共 3 個)。

執行緒可以是下列幾種狀態之一：

- NEW：執行緒已建立，但尚未得到處理
- RUNNABLE：執行緒正在 CPU 上執行
- BLOCKED：執行緒正在等待另一個執行緒釋放其鎖定
- WAITING：執行緒正在使用 wait()、join() 或 park() 方法等待
- TIMED_WAITING：執行緒正在使用睡眠、等待、聯結或駐留方法等待，但等待時間最長。

Note

在 Flink 1.13 中，執行緒傾印中單一堆疊追蹤的最大深度限制為 8。

Note

執行緒傾印必須是 Flink 應用程式中偵錯效能問題的最後手段，因為它們可能難以讀取，需要擷取和手動分析多個樣本。如果有可能，最好使用火焰圖。

Flink 中的執行緒傾印

在 Flink 中，透過選擇 Flink UI 左側導覽列上的任務管理員選項，選取特定任務管理員，然後瀏覽至執行緒傾印標籤，即可取得執行緒傾印。您可以下載執行緒傾印、複製到喜愛的文字編輯器 (或執行緒傾印分析器)，或直接在 Flink Web UI 的文字檢視中進行分析 (不過最後一個選項可能有點繁瑣)。

為了確定在選擇特定運算子後，要用來取得 TaskManager 選項卡執行緒傾印的任務管理員。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。

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics	FlameGraph
Host	LOG	Bytes received	Records received	Bytes sent	Records sent	Status	
ip-142-151-131-22:61 21	LOG	936 B	0	0 B	0	RUNNING	
ip-142-151-146-195:6 121	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，以下堆疊追蹤來自 KeyedProcess 運算子，並且是第 1 個子任務。

```
"KeyedProcess (1/3)#0" prio=5 Id=595 RUNNABLE
  at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:155)
```

```

    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(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(AbstractStreamTaskNetwo
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcesso
    ...

```

如果有多個運算子具有相同名稱，則可能會造成混淆，但我們可以透過命名運算子來解決這個問題。例如：

```

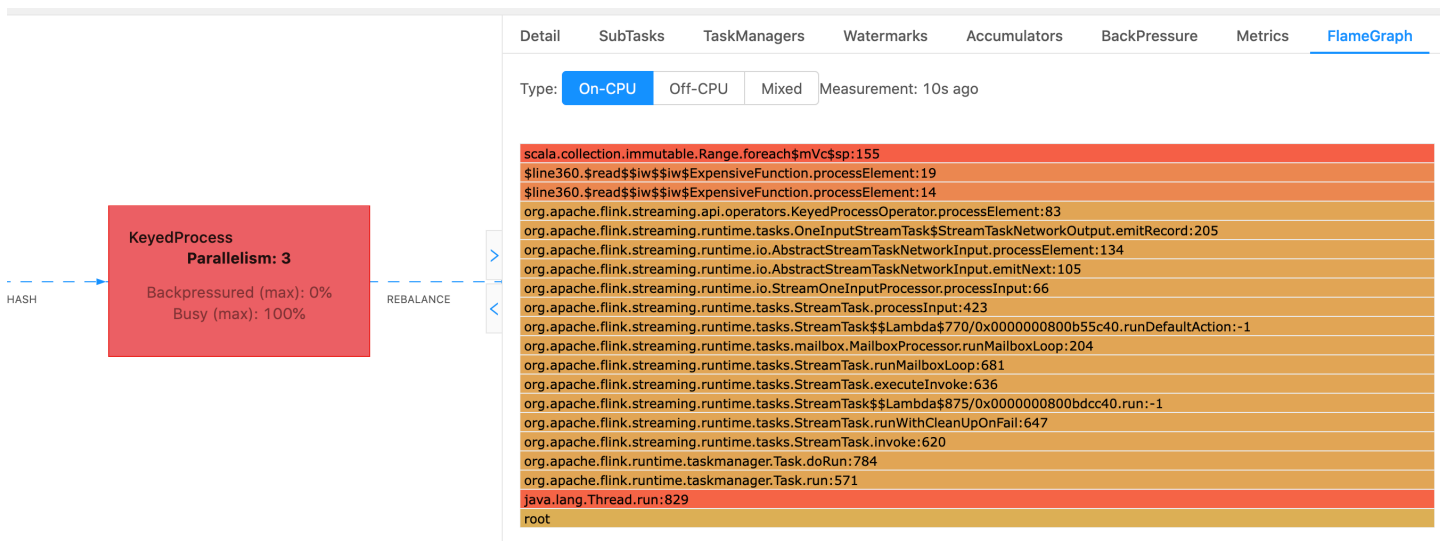
....
.process(new ExpensiveFunction).name("Expensive function")

```

火焰圖

火焰圖是一款有用的偵錯工具，它可以可視化目標程式碼的堆疊追蹤，從而允許識別最常見的程式碼路徑。它們透過對堆疊追蹤進行多次取樣來建立。火焰圖的 x 軸顯示不同的堆疊設定檔，y 軸顯示堆疊深度，以及堆疊追蹤中的呼叫。火焰圖中的單個矩形顯示在堆疊框架上，框架的寬度顯示它在堆疊中出現的頻率。如需火焰圖表及其用法的詳細資訊，請參閱[火焰圖](#)。

在 Flink 中，運算子的火焰圖可以透過 Web UI 存取，方法是選取運算子，然後選擇火焰圖標籤。一旦收集到足夠的樣本，火焰圖即會顯示。以下是花費了大量時間執行檢查點的 ProcessFunction 的火焰圖。



這是一個非常簡單的火焰圖，其中顯示了所有 CPU 時間都花費在一個 foreach 迴圈內的 ExpensiveFunction 運算子的 processElement 內。您還可以取得行號，以幫助確定程式碼的執行位置。

檢查點逾時

如果應用程式未最佳化或正確佈建，檢查點可能會失敗。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

如果應用程式的檢查點失敗，numberOfFailedCheckpoints 將會大於零。

檢查點可能會因為直接失敗 (例如應用程式錯誤) 或暫時性失敗 (例如應用程式資源不足) 而失敗。檢查應用程式日誌和指標是否有下列徵狀：

- 程式碼中有錯誤。
- 存取應用程式相依服務時發生錯誤。
- 序列化資料時發生錯誤。如果預設的序列化程式無法序列化應用程式資料，則應用程式將失敗。如需有關在應用程式中使用自訂序列化程式的資訊，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`

如需監控檢查點的詳細資訊，請參閱 Apache Flink 文件中的[監控檢查點](#)。

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：

- 應用程式佈建的 KPU 不足。如需增加應用程式佈建的相關資訊，請參閱[實作應用程式擴展](#)。
- 應用程式狀態大小太大。您可以使用 `lastCheckpointSize` 指標監控應用程式狀態大小。
- 應用程式的狀態資料在索引鍵之間分配不平均。如果應用程式使用 `KeyBy` 運算子，請確保您的傳入資料在索引鍵之間已平均分割。如果將大部分資料指派給單一索引鍵，則會產生瓶頸，從而導致失敗。
- 應用程式遇到記憶體背壓或垃圾回收背壓。監控應用程式的 `heapMemoryUtilization`、`oldGenerationGCtime` 以及 `oldGenerationGCCount`，看是否有值在急劇增加或穩定增加。

Apache Beam 應用程式檢查點失敗

如果設定 Beam 應用程式時將 `shutdownSourcesAfterIdleMs` 設定為 0ms，則檢查點可能無法觸發，因為任務處於「FINISHED」狀態。本節說明此狀況的徵狀和解決方案。

徵狀

前往 Managed Service for Apache Flink 應用程式 CloudWatch 日誌，檢查其中是否記錄了下列日誌訊息。下列日誌訊息指出檢查點無法觸發，因為某些任務已完成。

```
{
  "locationInformation":
    "org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(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，其中一些任務已進入「FINISHED」狀態，並且無法再執行檢查點。

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，可關閉閒置了一段設定時間 (毫秒) 的來源。一旦來源關閉，無法再執行檢查點。這可能導致[檢查點失敗](#)。

任務進入「FINISHED」狀態的其中一個原因是當 shutdownSourcesAfterIdleMs 設定為 0ms 時，意味著閒置的任務將立即關閉。

解決方案

若要防止任務立即進入「FINISHED」狀態，請將 shutdownSourcesAfterIdleMs 設定為 Long.MAX_VALUE。這可以透過兩種方式進行：

- 選項 1：如果 Beam 組態是在 Managed Service for Apache Flink 應用程式的組態頁面中設定，則可以新增一個鍵值對來設定 shutdownSourcesAfterIdleMs，如下所示：

Runtime properties (6)		
You can also group application properties into multiple groups. These are useful to store configuration settings without the need to change application code.		
Find groups, keys, and values		
Group	Key	Value
BeamApplicationProperties	ShutdownSourcesAfterIdleMs	9223372036854775807

- 選項 2：如果 Beam 組態是在 JAR 檔案中設定，您可以按如下方式設定 shutdownSourcesAfterIdleMs：


```
FlinkPipelineOptions options =  
PipelineOptionsFactory.create().as(FlinkPipelineOptions.class); // Initialize Beam  
Options object  
  
options.setShutdownSourcesAfterIdleMs(Long.MAX_VALUE); // set  
shutdownSourcesAfterIdleMs to Long.MAX_VALUE  
options.setRunner(FlinkRunner.class);  
  
Pipeline p = Pipeline.create(options); // attach specified  
options to Beam pipeline
```

背壓

Flink 使用背壓來調整個別運算子的處理速度。

由於許多原因，運算子可能難以跟上處理收到的訊息量。該操作可能需要比運算子可用資源更多的 CPU 資源，運算子可能會等待 I/O 操作完成。如果運算子無法以足夠快的速度處理事件，它會在饋送給慢速運算子的上游運算子中造成背壓。這會導致上游運算子減慢速度，從而進一步將背壓傳播到來源，並透過減慢速度來使來源適應應用程式的整體輸送量。您可以在 [Apache Flink™ 如何處理背壓](#) 中找到有關背壓及其運作方式的更深入說明。

知道應用程式中的哪些運算子速度緩慢，可為您提供重要資訊來了解應用程式效能問題的根本原因。背壓資訊[透過 Flink 控制面板公開](#)。若要識別慢速運算子，請尋找具有最接近接收器高背壓值的運算子（以下範例中為運算子 B）。造成緩慢的運算子就是其中一個下游運算子（範例中的運算子 C）。B 可以更快地處理事件，但由於無法將輸出轉發到實際的慢速運算子 C，因此會受到背壓。

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D  
(backpressured 0%)
```

一旦確定慢速運算子，試著了解它為什麼慢。可能有無數的原因，有時出了什麼問題並不明顯，可能需要數天的偵錯和分析來解決。以下是一些明顯和更常見的原因，其中一些進一步解釋如下：

- 運算子正在執行緩慢的 I/O，例如網路呼叫（考慮改用 AsyncIO）。
- 存在資料扭曲，一個運算子接收的事件比其他運算子多，可查看 Flink 儀表板中單個子任務（即同一運算子的多個執行個體）的進/出訊息數目進行驗證。
- 這是一個消耗資源的操作（如果沒有資料扭曲，請考慮橫向擴展 CPU/記憶體綁定的工作或增加 I/O 綁定工作的 ParallelismPerKPU）

- 運算子中存在大量記錄 (將生產應用程式的記錄減少到最低限度，或者考慮將偵錯輸出發送到資料串流)。

使用捨棄接收器測試輸送量

[丟棄接收器](#)只是忽略它在仍在執行應用程式時收到的所有事件 (沒有任何接收器的應用程式無法執行)。這對於輸送量測試、分析以及驗證應用程式是否正確擴展非常有用。這也是一種非常簡潔實用的完整性檢查，可以驗證接收器是否給應用程式導致背壓 (不過直接檢查背壓指標通常更容易和更直接)。

您可以透過用捨棄的接收器取代應用程式的所有接收器，並建立可產生與生產資料類似資料的模擬來源，來衡量應用程式在特定平行處理設定時的最大輸送量。然後，您也可以增加平行處理層級，以驗證應用程式是否正確擴展，並且沒有只在較高輸送量 (例如由於資料扭曲) 時才會出現的瓶頸。

資料扭曲

Flink 應用程式在叢集上分散式執行。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字任務層級的 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，某些字任務比其他字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字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 中，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。

與不同區域中的資源整合

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 支援中心](#) 填寫支援票證。

Amazon Managed Service for Apache Flink 的文件歷史記錄

下表說明自 Managed Service for Apache Flink 推出上一個版本以來後，文件內所進行的重大變更。

變更	描述	Date
支援 Apache Flink 2.2 版	Amazon Managed Service for Apache Flink 現在支援 Apache Flink 2.2 版。這是具有重大變更的主要版本升級。檢閱 中的重大變更和新功能 Amazon Managed Service for Apache Flink 2.2 。	2026 年 3 月 31 日
支援 AWS KMS 客戶受管金鑰	Amazon Managed Service for Apache Flink (Amazon MSF) 現在支援使用 AWS KMS 客戶受管金鑰 (CMKs) 加密靜態應用程式資料。這適用於執行 Apache Flink 1.20 的新應用程式或現有應用程式。如需詳細資訊，請參閱 Amazon Managed Service for Apache Flink 中的金鑰管理 。	2025 年 8 月 20 日
支援 Apache Flink 1.15.2 版	Managed Service for Apache Flink 現支援使用 Apache Flink 1.15.2 版的應用程式。使用 Apache Flink 資料表 API 建立 Kinesis Data Analytics 應用程式。如需詳細資訊，請參閱 建立應用程式 。	2022 年 11 月 22 日
支援 Apache Flink 1.13.2 版	Managed Service for Apache Flink 現支援使用 Apache Flink 1.13.2 版的應用程式。使用 Apache Flink 資料表 API 建	2021 年 10 月 13 日

變更	描述	Date
	立 Kinesis Data Analytics 應用程式。如需詳細資訊，請參閱 Flink 1.13.2 入門 。	
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 使用 Python 。	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 建立應用程式 。	2020 年 11 月 19 日
Apache Flink 儀表板	使用 Apache Flink 儀表板來監控應用程式運作狀態和效能。如需詳細資訊，請參閱 使用 Apache Flink 儀表板 。	2020 年 11 月 19 日
增強型扇出 (EFO) 取用者	建立使用增強型扇出 (EFO) 取用者從 Kinesis 資料串流讀取的應用程式。如需詳細資訊，請參閱 增強型扇出 (EFO) 取用者 。	2020 年 10 月 6 日
Apache Beam	建立使用 Apache Beam 處理串流資料的應用程式。如需詳細資訊，請參閱 使用 CloudFormation 。	2020 年 9 月 15 日

變更	描述	Date
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 ??? 。	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 自訂信任存放區 。	2020 年 6 月 10 日
CloudWatch 警示	有關使用 Managed Service for Apache Flink 建立 CloudWatch 警示的建議。如需詳細資訊，請參閱 ??? 。	2020 年 6 月 5 日
新的 CloudWatch 指標	Managed Service for Apache Flink 現在向 Amazon CloudWatch 指標發出 22 個指標。如需詳細資訊，請參閱 ??? 。	2020 年 5 月 12 日
CloudWatch 自訂指標	定義應用程式特定的指標，並將其發送至 Amazon CloudWatch 指標。如需詳細資訊，請參閱 ??? 。	2020 年 5 月 12 日
範例：從不同帳戶的 Kinesis 串流中讀取	了解如何在 Managed Service for Apache Flink 應用程式中存取不同 AWS 帳戶中的 Kinesis 串流。如需詳細資訊，請參閱 跨帳戶 。	2020 年 3 月 30 日

變更	描述	Date
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 建立應用程式 。	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 設定 MSF 以存取 Amazon VPC 中的資源 。	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 ??? 。	2019 年 10 月 14 日
分析 Managed Service for Apache Flink 應用程式日誌	使用 CloudWatch Logs Insights 監控 Managed Service for Apache Flink 應用程式。如需詳細資訊，請參閱 ??? 。	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 使用執行期屬性 。	2019 年 6 月 24 日
標記 Managed Service for Apache Flink 應用程式	使用應用程式標記來判斷每個應用程式的成本、控制存取或用於使用者定義之目的。如需詳細資訊，請參閱 將標籤新增至 Managed Service for Apache Flink 應用程式 。	2019 年 5 月 8 日

變更	描述	Date
使用 記錄 Managed Service for Apache Flink API 呼叫 AWS CloudTrail	Managed Service for Apache Flink 已與 整合 AWS CloudTrail，此服務提供使用者、角色或 Managed Service for Apache Flink 中 AWS 服務所採取動作的記錄。如需詳細資訊，請參閱 ??? 。	2019 年 3 月 22 日
建立應用程式 (Firehose Sink)	練習建立 Managed Service for Apache Flink，並將 Amazon Kinesis 資料串流做為來源，並將 Amazon Data Firehose 串流做為接收器。如需詳細資訊，請參閱 Firehose 接收器 。	2018 年 12 月 13 日
公開發行	這是《適用於 Java 應用程式的 Managed Service for Apache Flink 開發人員指南》的初始版本。	2018 年 11 月 27 日

Managed Service for Apache Flink API 範例程式碼

本主題包含適用於 Managed Service for Apache Flink 動作的範例請求區塊。

若要使用 JSON 做為 AWS Command Line Interface (AWS CLI) 動作的輸入，請將請求儲存在 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
```

如需搭配使用 JSON 的詳細資訊 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[產生 CLI 骨架和 CLI 輸入 JSON 參數](#)。

主題

- [AddApplicationCloudWatchLoggingOption](#)
- [AddApplicationInput](#)
- [AddApplicationInputProcessingConfiguration](#)
- [AddApplicationOutput](#)
- [AddApplicationReferenceDataSource](#)
- [AddApplicationVpcConfiguration](#)
- [CreateApplication](#)
- [CreateApplicationSnapshot](#)
- [DeleteApplication](#)
- [DeleteApplicationCloudWatchLoggingOption](#)
- [DeleteApplicationInputProcessingConfiguration](#)
- [DeleteApplicationOutput](#)
- [DeleteApplicationReferenceDataSource](#)
- [DeleteApplicationSnapshot](#)
- [DeleteApplicationVpcConfiguration](#)
- [DescribeApplication](#)
- [DescribeApplicationSnapshot](#)

- [DiscoverInputSchema](#)
- [ListApplications](#)
- [ListApplicationSnapshots](#)
- [StartApplication](#)
- [StopApplication](#)
- [UpdateApplication](#)

AddApplicationCloudWatchLoggingOption

[AddApplicationCloudWatchLoggingOption](#) 動作的下列範例請求程式碼會將 Amazon CloudWatch 日誌選項新增至 Managed Service for Apache Flink 應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOption": {
    "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
  },
  "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

[AddApplicationInput](#) 動作的下列範例請求程式碼會將應用程式輸入新增至 Managed Service for Apache Flink 應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Input": {
    "InputParallelism": {
      "Count": 2
    },
    "InputSchema": {
      "RecordColumns": [
        {
          "Mapping": "$.TICKER",
```

```

        "Name": "TICKER_SYMBOL",
        "SqlType": "VARCHAR(50)"
    },
    {
        "SqlType": "REAL",
        "Name": "PRICE",
        "Mapping": "$.PRICE"
    }
],
"RecordEncoding": "UTF-8",
"RecordFormat": {
    "MappingParameters": {
        "JSONMappingParameters": {
            "RecordRowPath": "$"
        }
    },
    "RecordFormatType": "JSON"
},
"KinesisStreamsInput": {
    "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
}
}
}

```

AddApplicationInputProcessingConfiguration

[AddApplicationInputProcessingConfiguration](#) 動作的下列範例請求程式碼會將應用程式輸入處理組態新增至 Managed Service for Apache Flink 應用程式：

```

{
    "ApplicationName": "MyApplication",
    "CurrentApplicationVersionId": 2,
    "InputId": "2.1",
    "InputProcessingConfiguration": {
        "InputLambdaProcessor": {
            "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
        }
    }
}

```

AddApplicationOutput

[AddApplicationOutput](#) 動作的下列範例請求程式碼會將 Kinesis 資料串流作為應用程式輸出新增至 Managed Service for Apache Flink 應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Output": {
    "DestinationSchema": {
      "RecordFormatType": "JSON"
    },
    "KinesisStreamsOutput": {
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
    },
    "Name": "DESTINATION_SQL_STREAM"
  }
}
```

AddApplicationReferenceDataSource

[AddApplicationReferenceDataSource](#) 動作的下列範例請求程式碼會將 CSV 應用程式參考資料來源新增至 Managed Service for Apache Flink 應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceDataSource": {
    "ReferenceSchema": {
      "RecordColumns": [
        {
          "Mapping": "$.TICKER",
          "Name": "TICKER",
          "SqlType": "VARCHAR(4)"
        },
        {
          "Mapping": "$.COMPANYNAME",
          "Name": "COMPANY_NAME",
          "SqlType": "VARCHAR(40)"
        }
      ]
    }
  }
}
```

```

    "RecordEncoding": "UTF-8",
    "RecordFormat": {
      "MappingParameters": {
        "CSVMappingParameters": {
          "RecordColumnDelimiter": " ",
          "RecordRowDelimiter": "\r\n"
        }
      },
      "RecordFormatType": "CSV"
    },
    "S3ReferenceDataSource": {
      "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
      "FileKey": "TickerReference.csv"
    },
    "TableName": "string"
  }
}

```

AddApplicationVpcConfiguration

[AddApplicationVpcConfiguration](#) 動作的下列範例請求程式碼會將 VPC 組態新增至現有的應用程式：

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfiguration": {
    "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
    "SubnetIds": [ "subnet-0123456789abcdef0" ]
  }
}

```

CreateApplication

[CreateApplication](#) 動作的下列範例請求程式碼會建立 Managed Service for Apache Flink 應用程式：

```

{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
  "RuntimeEnvironment": "FLINK-1_15",

```

```

"ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
"CloudWatchLoggingOptions":[
  {
    "LogStreamARN":"arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-
stream:My-LogStream"
  }
],
"ApplicationConfiguration": {
  "EnvironmentProperties":
    {"PropertyGroups":
      [
        {"PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap":
            {"aws.region": "us-east-1",
             "flink.stream.initpos": "LATEST"}
        },
        {"PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap":
            {"aws.region": "us-east-1"}
        },
      ]
    },
  "ApplicationCodeConfiguration":{
    "CodeContent":{
      "S3ContentLocation":{
        "BucketARN":"arn:aws:s3:::amzn-s3-demo-bucket",
        "FileKey":"myflink.jar",
        "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    },
    "CodeContentType":"ZIPFILE"
  },
  "FlinkApplicationConfiguration":{
    "ParallelismConfiguration":{
      "ConfigurationType":"CUSTOM",
      "Parallelism":2,
      "ParallelismPerKPU":1,
      "AutoScalingEnabled":true
    }
  }
}

```

CreateApplicationSnapshot

[CreateApplicationSnapshot](#) 動作的下列範例請求程式碼會建立應用程式狀態的快照：

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

DeleteApplication

[DeleteApplication](#) 動作的下列範例請求程式碼會刪除 Managed Service for Apache Flink 應用程式：

```
{"ApplicationName": "MyApplication",
  "CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

[DeleteApplicationCloudWatchLoggingOption](#) 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中刪除 Amazon CloudWatch 記錄選項：

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOptionId": "3.1"
  "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

[DeleteApplicationInputProcessingConfiguration](#) 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除輸入處理組態：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "InputId": "2.1"
}
```

DeleteApplicationOutput

[DeleteApplicationOutput](#) 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除應用程式輸出：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

[DeleteApplicationReferenceDataSource](#) 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除應用程式參考資料來源：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

[DeleteApplicationSnapshot](#) 動作的下列範例請求程式碼會刪除應用程式狀態的快照：

```
{
  "ApplicationName": "MyApplication",
  "SnapshotCreationTimestamp": 12345678912,
  "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

[DeleteApplicationVpcConfiguration](#) 動作的下列範例請求程式碼會從應用程式中移除現有的 VPC 組態：

```
{
```



```
"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 9,
"VpcConfigurationId": "1.1"
}
```

DescribeApplication

[DescribeApplication](#) 動作的下列範例請求程式碼會傳回 Managed Service for Apache Flink 應用程式的詳細資訊：

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

[DescribeApplicationSnapshot](#) 動作的下列請求程式碼範例會傳回應用程式狀態快照的詳細資訊：

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

DiscoverInputSchema

[DiscoverInputSchema](#) 動作的下列請求程式碼範例會從串流來源產生結構描述：

```
{
  "InputProcessingConfiguration": {
    "InputLambdaProcessor": {
      "ResourceARN": "arn:aws:lambda:us-east-1:012345678901:function:MyLambdaFunction"
    }
  },
  "InputStartingPositionConfiguration": {
    "InputStartingPosition": "NOW"
  },
  "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
  "S3Configuration": {
    "BucketARN": "string",

```

```
    "FileKey": "string"
  },
  "ServiceExecutionRole": "string"
}
```

[DiscoverInputSchema](#) 動作的下列請求程式碼範例會從參考來源產生結構描述：

```
{
  "S3Configuration": {
    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
    "FileKey": "TickerReference.csv"
  },
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

[ListApplications](#) 動作的下列範例請求程式碼會傳回您帳戶中 Managed Service for Apache Flink 應用程式的清單：

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

ListApplicationSnapshots

[ListApplicationSnapshots](#) 動作的下列請求程式碼範例會傳回應用程式狀態的快照清單：

```
{"ApplicationName": "MyApplication",
  "Limit": 50,
  "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

[StartApplication](#) 動作的下列範例請求程式碼會啟動 Managed Service for Apache Flink 應用程式，並從最新的快照 (如有) 載入應用程式狀態：

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

StopApplication

[API_StopApplication](#) 動作的下列範例請求程式碼會停止 Managed Service for Apache Flink 應用程式：

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

[UpdateApplication](#) 動作的下列範例請求程式碼會更新 Managed Service for Apache Flink 應用程式，以變更應用程式程式碼的位置：

```
{"ApplicationName": "MyApplication",
 "CurrentApplicationVersionId": 1,
 "ApplicationConfigurationUpdate": {
   "ApplicationCodeConfigurationUpdate": {
     "CodeContentTypeUpdate": "ZIPFILE",
     "CodeContentUpdate": {
       "S3ContentLocationUpdate": {
         "BucketARNUpdate": "arn:aws:s3:::amzn-s3-demo-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](#)。

發行版本

此內容已移至發行版本。請參閱[支援和取代的 Apache Flink 版本](#)。