aws

使用者指南

Amazon Managed Workflows for Apache Airflow

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Managed Workflows for Apache Airflow: 使用者指南

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

什麼是 Amazon MWAA？ 1
功能 1
Architecture 2
整合 3
支援的版本 3
後續步驟？ 3
快速入門 4
於本教學課程中 4
先決條件 5
步驟一：在本機儲存 CloudFormation 範本 5
步驟二：使用 建立堆疊 AWS CLI 15
步驟三：將 DAG 上傳至 Amazon S3，並在 Apache Airflow UI 中執行 16
步驟四：存取 CloudWatch Logs 中的日誌 16
後續步驟？ 17
開始使用 18
先決條件 18
關於本指南 18
開始之前 19
可用區域 19
建立 儲存貯體 21
開始之前 21
建立儲存貯體 21
後續步驟？ 23
建立 VPC 網路 23
先決條件 24
開始之前 24
建立 Amazon VPC 網路的選項 24
後續步驟？ 36
建立環境 36
開始之前 36
Apache Airflow 版本 37
建立環境 38
後續步驟？ 23
管理存取 43

存取 Amazon MWAA 環境 43
運作方式 44
完整主控台存取 45
完整 API 存取 51
唯讀主控台存取 56
Apache Airflow UI 存取 56
Apache Airflow Rest API 存取 57
Apache Airflow CLI 存取 58
建立 JSON 政策 59
範例使用案例 59
後續步驟？ 61
服務連結角色 61
Amazon MWAA 的服務連結角色許可 62
為 Amazon MWAA 建立服務連結角色 65
編輯 Amazon MWAA 的服務連結角色 65
刪除 Amazon MWAA 的服務連結角色 65
Amazon MWAA 服務連結角色支援的區域 66
政策更新 66
執行角色 66
執行角色概觀 67
Create a new role (建立新角色) 69
存取和更新執行角色政策 69
使用帳戶層級公有存取區塊授予 Amazon S3 儲存貯體的存取權 71
使用 Apache Airflow 連線 71
範例政策 71
後續步驟？ 77
預防跨服務混淆代理人 78
Apache Airflow 存取模式 79
Apache Airflow 存取模式 79
存取模式概觀 80
私有和公有存取模式的設定 81
存取 Apache Airflow Web 伺服器的 VPC 端點 （私有網路存取） 82
存取 Apache Airflow 83
先決條件 83
存取 83
AWS CLI 83

開啟 Apache Airflow UI 84
登入 Apache Airflow 84
建立 Web 伺服器存取字符 84
先決條件 85
使用 AWS CLI 85
使用 bash 指令碼 85
使用 Python 指令碼 86
後續步驟？ 87
設定自訂網域 87
設定自訂網域 88
設定網路基礎設施 88
Apache Airflow CLI 字符 93
先決條件 94
使用 AWS CLI 94
使用 curl 指令碼 94
使用 bash 指令碼 96
使用 Python 指令碼 98
後續步驟？ 101
使用 Apache Airflow REST API 101
授予 Apache Airflow REST API 的存取權： airflow:InvokeRestApi 103
呼叫 Apache Airflow REST API 104
建立 Web 伺服器工作階段字符並呼叫 Apache Airflow REST API 105
Apache Airflow CLI 命令參考 111
先決條件 111
有何變更？ 112
支援的 CLI 命令 112
範本程式碼 119
管理連線 122
概要 122
Apache Airflow 套件 122
限制條件檔案 123
特定版本提供者套件 123
連線類型 131
連線 URI 字串範例 132
連線範本範例 132
使用 HTTP 連線範本進行 Jdbc 連線的範例 132

設定 Secrets Manager 133
步驟一：提供 Amazon MWAA 存取 Secrets Manager 私密金鑰的許可 134
步驟二：建立 Secrets Manager 後端做為 Apache Airflow 組態選項 135
步驟三：產生 Apache Airflow AWS 連線 URI 字串 136
步驟四：在 Secrets Manager 中新增變數 138
步驟五：在 Secrets Manager 中新增連線 140
範本程式碼 141
Resources 141
後續步驟？ 141
管理環境 142
設定環境類別 142
環境功能 142
Apache Airflow 排程器 145
設定工作者自動擴展 145
工作者擴展的運作方式 146
使用 Amazon MWAA 主控台 146
高效能使用案例範例 147
對卡在執行中狀態的任務進行故障診斷 148
後續步驟？ 148
設定 Web 伺服器自動擴展 148
Web 伺服器擴展的運作方式 149
使用 Amazon MWAA 主控台 149
使用組態選項 149
先決條件 150
運作方式 151
使用組態選項載入外掛程式 151
組態選項概觀 151
組態參考 152
範例和範例程式碼 156
後續步驟？ 157
更新環境 157
開始之前 157
工作者替換策略 158
更新環境資源 158
更新環境 159
變更版本 162

升級或降級您的工作流程資源 162
指定新版本 163
使用啟動指令碼 164
設定啟動指令碼 165
安裝 Linux 執行時間 168
設定環境變數 169
使用 DAGs 173
Amazon S3 儲存貯體概觀 173
新增或更新 DAGs 174
先決條件 174
運作方式 174
有哪些變更？ 175
使用 Amazon MWAA CLI 公用程式測試 DAGs 175
將 DAG 程式碼上傳至 Amazon S3 176
指定 DAGs路徑 177
存取 Apache Airflow UI 上的變更 177
後續步驟？ 177
安裝自訂外掛程式 178
先決條件 178
運作方式 179
何時使用外掛程式 179
自訂外掛程式概觀 180
自訂外掛程式的範例 180
建立 plugins.zip 檔案 190
plugins.zip 上傳至 Amazon S3 191
在您的環境上安裝自訂外掛程式 192
plugins.zip 的範例使用案例 193
後續步驟？ 193
安裝 Python 相依性 193
先決條件 194
運作方式 194
Python 相依性概觀 195
建立 requirements.txt 檔案 195
requirements.txt 上傳至 Amazon S3 198
在您的環境上安裝 Python 相依性 199
存取 的日誌 requirements.txt 200

後續步驟？ 201
刪除 Amazon S3 上的檔案 201
先決條件 202
版本控制概觀 202
運作方式 202
在 Amazon S3 上刪除 DAG 202
移除 "current" plugins.zip 或 requirements.txt 203
刪除 "non-current" plugins.zip 或 requirements.txt 203
刪除具有生命週期的檔案 204
生命週期政策範例 204
後續步驟？ 204
聯網 205
關於聯網 205
條款 205
支援的內容 206
VPC 基礎設施概觀 206
Amazon VPC 和 Apache Airflow 存取模式的範例使用案例 209
VPC 中的安全性 211
條款 211
安全性概觀 211
網路存取控制清單 (ACL) 212
VPC security groups (VPC 安全群組) 212
VPC 端點政策 （僅限私有路由） 214
管理對 VPC 端點的存取 215
定價 216
VPC 端點概觀 216
使用其他服務的許可 AWS 217
存取 VPC 端點 217
存取 Apache Airflow Webserver 的 VPC 端點 （私有網路存取） 219
私有 Amazon VPC VPCs 服務端點 220
定價 221
私有網路和私有路由 221
（必要） VPC 端點 221
連接所需的 VPC 端點 222
（選用） 為您的 Amazon S3 VPC 介面端點啟用私有 IP 地址 225
管理您自己的 Amazon VPC 端點 226

在共用的 Amazon VPC 中建立環境 226
教學 236
教學課程： AWS Client VPN 236
私有網路 237
使用案例 237
開始之前 237
目標 237
（選用）步驟一：識別您的 VPC、CIDR 規則和 VPC 安全性 238
步驟二：建立伺服器和用戶端憑證 239
步驟三：在本機儲存 CloudFormation 範本 240
步驟四：建立 Client VPN CloudFormation 堆疊 241
步驟五：將子網路與 Client VPN 建立關聯 242
步驟六：將授權輸入規則新增至 Client VPN 242
步驟七：下載 Client VPN 端點組態檔案 243
步驟八：連線至 AWS Client VPN 244
後續步驟？ 245
教學課程：Linux 堡壘主機 245
私有網路 246
使用案例 246
開始之前 246
目標 246
步驟一：建立堡壘執行個體 247
步驟二：建立 SSH 通道 248
步驟三：將堡壘安全群組設定為傳入規則 249
步驟四：複製 Apache Airflow URL 250
步驟五：設定代理設定 250
步驟六：開啟 Apache Airflow UI 252
後續步驟？ 253
教學課程：將使用者限制為 DAGs的子集 253
先決條件 253
步驟一：使用預設 Public Apache Airflow 角色將 Amazon MWAA Webserver 存取權提供給 IAM 主體。 254
步驟二：建立新的 Apache Airflow 自訂角色 255
步驟三：將您建立的角色指派給 Amazon MWAA 使用者 255
後續步驟 256
相關資源 257

教學課程：自動化管理您自己的環境端點 ... 257
先決條件 ... 257
建立 Amazon VPC ... 258
建立 Lambda 函式 ... 258
建立 EventBridge 規則 ... 259
建立 環境 ... 259
程式碼範例 ... 261
匯入變數 DAG ... 262
版本 ... 262
先決條件 ... 262
權限 ... 262
相依性 ... 262
範例程式碼 ... 262
後續步驟？ ... 264
使用 SSHOperator ... 264
版本 ... 265
先決條件 ... 265
權限 ... 265
要求 ... 265
將您的私密金鑰複製到 Amazon S3 ... 266
建立新的 Apache Airflow 連線 ... 266
範例程式碼 ... 267
Secrets Manager 中的 Apache Airflow Snowflake 連線 ... 268
版本 ... 269
先決條件 ... 269
權限 ... 269
要求 ... 269
範例程式碼 ... 269
後續步驟？ ... 270
使用 DAG 撰寫自訂指標 ... 271
版本 ... 271
先決條件 ... 271
權限 ... 271
相依性 ... 271
程式碼範例 ... 271
Aurora PostgreSQL 資料庫清除 ... 275

版本 275
先決條件 275
相依性 275
範例程式碼 276
將環境中繼資料匯出至 Amazon S3 279
版本 279
先決條件 279
許可 280
要求 280
範例程式碼 280
在 Secrets Manager 中使用 Apache Airflow 變數 283
版本 283
先決條件 283
權限 283
要求 283
範例程式碼 284
後續步驟？ 285
在 Secrets Manager 中使用 Apache Airflow 連線 285
版本 285
先決條件 285
權限 286
要求 283
範例程式碼 286
後續步驟？ 287
搭配 Oracle 的自訂外掛程式 288
版本 288
先決條件 288
權限 288
要求 289
範例程式碼 289
建立自訂外掛程式 290
Airflow 組態選項 293
後續步驟？ 293
變更 DAG 的時區 293
版本 294
先決條件 294

許可 294
建立外掛程式以變更 Airflow 日誌中的時區 294
建立 plugins.zip 295
範例程式碼 295
後續步驟？ 296
在執行時間重新整理 AWS CodeArtifact 權杖 296
版本 297
先決條件 297
權限 297
範例程式碼 298
後續步驟？ 299
使用 Apache Hive 和 Hadoop 的自訂外掛程式 299
版本 300
先決條件 300
權限 300
要求 283
下載相依性 300
自訂外掛程式 301
Plugins.zip 302
範例程式碼 302
Airflow 組態選項 303
後續步驟？ 303
自訂外掛程式以修補 PythonVirtualenvOperator 303
版本 304
先決條件 304
權限 304
要求 304
自訂外掛程式範例程式碼 304
Plugins.zip 305
範例程式碼 306
Airflow 組態選項 307
後續步驟？ 307
使用 Lambda 叫用 DAGs 307
版本 308
先決條件 308
許可 308

相依性 309
程式碼範例 309
在不同的環境中叫用 DAGs 310
版本 310
先決條件 310
許可 311
相依性 311
程式碼範例 311
Amazon RDS 伺服器 313
版本 313
先決條件 314
相依性 275
Apache Airflow v2 連線 314
範例程式碼 315
後續步驟？ 317
Amazon EKS (eksctl) 317
版本 318
先決條件 318
建立 Amazon EC2 的公有金鑰 318
建立叢集 319
建立mwaa命名空間 319
建立 mwaa 命名空間的角色 320
建立並連接 Amazon EKS 叢集的 IAM 角色 321
建立 requirements.txt 檔案 324
建立 Amazon EKS 的身分映射 324
建立 kubeconfig 325
建立 DAG 325
將 DAG 和 kube_config.yaml 新增至 Amazon S3 儲存貯體 326
啟用並觸發範例 326
使用 ECSOperator 327
版本 327
先決條件 327
許可 327
建立 Amazon ECS 叢集 329
範例程式碼 333
搭配 Amazon MWAA 使用 dbt 337

版本 337
先決條件 337
相依性 337
將 dbt 專案上傳至 Amazon S3 339
使用 DAG 驗證 dbt 相依性安裝 339
使用 DAG 執行 dbt 專案 340
AWS 部落格和教學課程 341
最佳實務 342
Apache Airflow 的效能調校 342
新增 Apache Airflow 組態選項 342
Apache Airflow 排程器 343
DAG 資料夾 345
DAG 檔案 347
任務 349
管理 Python 相依性 352
使用 Amazon MWAA CLI 公用程式測試 DAGs 353
使用 PyPi.org 要求檔案格式安裝 Python 相依性 353
在 Amazon MWAA 主控台上啟用日誌 359
在 CloudWatch Logs 主控台上存取日誌 359
在 Apache Airflow UI 中存取錯誤 360
範例requirements.txt案例 360
監控和指標 362
概觀 362
Amazon CloudWatch 概觀 363
AWS CloudTrail 概觀 363
存取稽核日誌 363
在 CloudTrail 中建立追蹤 363
使用 CloudTrail 事件歷史記錄存取事件 364
的範例追蹤 CreateEnvironment 364
後續步驟？ 365
存取 Airflow 日誌 366
定價 366
開始之前 366
日誌類型 366
啟用 Apache Airflow 日誌 367
存取 Apache Airflow 日誌 368

排程器日誌範例 368
後續步驟？ 369
監控儀表板和警示 369
指標 369
警示狀態概觀 370
自訂儀表板和警示範例 370
刪除指標和儀表板 374
後續步驟？ 374
Apache Airflow 環境指標 374
條款 375
維度 375
在 CloudWatch 主控台中存取指標 376
CloudWatch 中可用的 Apache Airflow 指標 377
選擇報告哪些指標 393
後續步驟？ 394
容器、佇列和資料庫指標 394
條款 395
維度 395
存取 指標 396
指標清單 396
安全 400
資料保護 400
加密 401
使用客戶受管金鑰 403
AWS Identity and Access Management 406
目標對象 407
使用身分來驗證 407
使用政策管理存取權 408
允許使用者存取自己的許可 409
對 Amazon Managed Workflows for Apache Airflow 身分和存取進行故障診斷 410
Amazon MWAA 如何與 IAM 搭配使用 411
合規驗證 416
恢復能力 416
基礎設施安全性 416
組態與漏洞分析 417
最佳實務 417

Apache Airflow 中的安全最佳實務 417
版本 419
關於 Amazon MWAA 版本 419
最新版本 419
Apache Airflow 版本 419
Apache Airflow 元件 421
排程器 421
工作程序 421
升級 Apache Airflow 版本 421
降級 Apache Airflow 版本 421
Apache Airflow 已棄用版本 422
Apache Airflow 版本支援和常見問答集 422
常見問答集 422
端點和配額 424
服務端點 424
Service Quotas 424
增加配額 425
FAQs 426
支援的版本 427
Apache Airflow 支援 427
Python 版本 427
使用案例 428
我可以將 Amazon MWAA 與 Amazon SageMaker Unified Studio 搭配使用嗎？ 428
何時可以使用 AWS Step Functions 與 Amazon MWAA？ 429
環境規格 429
每個環境可以使用多少任務儲存體？ 429
預設作業系統 429
自訂映像 429
HIPAA 合規 429
Amazon MWAA 是否支援 Spot 執行個體？ 429
自訂網域 430
SSH 存取 430
自我參考規則 430
自訂指標 430
存放資料 431
工作者配額 431

共享 Amazon VPC 431
共享 Amazon VPC 431
指標 431
工作者指標 431
自訂指標 432
DAGs、運算子、連線和其他問題 432
PythonVirtualenvOperator 432
Amazon MWAA 需要多長時間才能辨識新的 DAG 檔案？ 432
Apache Airflow 為什麼不收取我的 DAG 檔案？ 432
移除 plugins.zip 或 requirements.txt 432
移除 plugins.zip 或 requirements.txt 433
我可以使用 AWS 資料庫遷移服務 (DMS) 運算子嗎？ 433
當我使用 AWS 登入資料存取 Airflow REST API 時，是否可以將限流限制提高到每秒超過 10 筆交易 (TPS)？ 433
Airflow 任務執行 API 伺服器在 Amazon MWAA 中的何處執行？ 433
疑難排解 434
Apache Airflow v2 和 v3 435
連線 436
Web 伺服器 438
任務 439
CLI 441
運算子 442
Amazon MWAA 建立/更新 443
更新 requirements.txt 444
外掛程式 445
建立儲存貯體 445
建立 環境 446
更新環境 448
存取環境 448
CloudWatch Logs 和 CloudTrail 449
日誌 449
Amazon MWAA 使用者指南歷史記錄 454
..... diii

什麼是 Amazon Managed Workflows for Apache Airflow？

使用 Apache Airflow 的受管服務 Amazon Managed Workflows for Apache Airflow，在雲端大規模設定和執行資料管道。Apache Airflow 是一種開放原始碼工具，用於建立、排程和監控工作流程。

透過 Amazon MWAA，您可以使用 Apache Airflow 和 Python 建立工作流程，而無需管理基礎設施以實現可擴展性、可用性和安全性。Amazon MWAA 會自動擴展以滿足您的工作流程需求。它與 AWS 安全服務整合，以提供快速、安全的資料存取。

內容

- 功能
- Architecture
- 整合
- 支援的版本
- 後續步驟？

功能

檢閱下列功能，了解 Amazon MWAA 如何簡化管理 Apache Airflow 工作流程。

- 自動氣流設定 – 在您建立 Amazon MWAA 環境時，透過選擇 Apache Airflow 版本快速設定 Apache Airflow。Amazon MWAA 會使用網際網路上可用的相同 Apache Airflow 使用者介面和開放原始碼，為您設定 Apache Airflow。
- 自動擴展 – 透過設定下限和上限，自動擴展 Apache Airflow 工作者 （執行任務的運算資源）。Amazon MWAA 會監控您環境中的工作者，並使用其自動調整規模元件來新增工作者以滿足需求，最高可達您定義的數目上限。
- 內建身分驗證 – 透過在 AWS Identity and Access Management (IAM) 中定義存取控制政策，為您的 Apache Airflow Web 伺服器啟用角色型身分驗證和授權。Apache Airflow 工作者會擔任這些政策，以安全存取 AWS 服務。
- 內建安全性 – Apache Airflow 工作者和排程器在 Amazon MWAA 的 Amazon VPC 中執行。資料也會使用 自動加密 AWS Key Management Service，因此您的環境預設是安全的。
- 公有或私有存取模式 – 使用私有或公有存取模式存取您的 Apache Airflow Web 伺服器。公有網路存取模式會針對可透過網際網路存取的 Apache Airflow Web 伺服器使用 VPC 端點。私有網路存取模式會針對可在 VPC 中存取的 Apache Airflow Web 伺服器使用 VPC 端點。在這兩種情

況下，Apache Airflow 使用者的存取權都由您在 AWS Identity and Access Management (IAM) 和 AWS SSO 中定義的存取控制政策控制。

- 簡化的升級和修補程式 – Amazon MWAA 會定期提供 Apache Airflow 的新版本。Amazon MWAA 團隊將更新和修補這些版本的映像。
- 工作流程監控 – 存取 Amazon CloudWatch 中的 Apache Airflow 日誌和 Apache Airflow 指標，以識別 Apache Airflow 任務延遲或工作流程錯誤，而不需要額外的第三方工具。Amazon MWAA 會自動將環境指標以及啟用時的 Apache Airflow 日誌傳送至 CloudWatch。
- AWS 整合 – Amazon MWAA 支援與 Amazon Athena AWS Batch、Amazon CloudWatch、Amazon DynamoDB AWS DataSync、Amazon EMR AWS Fargate、Amazon EKS、Amazon Data Firehose AWS Glue、 AWS Lambda Amazon Redshift、Amazon SQS、Amazon SNS、Amazon SageMaker AI 和 Amazon S3 的開放原始碼整合，以及數百個內建和社群建立的運算子和感應器。
- 工作者機群 – Amazon MWAA 支援使用容器隨需擴展工作者機群，並使用 Amazon ECS AWS Fargate on 減少排程器中斷。支援在 Amazon ECS 容器上叫用任務的運算子，以及在 Kubernetes 叢集上建立和執行 Pod 的 Kubernetes 運算子。

Architecture

外部方塊中包含的所有元件 （下圖中） 都會顯示為您帳戶中的單一 Amazon MWAA 環境。Apache Airflow 排程器和工作者是連線到您環境 Amazon VPC 中私有子網路的 AWS Fargate 容器。每個環境都有自己由 管理的 Apache Airflow 中繼資料庫 AWS ，可透過私有安全的 VPC 端點存取排程器和工作者 Fargate 容器。

Amazon CloudWatch、Amazon S3、Amazon SQS 和 AWS KMS 與 Amazon MWAA 分開，需要從 Fargate 容器中的 Apache Airflow 排程器和工作者存取。多個 Apache Airflow 排程器僅適用於 Apache Airflow v2 和更新版本。請參閱《Apache Airflow 參考指南》中的概念，進一步了解 Apache Airflow 任務生命週期。

您可以選取公有網路 Apache Airflow 存取模式，或選取私有網路 Apache Airflow 存取模式，透過網際網路存取 Apache Airflow Web 伺服器。在這兩種情況下，您 Apache Airflow 使用者的存取權是由您在 AWS Identity and Access Management (IAM) 中定義的存取控制政策所控制。

> **Note**
>
> 從 Apache Airflow v3 開始，Amazon MWAA Web 伺服器也會託管 Apache Airflow 的執行 API 伺服器。

整合

主動和不斷成長的 Apache Airflow 開放原始碼社群為 Apache Airflow 提供運算子 （簡化 服務連線的外掛程式），以便與 AWS 服務整合。這包括 Amazon S3、Amazon Redshift AWS Batch、Amazon EMR 和 Amazon SageMaker AI 等服務，以及其他雲端平台上的服務。

搭配 Amazon MWAA 使用 Apache Airflow 可完全支援與服務 AWS 和熱門第三方工具整合，例如 Apache Hadoop、Presto、Hive 和 Spark，以執行資料處理任務。Amazon MWAA 致力於維持與 Apache Airflow API 的相容性，Amazon MWAA 打算為 AWS 服務提供可靠的整合，並將其提供給社群，並參與社群功能開發。

如需範例程式碼，請參閱 Amazon Managed Workflows for Apache Airflow 的程式碼範例。

支援的版本

Amazon MWAA 支援多個版本的 Apache Airflow。如需我們支援的 Apache Airflow 版本和每個版本隨附的 Apache Airflow 元件的詳細資訊，請參閱 Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本。

後續步驟？

- 開始使用為 Airflow DAGs 和支援檔案建立 Amazon S3 儲存貯體的單一 CloudFormation 範本、具有公有路由的 Amazon VPC，以及 中的 Amazon MWAA 環境Amazon Managed Workflows for Apache Airflow 的快速入門教學課程。
- 為您的 Airflow DAGs 和支援檔案建立 Amazon S3 儲存貯體、從三個 Amazon VPC 聯網選項中選擇其中一個，以及在 中建立 Amazon MWAA 環境，以遞增方式開始開始使用 Amazon Managed Workflows for Apache Airflow。

Amazon Managed Workflows for Apache Airflow 的快速入門教學課程

此快速入門教學課程使用 AWS CloudFormation 範本來建立 Amazon VPC 基礎設施、具有dags資料夾的 Amazon S3 儲存貯體，以及 Amazon Managed Workflows for Apache Airflow 環境。

主題

- 於本教學課程中
- 先決條件
- 步驟一：在本機儲存 CloudFormation 範本
- 步驟二：使用 建立堆疊 AWS CLI
- 步驟三：將 DAG 上傳至 Amazon S3，並在 Apache Airflow UI 中執行
- 步驟四：存取 CloudWatch Logs 中的日誌
- 後續步驟？

於本教學課程中

使用此教學課程將 DAG 上傳至 Amazon S3、在 Apache Airflow 中執行 DAG，並使用三 AWS Command Line Interface (AWS CLI) 命令存取 CloudWatch 中的日誌。最後，您將學習如何為 Apache Airflow 開發團隊建立 IAM 政策。

> **Note**
>
> 此頁面上的 CloudFormation 範本會為 中可用的最新版本 Apache Airflow 建立 Amazon Managed Workflows for Apache Airflow 環境 CloudFormation。最新的可用版本是 Apache Airflow v3.0.6。

CloudFormation 範本會建立下列項目：

- VPC 基礎設施。範本使用 透過網際網路的公有路由。它會公有網路存取模式針對 中的 Apache Airflow Web 伺服器使用 `WebserverAccessMode: PUBLIC_ONLY`。
- Amazon S3 儲存貯體。範本會使用dags資料夾建立 Amazon S3 儲存貯體。其設定為封鎖所有公有存取，並啟用儲存貯體版本控制，如 中所定義為 Amazon MWAA 建立 Amazon S3 儲存貯體。

- Amazon MWAA 環境。範本會建立與 Amazon S3 儲存貯體上 dags 資料夾相關聯的 Amazon MWAA 環境、具有 Amazon MWAA 所使用 AWS 服務許可的執行角色，以及使用 AWS擁有的金鑰進行加密的預設值，如 中所定義建立 Amazon MWAA 環境。
- CloudWatch Logs。範本會在 CloudWatch 中於 INFO 層級開啟 Apache Airflow 日誌，並為 Airflow 排程器日誌群組、Airflow Web 伺服器日誌群組、Airflow 工作者日誌群組、Airflow DAG 處理日誌群組和 Airflow 任務日誌群組啟動，如 中所定義在 Amazon CloudWatch 中存取 Airflow 日誌。

在本教學課程中，您將完成下列任務：

- 上傳並執行 DAG。將最新 Amazon MWAA 支援的 Apache Airflow 版本的 Apache Airflow 教學 DAG 上傳至 Amazon S3，然後在 Apache Airflow UI 中執行，如 中所定義新增或更新 DAGs。
- 存取日誌。存取 CloudWatch Logs 中的 Airflow Web 伺服器日誌群組，如 中所定義在 Amazon CloudWatch 中存取 Airflow 日誌。
- 建立存取控制政策。在 IAM 中為您的 Apache Airflow 開發團隊建立存取控制政策，如 中所定義存取 Amazon MWAA 環境。

> **Note**
>
> 在託管 Amazon MWAA 環境的 VPC 中，`true`針對所有連接的子網路`assignIpv6AddressOnCreation`將 設定為 。此設定可確保自動指派網際網路通訊協定第 6 版 (IPv6) 地址給這些子網路內的資源。

先決條件

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

步驟一：在本機儲存 CloudFormation 範本

- 複製下列範本的內容，並在本機儲存為 `mwaa-public-network.yml`。您也可以下載 範本。

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:

  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: MWAAEnvironment

  VpcCIDR:
    Description: The IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.21.0/24
  MaxWorkerNodes:
    Description: The maximum number of workers that can run in the environment
    Type: Number
    Default: 2
  DagProcessingLogs:
    Description: Log level for DagProcessing
    Type: String
```

```
    Default: INFO
  SchedulerLogsLevel:
    Description: Log level for SchedulerLogs
    Type: String
    Default: INFO
  TaskLogsLevel:
    Description: Log level for TaskLogs
    Type: String
    Default: INFO
  WorkerLogsLevel:
    Description: Log level for WorkerLogs
    Type: String
    Default: INFO
  WebserverLogsLevel:
    Description: Log level for WebserverLogs
    Type: String
    Default: INFO

Resources:

 ##################################################################################################
  # CREATE VPC

 ##################################################################################################

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: MWAAEnvironment

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: MWAAEnvironment

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
```

```
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
```

```
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
```

```
  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1

  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
```

```
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  EnvironmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true


 ################################################################################################
  # CREATE MWAA

 ################################################################################################

  MwaaEnvironment:
    Type: AWS::MWAA::Environment
```

```
    DependsOn: MwaaExecutionPolicy
    Properties:
      Name: !Sub "${AWS::StackName}-MwaaEnvironment"
      SourceBucketArn: !GetAtt EnvironmentBucket.Arn
      ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
      DagS3Path: dags/
      NetworkConfiguration:
        SecurityGroupIds:
          - !GetAtt SecurityGroup.GroupId
        SubnetIds:
          - !Ref PrivateSubnet1
          - !Ref PrivateSubnet2
      WebserverAccessMode: PUBLIC_ONLY
      MaxWorkers: !Ref MaxWorkerNodes
      LoggingConfiguration:
        DagProcessingLogs:
          LogLevel: !Ref DagProcessingLogs
          Enabled: true
        SchedulerLogs:
          LogLevel: !Ref SchedulerLogsLevel
          Enabled: true
        TaskLogs:
          LogLevel: !Ref TaskLogsLevel
          Enabled: true
        WorkerLogs:
          LogLevel: !Ref WorkerLogsLevel
          Enabled: true
        WebserverLogs:
          LogLevel: !Ref WebserverLogsLevel
          Enabled: true

  MwaaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17&TCX5-2025-waiver;
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - airflow-env.amazonaws.com
                - airflow.amazonaws.com
            Action:
             - "sts:AssumeRole"
```

```
      Path: "/service-role/"

  MwaaExecutionPolicy:
    DependsOn: EnvironmentBucket
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
        - !Ref MwaaExecutionRole
      PolicyDocument:
        Version: 2012-10-17&TCX5-2025-waiver;
        Statement:
          - Effect: Allow
            Action: airflow:PublishMetrics
            Resource:
              - !Sub "arn:aws:airflow:${AWS::Region}:${AWS::AccountId}:environment/
${EnvironmentName}"
          - Effect: Deny
            Action: s3:ListAllMyBuckets
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"

          - Effect: Allow
            Action:
              - "s3:GetObject*"
              - "s3:GetBucket*"
              - "s3:List*"
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"
          - Effect: Allow
            Action:
              - logs:DescribeLogGroups
            Resource: "*"

          - Effect: Allow
            Action:
              - logs:CreateLogStream
              - logs:CreateLogGroup
              - logs:PutLogEvents
              - logs:GetLogEvents
              - logs:GetLogRecord
              - logs:GetLogGroupFields
              - logs:GetQueryResults
```

```
              - logs:DescribeLogGroups
            Resource:
              - !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:airflow-${AWS::StackName}*"
          - Effect: Allow
            Action: cloudwatch:PutMetricData
            Resource: "*"
          - Effect: Allow
            Action:
              - sqs:ChangeMessageVisibility
              - sqs:DeleteMessage
              - sqs:GetQueueAttributes
              - sqs:GetQueueUrl
              - sqs:ReceiveMessage
              - sqs:SendMessage
            Resource:
              - !Sub "arn:aws:sqs:${AWS::Region}:*:airflow-celery-*"
          - Effect: Allow
            Action:
              - kms:Decrypt
              - kms:DescribeKey
              - "kms:GenerateDataKey*"
              - kms:Encrypt
            NotResource: !Sub "arn:aws:kms:*:${AWS::AccountId}:key/*"
            Condition:
              StringLike:
                "kms:ViaService":
                  - !Sub "sqs.${AWS::Region}.amazonaws.com"
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1
```

```
  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress

  MwaaApacheAirflowUI:
    Description: MWAA Environment
    Value: !Sub  "https://${MwaaEnvironment.WebserverUrl}"
```

步驟二：使用 建立堆疊 AWS CLI

1. 在命令提示字元中，導覽至mwaa-public-network.yml存放 的目錄。例如：

```
cd mwaaproject
```

2. 使用 aws cloudformation create-stack命令來使用 建立堆疊 AWS CLI。

```
aws cloudformation create-stack --stack-name mwaa-environment-public-network --
template-body file://mwaa-public-network.yml --capabilities CAPABILITY_IAM
```

> **Note**
>
> 建立 Amazon VPC 基礎設施、Amazon S3 儲存貯體和 Amazon MWAA 環境需要 30 分鐘以上。

步驟三：將 DAG 上傳至 Amazon S3，並在 Apache Airflow UI 中執行

1. 複製最新支援的 Apache Airflow 版本的 tutorial.py 檔案內容，並在本機儲存為 tutorial.py。
2. 在您的命令提示字元中，導覽至tutorial.py存放 的目錄。例如：

```
cd mwaaproject
```

3. 使用下列命令列出所有 Amazon S3 儲存貯體。

```
aws s3 ls
```

4. 使用下列命令列出您環境的 Amazon S3 儲存貯體中的檔案和資料夾。

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

5. 使用下列指令碼將tutorial.py檔案上傳至您的dags資料夾。以 *amzn-s3-demo-bucket* 取代範例值。

```
aws s3 cp tutorial.py s3://amzn-s3-demo-bucket/dags/
```

6. 在 Amazon MWAA 主控台上開啟環境頁面。
7. 選擇環境。
8. 選擇開啟氣流使用者介面。
9. 在 Apache Airflow UI 上，從可用的 DAGs清單中，選擇教學課程 DAG。
10. 在 DAG 詳細資訊頁面上，選擇 DAG 名稱旁的暫停/取消暫停 DAG 切換，以取消暫停 DAG。
11. 選擇觸發 DAG。

步驟四：存取 CloudWatch Logs 中的日誌

您可以在 CloudWatch 主控台中存取 CloudFormation 由堆疊開啟的所有 Apache Airflow 日誌的 Apache Airflow 日誌。下一節說明如何存取 Airflow Web 伺服器日誌群組的日誌。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。

3. 在監控窗格中選擇 Airflow Web 伺服器日誌群組。
4. 在webserver_console_ip日誌串流中選擇日誌。

後續步驟？

- 進一步了解如何上傳 DAGs、在 中指定 Python 相依性，requirements.txt以及在 plugins.zip 中指定自訂外掛程式在 Amazon MWAA 上使用 DAG。
- 進一步了解建議您在 中調整環境效能的最佳實務Amazon MWAA 上 Apache Airflow 的效能調校。
- 在 中為您的環境建立監控儀表板在 Amazon MWAA 上監控儀表板和警示。
- 在 中執行一些 DAG 程式碼範例Amazon Managed Workflows for Apache Airflow 的程式碼範例。

開始使用 Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow 會使用 Amazon S3 儲存貯體中的 Amazon VPC、DAG 檔案和支援檔案來建立環境。本章說明開始使用 Amazon MWAA 所需的先決條件 AWS 和資源。

主題

- 先決條件
- 關於本指南
- 開始之前
- 可用區域
- 為 Amazon MWAA 建立 Amazon S3 儲存貯體
- 建立 VPC 網路
- 建立 Amazon MWAA 環境
- 後續步驟？

先決條件

若要建立 Amazon MWAA 環境，請確定您擁有建立所需 AWS 資源的許可。

- AWS 帳戶 – AWS 帳戶 具有使用 Amazon MWAA 和您環境所用 AWS 服務和資源之許可的 。

關於本指南

本指南涵蓋您將建立的 AWS 基礎設施和資源。

- Amazon VPC – Amazon MWAA 環境所需的 Amazon VPC 網路元件。您可以設定符合這些需求（進階）的現有 VPC關於 Amazon MWAA 上的聯網，或建立 VPC 和網路元件，如 中所定義the section called “建立 VPC 網路”。
- Amazon S3 儲存貯體 – 用來存放 DAGs和相關聯檔案的 Amazon S3 儲存貯體，例如 `plugins.zip`和 `requirements.txt`。您的 Amazon S3 儲存貯體必須設定為封鎖所有公有存取，並啟用儲存貯體版本控制，如 中所定義為 Amazon MWAA 建立 Amazon S3 儲存貯體。

- Amazon MWAA 環境 – Amazon MWAA 環境，設定 Amazon S3 儲存貯體的位置、DAG 程式碼和任何自訂外掛程式或 Python 相依性的路徑，以及 Amazon VPC 及其安全群組，如 中所定義建立 Amazon MWAA 環境。

開始之前

若要建立 Amazon MWAA 環境，您可以在建立環境之前採取其他步驟來建立和設定其他 AWS 資源。

若要建立環境，您需要下列項目：

- AWS KMS 金鑰 – 您環境中資料加密的 AWS KMS 金鑰。您可以在 Amazon MWAA 主控台上選擇預設選項，以在建立 環境時建立 AWS擁有的金鑰，或指定具有環境設定（進階）AWS 使用之其他服務許可的現有客戶受管金鑰。若要進一步了解，請參閱 使用客戶受管金鑰進行加密。
- 執行角色 – 允許 Amazon MWAA 存取您環境中 AWS 資源的執行角色。您可以在 Amazon MWAA 主控台上選擇預設選項，以在建立環境時建立執行角色。若要進一步了解，請參閱 Amazon MWAA 執行角色。
- VPC 安全群組 – 允許 Amazon MWAA 存取 VPC 網路中其他 AWS 資源的 VPC 安全群組。您可以在 Amazon MWAA 主控台上選擇預設選項，以在建立環境時建立安全群組，或為安全群組提供適當的傳入和傳出規則（進階）。若要進一步了解，請參閱 Amazon MWAA 上 VPC 的安全性。

可用區域

下列提供 Amazon MWAA AWS 區域。若要進一步了解每個區域，例如預設啟用或停用的區域，請參閱 AWS 區域。

Code	Name
us-east-1	美國東部 (維吉尼亞北部)
us-east-2	美國東部 (俄亥俄)
us-west-1	美國西部 (加利佛尼亞北部)
us-west-2	美國西部 (奧勒岡)
af-south-1	非洲 (開普敦)

Code	Name
ap-east-1	亞太地區 (香港)
ap-south-2	亞太地區 (海德拉巴)
ap-southeast-3	亞太地區 (雅加達)
ap-southeast-5	亞太地區 (馬來西亞)
ap-southeast-4	亞太地區 (墨爾本)
ap-south-1	亞太地區 (孟買)
ap-northeast-3	亞太地區 (大阪)
ap-northeast-2	亞太地區 (首爾)
ap-southeast-1	亞太地區 (新加坡)
ap-southeast-2	亞太地區 (雪梨)
ap-northeast-1	亞太地區 (東京)
ca-central-1	加拿大 (中部)
ca-west-1	加拿大西部 (卡加利)
eu-central-1	歐洲 (法蘭克福)
eu-west-1	歐洲 (愛爾蘭)
eu-west-2	歐洲 (倫敦)
eu-south-1	歐洲 (米蘭)
eu-west-3	Europe (Paris)
eu-south-2	歐洲 (西班牙)
eu-north-1	歐洲 (斯德哥爾摩)

Code	Name
eu-central-2	歐洲 (蘇黎世)
il-central-1	以色列 (特拉維夫)
me-south-1	Middle East (Bahrain)
me-central-1	中東 (阿拉伯聯合大公國)
sa-east-1	南美洲 (聖保羅)

為 Amazon MWAA 建立 Amazon S3 儲存貯體

本指南說明建立 Amazon S3 儲存貯體以將 Apache Airflow 導向無環圖 (DAGs)、`plugins.zip`檔案中的自訂外掛程式，以及`requirements.txt`檔案中的 Python 相依性存放的步驟。

內容

- 開始之前
- 建立儲存貯體
- 後續步驟？

開始之前

- 建立儲存貯體後，無法變更 Amazon S3 儲存貯體名稱。若要進一步了解，請參閱《Amazon Simple Storage Service 使用者指南》中的儲存貯體命名規則。
- 用於 Amazon MWAA 環境的 Amazon S3 儲存貯體必須設定為封鎖所有公開存取，並啟用儲存貯體版本控制。
- 用於 Amazon MWAA 環境的 Amazon S3 儲存貯體必須位於與 Amazon MWAA 環境 AWS 區域 相同的 中。若要存取 Amazon MWAA AWS 區域 的 清單，請參閱 中的 Amazon MWAA 端點和配額AWS 一般參考。

建立儲存貯體

本節說明為您的環境建立 Amazon S3 儲存貯體的步驟。

建立儲存貯體

1. 登入 AWS 管理主控台 ，並在 https://console.aws.amazon.com/s3/：// 開啟 Amazon S3 主控台。
2. 選擇建立儲存貯體。
3. 在 Bucket name (儲存貯體名稱) 中，為儲存貯體輸入符合 DNS 規範的名稱。

   儲存貯體名稱必須；

   - 在所有 Amazon S3 中都為唯一。
   - 長度必須介於 3 與 63 個字元之間。
   - 不含大寫字元。
   - 以小寫字母或數字開頭。

   > **Important**
   >
   > 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。儲存貯體名稱可在指向儲存貯體中物件URLs 中使用。

4. AWS 區域 在 區域中選擇 。這必須與 Amazon MWAA 環境 AWS 區域 相同。

   - 我們建議您選擇靠近您的區域，將延遲和成本降至最低，並解決法規要求。

5. 選擇 Block all public access (封鎖所有公用存取)。
6. 在儲存貯體版本控制中選擇啟用。
7. 選用 - 標籤。新增鍵值標籤對，以識別標籤中的 Amazon S3 儲存貯體。例如，`Bucket : Staging`。
8. 選用 - 伺服器端加密。您可以選擇性地在 Amazon S3 儲存貯體上啟用下列其中一個加密選項。

   a. 在伺服器端加密中選擇 Amazon S3 金鑰 (SSE-S3)，以啟用儲存貯體的伺服器端加密。

   b. 選擇AWS Key Management Service 金鑰 (SSE-KMS) 以在 Amazon S3 儲存貯體上使用 AWS KMS 金鑰進行加密：

      i. AWS 受管金鑰 (aws/s3) - 如果您選擇此選項，您可以使用 Amazon MWAA 管理的 AWS 擁有金鑰，或指定客戶受管金鑰來加密 Amazon MWAA 環境。

      ii. 從 AWS KMS 金鑰中選擇或輸入 AWS KMS 金鑰 ARN - 如果您在此步驟中選擇指定客戶受管金鑰，則必須指定 AWS KMS 金鑰 ID 或 ARN。AWS KMS Amazon MWAA 不支援

別名和多區域金鑰。您指定的 AWS KMS 金鑰也必須用於 Amazon MWAA 環境上的加密。

9. 選用 - 進階設定。如果您想要啟用 Amazon S3 物件鎖定：

   a. 選擇進階設定，啟用。

   > **Important**
   >
   > 啟用物件鎖定將永久允許鎖定此儲存貯體中的物件。若要進一步了解，請參閱《Amazon Simple Storage Service 使用者指南》中的使用 Amazon S3 物件鎖定鎖定物件。

   b. 選擇確認。

10. 選擇建立儲存貯體。

後續步驟？

- 了解如何在 中為環境建立所需的 Amazon VPC 網路建立 VPC 網路。
- 了解如何在如何設定 ACL 儲存貯體許可中管理存取許可？
- 了解如何在如何刪除 S3 儲存貯體中刪除儲存貯體？。

建立 VPC 網路

Amazon Managed Workflows for Apache Airflow 需要 Amazon VPC 和特定聯網元件來支援環境。本指南說明為 Amazon Managed Workflows for Apache Airflow 環境建立 Amazon VPC 網路的不同選項。

> **Note**
>
> Apache Airflow 在低延遲的網路環境中效果最佳。如果您使用的是將流量路由到另一個區域或內部部署環境的現有 Amazon VPC，建議您為 Amazon SQS、CloudWatch、Amazon S3 和新增 AWS PrivateLink 端點 AWS KMS。如需 AWS PrivateLink 設定 Amazon MWAA 的詳細資訊，請參閱建立沒有網際網路存取權的 Amazon VPC 網路。

內容

- 先決條件
- 開始之前
- 建立 Amazon VPC 網路的選項
  - 選項一：在 Amazon MWAA 主控台上建立 VPC 網路
  - 選項二：建立具有網際網路存取的 Amazon VPC 網路
  - 選項三：在沒有網際網路存取的情況下建立 Amazon VPC 網路
- 後續步驟？

先決條件

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

開始之前

- 您為環境指定的 VPC 網路無法在環境建立後變更。
- 您可以為 Amazon VPC 和 Apache Airflow Web 伺服器使用私有或公有路由。若要存取選項清單，請參閱 the section called “Amazon VPC 和 Apache Airflow 存取模式的範例使用案例”。

建立 Amazon VPC 網路的選項

下一節說明為 環境建立 Amazon VPC 網路的可用選項。

> **Note**
>
> Amazon MWAA 不支援在美國東部（維吉尼亞北部）區域使用`use1-az3`可用區域 (AZ)。在美國東部（維吉尼亞北部）區域中建立 Amazon MWAA 的 VPC 時，您必須在 CloudFormation (CFN) 範本`AvailabilityZone`中明確指派 。指派的可用區域名稱不得映射至 `use1-az3`。您可以執行下列命令，將 AZ 名稱的詳細映射擷取到其對應的 AZ IDs：
>
> ```
> aws ec2 describe-availability-zones --region us-east-1
> ```

選項一：在 Amazon MWAA 主控台上建立 VPC 網路

下一節說明如何在 Amazon MWAA 主控台上建立 Amazon VPC 網路。此選項使用 透過網際網路的公有路由。它可用於具有私有網路或公有網路存取模式的 Apache Airflow Web 伺服器。

下圖說明您可以在 Amazon MWAA 主控台上找到建立 MWAA VPC 按鈕的位置。

選項二：建立具有網際網路存取的 Amazon VPC 網路

下列 CloudFormation 範本會在您的預設 中建立具有網際網路存取的 Amazon VPC 網路 AWS 區域。此選項使用 透過網際網路的公有路由。此範本可用於具有私有網路或公有網路存取模式的 Apache Airflow Web 伺服器。

1. 複製下列範本的內容，並在本機儲存為 `cfn-vpc-public-private.yaml`。您也可以下載 範本。

```
Description:  This template deploys a VPC, with a pair of public and private
 subnets spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: mwaa-

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the second Availability Zone
```

```
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the first Availability Zone
    Type: String
    Default: 10.192.20.0/24

  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
```

```
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
```

```
  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
```

```
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1

  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
```

```
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2
```

```
  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress
```

2. 在您的命令提示字元中，導覽至cfn-vpc-public-private.yaml存放 的目錄。例如：

```
cd mwaaproject
```

3. 使用 aws cloudformation create-stack命令來使用 建立堆疊 AWS CLI。

```
aws cloudformation create-stack --stack-name mwaa-environment --template-body
 file://cfn-vpc-public-private.yaml
```

> **Note**
>
> 建立 Amazon VPC 基礎設施大約需要 30 分鐘。

選項三：在沒有網際網路存取的情況下建立 Amazon VPC 網路

下列 CloudFormation 範本會在您的預設 中建立沒有網際網路存取權的 Amazon VPC 網路 AWS 區域。

此選項使用 沒有網際網路存取的私有路由。此範本僅適用於具有私有網路存取模式的 Apache Airflow Web 伺服器。它會為環境使用 AWS 的服務建立所需的 VPC 端點。

1. 複製下列範本的內容，並在本機儲存為 cfn-vpc-private.yaml。您也可以下載 範本。

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:
   VpcCIDR:
     Description: The IP range (CIDR notation) for this VPC
     Type: String
     Default: 10.192.0.0/16

   PrivateSubnet1CIDR:
     Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
     Type: String
```

```
    Default: 10.192.10.0/24

  PrivateSubnet2CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
       - Key: Name
         Value: !Ref AWS::StackName

  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName}-route-table"

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName} Private Subnet (AZ1)"

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
```

```
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName} Private Subnet (AZ2)"

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet1

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet2

  S3VpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.s3"
      VpcEndpointType: Gateway
      VpcId: !Ref VPC
      RouteTableIds:
       - !Ref RouteTable

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId: !Ref VPC
      GroupDescription: Security Group for Amazon MWAA Environments to access VPC
endpoints
      GroupName: !Sub "${AWS::StackName}-mwaa-vpc-endpoints"

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  SqsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
```

```
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.sqs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchLogsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.logs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchMonitoringVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.monitoring"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  KmsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.kms"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
```

```
        - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
      SecurityGroupIds:
        - !Ref SecurityGroup


Outputs:
   VPC:
     Description: A reference to the created VPC
     Value: !Ref VPC

   MwaaSecurityGroupId:
     Description: Associates the Security Group to the environment to allow access
 to the VPC endpoints
     Value: !Ref SecurityGroup

   PrivateSubnets:
     Description: A list of the private subnets
     Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

   PrivateSubnet1:
     Description: A reference to the private subnet in the 1st Availability Zone
     Value: !Ref PrivateSubnet1

   PrivateSubnet2:
     Description: A reference to the private subnet in the 2nd Availability Zone
     Value: !Ref PrivateSubnet2
```

2. 在您的命令提示字元中，導覽至cfn-vpc-private.yml存放 的目錄。例如：

```
cd mwaaproject
```

3. 使用 aws cloudformation create-stack命令來使用 建立堆疊 AWS CLI。

```
aws cloudformation create-stack --stack-name mwaa-private-environment --template-
body file://cfn-vpc-private.yml
```

> **Note**
>
> 建立 Amazon VPC 基礎設施大約需要 30 分鐘。

4. 您需要建立機制，才能從電腦存取這些 VPC 端點。若要進一步了解，請參閱 在 Amazon MWAA 上管理對服務特定 Amazon VPC 端點的存取。

> **Note**
>
> 您可以在 Amazon MWAA 安全群組的 CIDR 中進一步限制傳出存取。例如，您可以透過新增自我參考傳出規則、Amazon S3 的字首清單，以及 Amazon VPC 的 CIDR 來限制 本身。

後續步驟？

- 了解如何在 中建立 Amazon MWAA 環境建立 Amazon MWAA 環境。
- 了解如何在 中使用私有路由從您的電腦建立 VPN 通道至 Amazon VPC教學課程：使用 設定私有網路存取 AWS Client VPN。

建立 Amazon MWAA 環境

Amazon Managed Workflows for Apache Airflow 會使用 Apache 提供的相同開放原始碼 Apache Airflow 和使用者介面，在所選版本的環境中設定 Apache Airflow。本指南說明建立 Amazon MWAA 環境的步驟。

內容

- 開始之前
- Apache Airflow 版本
- 建立環境
  - 步驟一：指定詳細資訊
  - 步驟二：設定進階設定
  - 步驟三：檢閱並建立

開始之前

- 建立環境後，無法修改您為環境指定的 VPC 網路。
- 您需要將 Amazon S3 儲存貯體設定為封鎖所有公有存取，並啟用儲存貯體版本控制。

- 您需要 AWS 帳戶 具有使用 Amazon MWAA 許可的 ，以及 AWS Identity and Access Management (IAM) 中建立 IAM 角色的許可。如果您選擇 Apache Airflow Webserver 的私有網路存取模式，這會限制 Amazon VPC 內的 Apache Airflow 存取，則需要 IAM 中的許可才能建立 Amazon VPC 端點。

> **Note**
>
> Amazon MWAA 會在建立期間動態決定網路。如果您使用 IPv6 子網路，Amazon MWAA 會建立與資料庫和 Web 伺服器的 IPv6 私有連結連線。Amazon MWAA 不支援在網路類型之間轉換，也無法將現有環境升級至 IPv6。

Apache Airflow 版本

Amazon Managed Workflows for Apache Airflow 支援下列 Apache Airflow 版本。

> **Note**
>
> - 自 2025 年 12 月 30 日起，Amazon MWAA 將終止對 Apache Airflow 版本 v2.4.3、v2.5.1 和 v2.6.3 的支援。如需詳細資訊，請參閱 Apache Airflow 版本支援和常見問答集。
> - 從 Apache Airflow v2.2.2 開始，Amazon MWAA 支援直接在 Apache Airflow Web 伺服器上安裝 Python 需求、供應商套件和自訂外掛程式。
> - 從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制，Amazon MWAA 會為您指定一個，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。
>
>   如需有關在需求檔案中設定限制的詳細資訊，請參閱安裝 Python 相依性。

Apache Airflow 版本	Apache Airflow 發行日期	Amazon MWAA 可用性日期	Apache Airflow 限制條件	Python 版本
2.11.0 版	2025 年 5 月 20 日	2026 年 1 月 7 日	v2.11.0 限制條件檔案	Python 3.12
3.0.6 版	2025 年 8 月 29 日	2025 年 10 月 1 日	v3.0.6 限制條件檔案	Python 3.12

Apache Airflow 版本	Apache Airflow 發行日期	Amazon MWAA 可用性日期	Apache Airflow 限制條件	Python 版本
v2.10.3	2024 年 11 月 4 日	2024 年 12 月 18 日	v2.10.3 限制條件檔案	Python 3.11
2.10.1 版	2024 年 9 月 5 日	2024 年 9 月 26 日	v2.10.1 限制條件檔案	Python 3.11
v2.9.2	2024 年 6 月 10 日	2024 年 7 月 9 日	v2.9.2 限制條件檔案	Python 3.11
2.8.1 版	2024 年 1 月 19 日	2024 年 2 月 23 日	v2.8.1 限制條件檔案	Python 3.11
2.7.2 版	2023 年 10 月 12 日	2023 年 11 月 6 日	v2.7.2 限制條件檔案	Python 3.11

如需遷移自我管理 Apache Airflow 部署或遷移現有 Amazon MWAA 環境的詳細資訊，包括備份中繼資料資料庫的說明，請參閱 Amazon MWAA 遷移指南。

建立環境

下一節說明建立 Amazon MWAA 環境的步驟。

步驟一：指定詳細資訊

指定環境的詳細資訊

1. 開啟 Amazon MWAA 主控台。
2. 選取您的 AWS 區域。
3. 選擇 Create environment (建立環境)。
4. 在指定詳細資訊頁面的環境詳細資訊下：
    a. 在名稱中為您的環境輸入唯一名稱。
    b. 在 Airflow 版本中選擇 Apache Airflow 版本。

> **Note**
>
> 如果未指定任何值，會預設為最新的 Apache Airflow 版本。最新的可用版本是 Apache Airflow v3.0.6。

5. 在 Amazon S3 中的 DAG 程式碼下，指定下列項目：

   a. S3 儲存貯體。選擇瀏覽 S3 並選取您的 Amazon S3 儲存貯體，或輸入 Amazon S3 URI。

   b. DAGs。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體中的`dags`資料夾，或輸入 Amazon S3 URI。

   c. 外掛程式檔案 - 選用。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體上的`plugins.zip`檔案，或輸入 Amazon S3 URI。

   d. 需求檔案 - 選用。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體上的`requirements.txt`檔案，或輸入 Amazon S3 URI。

   e. 啟動指令碼檔案 - 選用，選擇瀏覽S3並選取 Amazon S3 儲存貯體上的指令碼檔案，或輸入 Amazon S3 URI。

6. 選擇下一步。

步驟二：設定進階設定

設定進階設定

1. 在設定進階設定頁面的網路下：

   - 選擇您的 Amazon VPC。

     此步驟會在 Amazon VPC 中填入兩個私有子網路。

2. 在 Web 伺服器存取下，選取您偏好的 Apache Airflow 存取模式：

   a. 私有網路。這會將 Apache Airflow UI 的存取權限制在 Amazon VPC 中已授予您環境 IAM 政策存取權的使用者。您需要許可才能建立此步驟的 Amazon VPC 端點。

   > **Note**
   >
   > 如果您的 Apache Airflow UI 僅在公司網路內存取，而且您不需要存取公有儲存庫以進行 Web 伺服器需求安裝，請選擇私有網路選項。如果您選擇此存取模式選項，則需

要建立機制來存取 Amazon VPC 中的 Apache Airflow Web 伺服器。如需詳細資訊，請參閱 存取 Apache Airflow Webserver 的 VPC 端點（私有網路存取）。

b. 公有網路。這可讓獲授予環境 IAM 政策存取權的使用者透過網際網路存取 Apache Airflow UI。

3. 在安全群組下，選擇用來保護 Amazon VPC 的安全群組：

a. 根據預設，Amazon MWAA 會在您的 Amazon VPC 中建立安全群組，並在建立新的安全群組中使用特定傳入和傳出規則。

b. 「選用」。取消選取建立新安全群組中的核取方塊，以選取最多 5 個安全群組。

> **Note**
>
> 現有的 Amazon VPC 安全群組必須設定特定的傳入和傳出規則，以允許網路流量。若要進一步了解，請參閱 Amazon MWAA 上 VPC 的安全性。

4. 在環境類別下，選擇環境類別。

我們建議您選擇支援工作負載所需的最小大小。您可以隨時變更環境類別。

5. 針對工作者計數上限，指定要在環境中執行的 Apache Airflow 工作者數目上限。

如需詳細資訊，請參閱 高效能使用案例範例。

6. 指定 Web 伺服器計數上限和 Web 伺服器計數下限，以設定 Amazon MWAA 如何擴展您環境中的 Apache Airflow Web 伺服器。

如需 Web 伺服器自動擴展的詳細資訊，請參閱 the section called “設定 Web 伺服器自動擴展”。

7. 在加密下，選擇資料加密選項：

a. 根據預設，Amazon MWAA 會使用 擁有 AWS的金鑰來加密您的資料。

b. 「選用」。選擇自訂加密設定（進階）以選擇不同的 AWS KMS 金鑰。如果您選擇在此步驟中指定客戶受管金鑰，則必須指定 AWS KMS 金鑰 ID 或 ARN。AWS KMS Amazon MWAA 不支援別名和多區域金鑰。如果您在 Amazon S3 儲存貯體上為伺服器端加密指定了 Amazon S3 金鑰，則必須為 Amazon MWAA 環境指定相同的金鑰。

> **Note**
>
> 您必須擁有 金鑰的許可，才能在 Amazon MWAA 主控台上選取金鑰。您也必須透過連接 中所述的政策，授予 Amazon MWAA 使用金鑰的許可連接金鑰政策。

8. 建議使用。在監控下，為 Airflow 記錄組態選擇一或多個日誌類別，將 Apache Airflow 日誌傳送至 CloudWatch Logs：

   a. Airflow 任務日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 任務日誌類型。

   b. Airflow Web 伺服器日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow Webserver 日誌類型。

   c. Airflow 排程器日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 排程器日誌類型。

   d. 氣流工作者日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 工作者日誌類型。

   e. Airflow DAG 處理日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow DAG 處理日誌類型。

9. 「選用」。針對 Airflow 組態選項，選擇新增自訂組態選項。

   您可以從建議的 Apache Airflow 組態選項下拉式清單中選擇 Apache Airflow 版本，或指定自訂組態選項。例如，`core.default_task_retries：3`。

10. 「選用」。在標籤下，選擇新增標籤，將標籤與環境建立關聯。例如，`Environment：Staging`。

11. 在許可下，選擇執行角色：

    a. 根據預設，Amazon MWAA 會在建立新角色中建立執行角色。 您必須擁有建立 IAM 角色的許可，才能使用此選項。

    b. 「選用」。選擇輸入角色 ARN 以輸入現有執行角色的 Amazon Resource Name (ARN)。

12. 選擇下一步。

步驟三：檢閱並建立

檢閱環境摘要

- 檢閱環境摘要，選擇建立環境。

> **Note**
>
> 建立環境大約需要二十到三十分鐘。

後續步驟？

- 了解如何在 中建立 Amazon S3 儲存貯體為 Amazon MWAA 建立 Amazon S3 儲存貯體。

管理對 Amazon MWAA 環境的存取

必須允許 Amazon Managed Workflows for Apache Airflow 使用環境使用的其他 AWS 服務和資源。您也需要獲得在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 環境和 Apache Airflow UI 的許可。本節說明用來授予環境 AWS 資源存取權的執行角色，以及如何新增許可，以及存取 Amazon MWAA 環境和 Apache Airflow UI 所需的 AWS 帳戶 許可。

主題

- 存取 Amazon MWAA 環境
- Amazon MWAA 的服務連結角色
- Amazon MWAA 執行角色
- 預防跨服務混淆代理人
- Apache Airflow 存取模式

存取 Amazon MWAA 環境

若要使用 Amazon Managed Workflows for Apache Airflow，您必須使用具有必要許可的帳戶和 IAM 實體。本主題說明您可以連接到 Apache Airflow 開發團隊和 Amazon Managed Workflows for Apache Airflow 環境的 Apache Airflow 使用者存取政策。

我們建議您使用臨時登入資料並設定聯合身分與群組和角色，以存取您的 Amazon MWAA 資源。最佳實務是避免將政策直接連接到您的 IAM 使用者。反之，請定義群組或角色，以提供 AWS 資源的暫時存取權。

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。IAM 角色類似於 IAM 使用者，因為它是具有許可政策的 AWS 身分，可決定身分可以和不可以執行的操作 AWS。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。

若要將許可指派給聯合身分，您可以建立角色並定義角色的許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。

您可以使用帳戶中的 IAM 角色，授予另一個 AWS 帳戶 許可來存取您帳戶的資源。如需範例，請參閱《IAM 使用者指南》中的 IAM 教學課程： AWS 帳戶 使用 IAM 角色委派跨 的存取權。

章節

- 運作方式
- 完整主控台存取政策：AmazonMWAAFullConsoleAccess
- 完整 API 和主控台存取政策：AmazonMWAAFullApiAccess
- 唯讀主控台存取政策：AmazonMWAAReadOnlyAccess
- Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess
- Apache Airflow Rest API 存取政策：AmazonMWAARestAPIAccess
- Apache Airflow CLI 政策：AmazonMWAAAirflowCliAccess
- 建立 JSON 政策
- 將政策連接至開發人員群組的範例使用案例
- 後續步驟？

運作方式

Amazon MWAA 環境中使用的資源和服務無法供所有 AWS Identity and Access Management (IAM) 實體存取。您必須建立政策，授予 Apache Airflow 使用者存取這些資源的許可。例如，您需要將存取權授予 Apache Airflow 開發團隊。

Amazon MWAA 使用這些政策來驗證使用者是否具有在 AWS 主控台或透過環境使用的 APIs執行動作所需的許可。

您可以使用本主題中的 JSON 政策，為 IAM 中的 Apache Airflow 使用者建立政策，然後將政策連接到 IAM 中的使用者、群組或角色。

- AmazonMWAAFullConsoleAccess – 使用此政策授予在 Amazon MWAA 主控台上設定環境的許可。
- AmazonMWAAFullApiAccess – 使用此政策授予所有用於管理環境的 Amazon MWAA APIs 存取權。
- AmazonMWAAReadOnlyAccess – 使用此政策授予環境在 Amazon MWAA 主控台上使用的資源存取權。
- AmazonMWAAWebServerAccess – 使用此政策授予對 Apache Airflow Web 伺服器的存取權。
- AmazonMWAAAirflowCliAccess – 使用此政策授予執行 Apache Airflow CLI 命令的存取權。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center：

建立權限合集。請按照《AWS IAM Identity Center 使用者指南》中的建立權限合集說明進行操作。

- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。

- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的新增許可到使用者 (主控台) 中的指示。

完整主控台存取政策：AmazonMWAAFullConsoleAccess

如果需要在 Amazon MWAA 主控台上設定環境，使用者可能需要存取`AmazonMWAAFullConsoleAccess`許可政策。

> **Note**
>
> 您的完整主控台存取政策必須包含執行 的許可`iam:PassRole`。這可讓使用者將服務連結角色和執行角色傳遞給 Amazon MWAA。Amazon MWAA 會擔任每個角色來代表您呼叫其他 AWS 服務。下列範例使用 `iam:PassedToService`條件金鑰，將 Amazon MWAA 服務主體 (`airflow.amazonaws.com`) 指定為可傳遞角色的服務。
> 如需 的詳細資訊`iam:PassRole`，請參閱《IAM 使用者指南》中的授予使用者將角色傳遞至 AWS 服務的許可。

如果您想要使用 建立和管理靜態加密AWS 擁有的金鑰的 Amazon MWAA 環境，請使用下列政策。

使用 AWS 擁有的金鑰

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

如果您想要使用客戶受管金鑰建立和管理靜態加密的 Amazon MWAA 環境，請使用下列政策。若要使用客戶受管金鑰，IAM 主體必須具有使用您帳戶中存放的金鑰存取 AWS KMS 資源的許可。

使用客戶受管金鑰

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
```

```
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ListGrants",
```

```
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

完整 API 和主控台存取政策：AmazonMWAAFullApiAccess

如果使用者需要存取用於管理環境的所有 Amazon MWAA APIs則可能需要存取`AmazonMWAAFullApiAccess`許可政策。它不會授予存取 Apache Airflow UI 的許可。

> **Note**
>
> 完整的 API 存取政策必須包含執行 的許可`iam:PassRole`。這可讓使用者將服務連結角色和執行角色傳遞至 Amazon MWAA。Amazon MWAA 會擔任每個角色來代表您呼叫其他

AWS 服務。下列範例使用 iam:PassedToService條件金鑰，將 Amazon MWAA 服務主體 (airflow.amazonaws.com) 指定為可傳遞角色的服務。
如需 的詳細資訊iam:PassRole，請參閱《IAM 使用者指南》中的授予使用者將角色傳遞至 AWS 服務的許可。

如果您想要使用 建立和管理靜態加密 AWS 擁有的金鑰 的 Amazon MWAA 環境，請使用下列政策。

使用 AWS 擁有的金鑰

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
```

```
            "Action":[
               "ec2:DescribeSecurityGroups",
               "ec2:DescribeSubnets",
               "ec2:DescribeVpcs",
               "ec2:DescribeRouteTables"
            ],
            "Resource":"*"
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:GetEncryptionConfiguration"
            ],
            "Resource":"arn:aws:s3:::*"
         },
         {
            "Effect":"Allow",
            "Action":"ec2:CreateVpcEndpoint",
            "Resource":[
               "arn:aws:ec2:*:*:vpc-endpoint/*",
               "arn:aws:ec2:*:*:vpc/*",
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:security-group/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "ec2:CreateNetworkInterface"
            ],
            "Resource":[
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:network-interface/*"
            ]
         }
      ]
   }
```

如果您想要使用客戶受管金鑰建立和管理靜態加密的 Amazon MWAA 環境，請使用下列政策。若要使用客戶受管金鑰，IAM 主體必須具有使用您帳戶中存放的金鑰存取 AWS KMS 資源的許可。

使用客戶受管金鑰

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ListGrants",
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

唯讀主控台存取政策：AmazonMWAAReadOnlyAccess

如果使用者需要存取 Amazon MWAA 主控台環境詳細資訊頁面上環境使用的資源，則可能需要存取AmazonMWAAReadOnlyAccess許可政策。它不允許使用者建立新環境、編輯現有環境，或允許使用者存取 Apache Airflow UI。

JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:ListEnvironments",
                "airflow:GetEnvironment",
                "airflow:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess

如果使用者需要存取 Apache Airflow UI，則可能需要存取AmazonMWAAWebServerAccess許可政策。它不允許使用者存取 Amazon MWAA 主控台上的環境，或使用 Amazon MWAA APIs 執行任何動作。在 中指定 Admin、User、 OpViewer或 Public角色{airflow-role}，以自訂 Web 字符使用者的存取層級。如需詳細資訊，請參閱 Apache Airflow 參考指南中的預設角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
```

```
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-
name}/{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> - Amazon MWAA 提供 IAM 與五個預設 Apache Airflow 角色型存取控制 (RBAC) 角色的整合。如需使用自訂 Apache Airflow 角色的詳細資訊，請參閱 the section called “教學課程：將使用者限制為 DAGs的子集”。
> - 此政策中的 `Resource` 欄位可用來指定 Amazon MWAA 環境的 Apache Airflow 角色型存取控制角色。不過，它不支援政策 `Resource`欄位中的 Amazon MWAA 環境 ARN (Amazon Resource Name)。

Apache Airflow Rest API 存取政策：AmazonMWAARestAPIAccess

若要存取 Apache Airflow REST API，您必須在 IAM 政策中授予 `airflow:InvokeRestApi`許可。在下列政策範例中，在 中指定 `Admin`、`User`、 `OpViewer`或 `Public`角色，`{airflow-role}`以自訂使用者存取層級。如需詳細資訊，請參閱 Apache Airflow 參考指南中的預設角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
```

```
    ]
}
```

> **Note**
>
> - 設定私有 Web 伺服器時，無法從 Virtual Private Cloud (VPC) 外部叫用 `InvokeRestApi`動作。您可以使用 `aws:SourceVpc`金鑰，為此操作套用更精細的存取控制。如需詳細資訊，請參閱 aws：SourceVpc
> - 此政策中的 `Resource` 欄位可用來指定 Amazon MWAA 環境的 Apache Airflow 角色型存取控制角色。不過，它不支援政策 `Resource`欄位中的 Amazon MWAA 環境 ARN (Amazon Resource Name)。

Apache Airflow CLI 政策：AmazonMWAAAirflowCliAccess

如果使用者需要執行 Apache Airflow CLI 命令（例如），則可能需要存取`AmazonMWAAAirflowCliAccess`許可政策`trigger_dag`。它不允許使用者存取 Amazon MWAA 主控台上的環境，或使用 Amazon MWAA APIs 執行任何動作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
${EnvironmentName}"
        }
    ]
}
```

建立 JSON 政策

您可以在 IAM 主控台上建立 JSON 政策，並將政策連接至您的使用者、角色或群組。下列步驟說明如何在 IAM 中建立 JSON 政策。

建立 JSON 政策

1. 在 IAM 主控台上開啟政策頁面。
2. 選擇建立政策。
3. 請選擇 JSON 標籤。
4. 新增您的 JSON 政策。
5. 選擇檢閱政策。
6. 在名稱和描述的文字欄位中輸入值 （選用）。

   例如，您可以命名政策 `AmazonMWAAReadOnlyAccess`。
7. 選擇建立政策。

將政策連接至開發人員群組的範例使用案例

假設您正在 IAM 中使用名為 的群組`AirflowDevelopmentGroup`，將許可套用至 Apache Airflow 開發團隊中的所有開發人員。這些使用者需要存取 `AmazonMWAAFullConsoleAccess`、`AmazonMWAAAirflowCliAccess`和 `AmazonMWAAWebServerAccess`許可政策。本節說明如何在 IAM 中建立群組、建立和連接這些政策，並將群組與 IAM 使用者建立關聯。這些步驟假設您使用 AWS 擁有的金鑰。

建立 AmazonMWAAFullConsoleAccess 政策

1. 下載 AmazonMWAAFullConsoleAccess 存取政策。
2. 在 IAM 主控台上開啟政策頁面。
3. 選擇建立政策。
4. 請選擇 JSON 標籤。
5. 貼上 的 JSON 政策`AmazonMWAAFullConsoleAccess`。
6. 取代下列值：

   a. *123456789012* – 您的 AWS 帳戶 ID （例如 0123456789)

b. *{your-kms-id}* – 客戶受管金鑰的唯一識別符，僅適用於使用客戶受管金鑰進行靜態加密的情況。

7. 選擇檢閱政策。
8. 在名稱AmazonMWAAFullConsoleAccess中輸入 。
9. 選擇建立政策。

建立 AmazonMWAAWebServerAccess 政策

1. 下載 AmazonMWAAWebServerAccess 存取政策。
2. 在 IAM 主控台上開啟政策頁面。
3. 選擇建立政策。
4. 請選擇 JSON 標籤。
5. 貼上 的 JSON 政策AmazonMWAAWebServerAccess。
6. 取代下列值：

   a. *us-east-1* – Amazon MWAA 環境的區域 （ 例如 us-east-1)

   b. *123456789012* – 您的 AWS 帳戶 ID （ 例如 0123456789)

   c. *{your-environment-name}* – 您的 Amazon MWAA 環境名稱 （ 例如 MyAirflowEnvironment)

   d. *{airflow-role}* – Admin Apache Airflow 預設角色

7. 選擇檢閱政策。
8. 在名稱AmazonMWAAWebServerAccess中輸入 。
9. 選擇建立政策。

建立 AmazonMWAAAirflowCliAccess 政策

1. 下載 AmazonMWAAAirflowCliAccess 存取政策。
2. 在 IAM 主控台上開啟政策頁面。
3. 選擇建立政策。
4. 請選擇 JSON 標籤。
5. 貼上 的 JSON 政策AmazonMWAAAirflowCliAccess。
6. 選擇檢閱政策。

7. 在名稱AmazonMWAAAirflowCliAccess中輸入 。
8. 選擇建立政策。

建立 群組

1. 在 IAM 主控台上開啟群組頁面。
2. 輸入 的名稱AirflowDevelopmentGroup。
3. 選擇 Next Step (後續步驟)。
4. 輸入 AmazonMWAA以篩選篩選條件中的結果。
5. 選取您建立的三個政策。
6. 選擇 Next Step (後續步驟)。
7. 選擇 Create Group (建立群組)。

與使用者建立關聯

1. 在 IAM 主控台上開啟使用者頁面。
2. 選擇使用者。
3. 選擇 Groups (群組)。
4. 選擇將使用者新增至群組。
5. 選取 AirflowDevelopmentGroup。
6. 選擇 Add to Groups (新增至群組)。

後續步驟？

- 了解如何在 中產生權杖以存取 Apache Airflow UI存取 Apache Airflow。
- 進一步了解如何在建立 IAM 政策中建立 IAM 政策。

Amazon MWAA 的服務連結角色

Amazon Managed Workflows for Apache Airflow 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Amazon MWAA 的唯一 IAM 角色類型。服務連結角色是由 Amazon MWAA 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 Amazon MWAA，因為您不需要手動新增必要的許可。Amazon MWAA 會定義其服務連結角色的許可，除非另有定義，否則只有 Amazon MWAA 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 Amazon MWAA 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他 服務的資訊，請參閱AWS 使用 IAM 的服務，並在服務連結角色欄中搜尋具有是的服務。選擇有連結的是，以存取該服務的服務連結角色文件。

Amazon MWAA 的服務連結角色許可

Amazon MWAA 使用名為 的服務連結角色 `AWSServiceRoleForAmazonMWAA` – 在您帳戶中建立的服務連結角色會授予 Amazon MWAA 對下列 AWS 服務的存取權：

- Amazon CloudWatch Logs (CloudWatch Logs) – 為 Apache Airflow 日誌建立日誌群組。
- Amazon CloudWatch (CloudWatch) – 將與您的環境及其基礎元件相關的指標發佈至您的帳戶。
- Amazon Elastic Compute Cloud (Amazon EC2) – 建立下列資源：
  - 您 VPC 中要供 Apache Airflow 排程器和工作者使用的受 AWS管 Amazon Aurora PostgreSQL 資料庫叢集的 Amazon VPC 端點。
  - 如果您為 Apache Airflow Web 伺服器選擇私有網路選項，則可使用額外的 Amazon VPC 端點來啟用 Web 伺服器的網路存取。
  - Amazon VPC 中的彈性網路界面 (ENIs) 可讓您的網路存取 Amazon VPC 中託管 AWS 的資源。

下列信任政策允許服務主體擔任服務連結角色。Amazon MWAA 的服務主體`airflow.amazonaws.com`如政策所示。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "airflow.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

名為 的角色許可政策AmazonMWAAServiceRolePolicy允許 Amazon MWAA 對指定的資源完成下列動作：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:airflow-*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpoint",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonMWAAManaged": false
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:ModifyVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/MWAA"
                    ]
                }
            }
        }
    ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

為 Amazon MWAA 建立服務連結角色

您不需要手動建立服務連結角色，當您使用 AWS 管理主控台、 AWS CLI或 AWS API 建立新的 Amazon MWAA 環境時，Amazon MWAA 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立另一個環境時，Amazon MWAA 會再次為您建立服務連結角色。

編輯 Amazon MWAA 的服務連結角色

Amazon MWAA 不允許編輯 AWSServiceRoleForAmazonMWAA 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 Amazon MWAA 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。

當您刪除 Amazon MWAA 環境時，Amazon MWAA 會刪除它做為服務一部分使用的所有相關資源。不過，您必須先等待 Amazon MWAA 完成刪除您的環境，然後再嘗試刪除服務連結角色。如果您在

Amazon MWAA 刪除環境之前刪除服務連結角色，Amazon MWAA 可能無法刪除所有環境的相關聯資源。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForAmazonMWAA 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Amazon MWAA 服務連結角色支援的區域

Amazon MWAA 支援在提供服務的所有區域中使用服務連結角色。如需詳細資訊，請參閱 Amazon Managed Workflows for Apache Airflow 端點和配額。

政策更新

變更	描述	Date
Amazon MWAA 會更新其服務連結角色許可政策	AmazonMWAAServiceRolePolicy – Amazon MWAA 會更新其服務連結角色的許可政策，以授予 Amazon MWAA 許可，將與服務基礎資源相關的其他指標發佈至客戶帳戶。這些新指標發佈於 AWS/MWAA	2022 年 11 月 18 日
Amazon MWAA 已開始追蹤變更	Amazon MWAA 開始追蹤其AWS受管服務連結角色許可政策的變更。	2022 年 11 月 18 日

Amazon MWAA 執行角色

執行角色是具有許可政策的 AWS Identity and Access Management (IAM) 角色，授予 Amazon Managed Workflows for Apache Airflow AWS 代表您調用其他服務資源的許可。這可能包括您的 Amazon S3 儲存貯體、 AWS擁有的金鑰和 CloudWatch Logs 等資源。Amazon MWAA 環境每個環境都需要一個執行角色。本主題說明如何使用和設定您環境的執行角色，以允許 Amazon MWAA 存取您環境使用的其他 AWS 資源。

內容

- 執行角色概觀
  - 預設連接的許可
  - 如何新增使用其他服務的許可 AWS
  - 如何關聯新的執行角色
- Create a new role (建立新角色)
- 存取和更新執行角色政策
  - 連接 JSON 政策以使用其他 AWS 服務
- 使用帳戶層級公有存取區塊授予 Amazon S3 儲存貯體的存取權
- 使用 Apache Airflow 連線
- 執行角色的 JSON 政策範例
  - 客戶受管金鑰的範例政策
  - AWS擁有金鑰的範例政策
- 後續步驟？

執行角色概觀

Amazon MWAA 使用您環境 AWS 所用其他服務的許可來自執行角色。Amazon MWAA 執行角色需要環境使用的下列 AWS 服務許可：

- Amazon CloudWatch (CloudWatch) – 傳送 Apache Airflow 指標和日誌。
- Amazon Simple Storage Service (Amazon S3) – 剖析您環境的 DAG 程式碼和支援檔案 （例如 `requirements.txt`)。
- Amazon Simple Queue Service (Amazon SQS) – 在 Amazon MWAA 擁有的 Amazon SQS 佇列中將環境的 Apache Airflow 任務排入佇列。
- AWS Key Management Service (AWS KMS) – 適用於您環境的資料加密 （使用 AWS擁有的金鑰或客戶管理的金鑰)。

> **Note**
>
> 如果您已選擇讓 Amazon MWAA 使用 AWS 擁有的 KMS 金鑰來加密資料，則必須在連接至 Amazon MWAA 執行角色的政策中定義許可，以透過 Amazon SQS 授予帳戶外部存放之任意 KMS 金鑰的存取權。需要以下兩個條件，您環境的執行角色才能存取任意 KMS 金鑰：
>
> - 第三方帳戶中的 KMS 金鑰需要透過其資源政策允許此跨帳戶存取。

- 您的 DAG 程式碼需要存取在第三方帳戶中以 開頭的 Amazon SQS 佇列`airflow-celery-`，並使用相同的 KMS 金鑰進行加密。

為了降低與跨帳戶存取 資源相關的風險，我們建議您檢閱 DAGs 中放置的程式碼，以確保您的工作流程不會存取您帳戶外部的任意 Amazon SQS 佇列。此外，您可以使用存放在您自己帳戶中的客戶受管 KMS 金鑰來管理 Amazon MWAA 上的加密。這會將您環境的執行角色限制為僅存取您帳戶中的 KMS 金鑰。

請記住，選擇加密選項後，您無法變更現有環境的選擇。

執行角色還需要下列 IAM 動作的許可：

- `airflow:PublishMetrics` – 允許 Amazon MWAA 監控環境的運作狀態。

預設連接的許可

您可以使用 Amazon MWAA 主控台上的預設選項來建立執行角色和 AWS擁有的金鑰，然後使用此頁面上的步驟將許可政策新增至您的執行角色。

- 當您在主控台上選擇建立新角色選項時，Amazon MWAA 會將環境所需的最低許可連接至您的執行角色。
- 在某些情況下，Amazon MWAA 會連接最大許可。例如，我們建議您在建立環境時，選擇 Amazon MWAA 主控台上的 選項來建立執行角色。Amazon MWAA 會使用執行角色中的 regex 模式作為 ，自動新增所有 CloudWatch Logs 群組的許可政策`"arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-name-*"`。

如何新增使用其他服務的許可 AWS

建立環境後，Amazon MWAA 無法新增或編輯許可政策至現有的執行角色。您必須使用環境所需的其他許可政策來更新您的執行角色。例如，如果您的 DAG 需要存取 AWS Glue，Amazon MWAA 無法自動偵測您的環境所需的這些許可，或將許可新增至您的執行角色。

您可以透過兩種方式將許可新增至執行角色：

- 透過為執行角色內嵌修改 JSON 政策。您可以使用此頁面上的範例 JSON 政策文件，在 IAM 主控台上新增或取代執行角色的 JSON 政策。
- 為 AWS 服務建立 JSON 政策，並將其連接至您的執行角色。您可以使用此頁面上的步驟，將 AWS 服務的新 JSON 政策文件與 IAM 主控台上的執行角色建立關聯。

假設執行角色已與您的環境相關聯，Amazon MWAA 可以立即開始使用新增的許可政策。這也表示如果您從執行角色移除任何必要的許可，您的 DAGs可能會失敗。

如何關聯新的執行角色

您可以隨時變更環境的執行角色。如果新的執行角色尚未與您的環境建立關聯，請使用此頁面的步驟建立新的執行角色政策，並將角色與您的環境建立關聯。

Create a new role (建立新角色)

根據預設，Amazon MWAA 會代表您建立 AWS擁有的金鑰以進行資料加密和執行角色。您可以在建立環境時選擇 Amazon MWAA 主控台上的預設選項。下圖顯示為環境建立執行角色的預設選項。

> **Important**
>
> 當您建立新的執行角色時，請勿重複使用已刪除執行角色的名稱。唯一名稱有助於防止衝突，並確保適當的資源管理。

存取和更新執行角色政策

您可以在 Amazon MWAA 主控台上存取您環境的執行角色，並在 IAM 主控台上更新角色的 JSON 政策。

更新執行角色政策

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在許可窗格中選擇執行角色，以在 IAM 中開啟許可頁面。
4. 選擇執行角色名稱以開啟許可政策。
5. 選擇 Edit Policy (編輯政策)。
6. 選擇 JSON 標籤。
7. 更新您的 JSON 政策。
8. 選擇檢閱政策。
9. 選擇儲存變更。

連接 JSON 政策以使用其他 AWS 服務

您可以為 AWS 服務建立 JSON 政策，並將其連接至您的執行角色。例如，您可以連接下列 JSON 政策，授予 中所有資源的唯讀存取權 AWS Secrets Manager。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "secretsmanager:GetResourcePolicy",
            "secretsmanager:GetSecretValue",
            "secretsmanager:DescribeSecret",
            "secretsmanager:ListSecretVersionIds"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

將政策連接至執行角色

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在許可窗格中選擇您的執行角色。
4. 選擇連接政策。
5. 選擇 Create policy (建立政策)。
6. 選擇 JSON。
7. 貼上 JSON 政策。
8. 選擇下一步：標籤、下一步：檢閱。
9. 輸入政策的描述性名稱 （ 例如 `SecretsManagerReadPolicy`) 和描述。
10. 選擇建立政策。

使用帳戶層級公有存取區塊授予 Amazon S3 儲存貯體的存取權

您可能想要使用 Amazon S3 PutPublicAccessBlock 操作來封鎖對您帳戶中所有儲存貯體的存取。當您封鎖對帳戶中所有儲存貯體的存取時，您的環境執行角色必須在許可政策中包含 s3:GetAccountPublicAccessBlock動作。

下列範例示範當封鎖對您帳戶中所有 Amazon S3 儲存貯體的存取時，您必須連接至執行角色的政策。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": "s3:GetAccountPublicAccessBlock",
       "Resource": "*"
     }
  ]
}
```

如需限制存取 Amazon S3 儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的封鎖對 Amazon S3 儲存體的公開存取。

使用 Apache Airflow 連線

您也可以建立 Apache Airflow 連線，並在 Apache Airflow 連線物件中指定執行角色及其 ARN。若要進一步了解，請參閱 管理 Apache Airflow 的連線。

執行角色的 JSON 政策範例

您可以使用本節中的兩個範例許可政策來取代用於現有執行角色的許可政策，或建立新的執行角色並用於您的環境。這些政策包含 Apache Airflow 日誌群組的資源 ARN 預留位置、Amazon S3 儲存貯體和 Amazon MWAA 環境。

我們建議複製範例政策、取代範例 ARNs 或預留位置，然後使用 JSON 政策來建立或更新執行角色。

客戶受管金鑰的範例政策

下列範例提供可用於客戶受管金鑰的執行角色政策。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": "s3:ListAllMyBuckets",
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
   ],
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
   ],
   "Resource": [
    "arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-
name:*"
   ]
  },
  {
   "Effect": "Allow",
```

```
    "Action": [
     "logs:DescribeLogGroups"
    ],
    "Resource": [
     "*"
    ]
   },
   {
    "Effect": "Allow",
    "Action": [
     "s3:GetAccountPublicAccessBlock"
    ],
    "Resource": [
     "*"
    ]
   },
   {
    "Effect": "Allow",
    "Action": "cloudwatch:PutMetricData",
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "sqs:ChangeMessageVisibility",
     "sqs:DeleteMessage",
     "sqs:GetQueueAttributes",
     "sqs:GetQueueUrl",
     "sqs:ReceiveMessage",
     "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "kms:Decrypt",
     "kms:DescribeKey",
     "kms:GenerateDataKey*",
     "kms:Encrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:111122223333:key/your-kms-cmk-id",
    "Condition": {
     "StringLike": {
```

```
      "kms:ViaService": [
       "sqs.us-east-1.amazonaws.com",
       "s3.us-east-1.amazonaws.com"
      ]
     }
    }
   }
 ]
}
```

接著，您需要允許 Amazon MWAA 擔任此角色，以代表您執行動作。這可以透過使用 IAM 主控台將 "airflow.amazonaws.com" "airflow-env.amazonaws.com"和服務主體新增至此執行角色的信任實體清單，或使用 將這些服務主體放入此執行角色的擔任角色政策文件中https://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html AWS CLI。請參閱下列擔任角色政策文件範例：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
            "Service": ["airflow.amazonaws.com","airflow-env.amazonaws.com"]
        },
        "Action": "sts:AssumeRole"
      }
   ]
}
```

然後將下列 JSON 政策連接至您的客戶受管金鑰。此政策使用 kms:EncryptionContext 條件金鑰字首，以允許存取 CloudWatch Logs 中的 Apache Airflow 日誌群組。

```
{
  "Sid": "Allow logs access",
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
```

```
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:*"
    }
  }
}
```

AWS擁有金鑰的範例政策

下列範例顯示您可以用於 AWS擁有的金鑰的執行角色政策。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
{your-environment-name}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
```

```
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:log-group:airflow-{your-
environment-name}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

後續步驟？

- 了解您和您的 Apache Airflow 使用者在 中存取您的環境所需的許可[存取 Amazon MWAA 環境](#)。
- 了解 [使用客戶受管金鑰進行加密](#)。
- 探索更多[客戶受管政策範例](#)。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 中 AWS，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務，以使用其許可對其他客戶的資源採取動作，方式是它沒有存取的許可。為了避免這種情況， AWS 提供的工具可協助您保護所有 服務的資料，其服務主體已獲得您帳戶中資源的存取權。

我們建議您在環境的執行角色中使用 aws:SourceArn和 aws:SourceAccount全域條件內容金鑰，以限制 Amazon MWAA 提供其他服務存取資源的許可。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。

防範混淆代理人問題的最有效方法是使用 aws:SourceArn 全域條件內容索引鍵，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 全域內容條件索引鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如 arn:aws:airflow:*:*123456789012*:environment/*。

的值aws:SourceArn必須是您要為其建立執行角色的 Amazon MWAA 環境 ARN。

使用下列範例，在環境的執行角色信任政策中套用 aws:SourceArn和 aws:SourceAccount全域條件內容金鑰，以防止混淆代理人問題。您可以在建立新的執行角色時使用下列信任政策。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "airflow.amazonaws.com",
                    "airflow-env.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:airflow:us-
east-1:123456789012:environment/your-environment-name"
```

```
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Apache Airflow 存取模式

Amazon Managed Workflows for Apache Airflow 主控台包含內建選項，可設定您環境中 Apache Airflow Web 伺服器的私有或公有路由。本指南說明 Amazon Managed Workflows for Apache Airflow 環境上 Apache Airflow Web 伺服器可用的存取模式，以及如果您選擇私有網路選項，則需要在 Amazon VPC 中設定的其他資源。

內容

- Apache Airflow 存取模式
  - 公有網路
  - 私有網路
- 存取模式概觀
  - 公有網路存取模式
  - 私有網路存取模式
- 私有和公有存取模式的設定
  - 公有網路的設定
  - 私有網路的設定
- 存取 Apache Airflow Web 伺服器的 VPC 端點 （私有網路存取）

Apache Airflow 存取模式

您可以為 Apache Airflow Web 伺服器選擇私有或公有路由。若要啟用私有路由，請選擇私有網路。這會限制使用者存取 Amazon VPC 內的 Apache Airflow Web 伺服器。若要啟用公有路由，請選擇公有網路。這可讓使用者透過網際網路存取 Apache Airflow Web 伺服器。

公有網路

下列架構圖說明具有公有 Web 伺服器的 Amazon MWAA 環境。

公有網路存取模式允許授予您環境 IAM 政策存取權的使用者透過網際網路存取 Apache Airflow UI。

下圖說明在 Amazon MWAA 主控台上尋找公有網路選項的位置。

私有網路

下列架構圖說明具有私有 Web 伺服器的 Amazon MWAA 環境。

私有網路存取模式會將對 Apache Airflow UI 的存取限制在 Amazon VPC 中已授予您環境 IAM 政策存取權的使用者。

當您建立具有私有 Web 伺服器存取權的環境時，您必須在 Python wheel 封存檔 (`.whl`) 中封裝所有相依性，然後在 `.whl`中參考 `requirements.txt`。如需使用 wheel 封裝和安裝相依性的說明，請參閱使用 Python wheel 管理相依性。

下圖說明在 Amazon MWAA 主控台上尋找私有網路選項的位置。

存取模式概觀

本節說明當您選擇公有網路或私有網路存取模式時，在 Amazon VPC 中建立的 VPC 端點 (AWS PrivateLink)。

公有網路存取模式

如果您為 Apache Airflow Web 伺服器選擇公有網路存取模式，網路流量會透過網際網路公開路由。

- Amazon MWAA 會為您的 Amazon Aurora PostgreSQL 中繼資料資料庫建立 VPC 介面端點。端點是在映射到您的私有子網路的可用區域中建立的，並且獨立於其他子網路 AWS 帳戶。
- 然後，Amazon MWAA 會將私有子網路的 IP 地址繫結至介面端點。這旨在支援從 Amazon VPC 的每個可用區域繫結單一 IP 的最佳實務。

私有網路存取模式

如果您為 Apache Airflow Web 伺服器選擇私有網路存取模式，網路流量會在 Amazon VPC 內私下路由。

- Amazon MWAA 會為您的 Apache Airflow Web 伺服器建立 VPC 介面端點，並為 Amazon Aurora PostgreSQL 中繼資料資料庫建立介面端點。端點會在映射至您私有子網路的可用區域中建立，並獨立於其他子網路 AWS 帳戶。
- 然後，Amazon MWAA 會將私有子網路的 IP 地址繫結至介面端點。這旨在支援從 Amazon VPC 的每個可用區域繫結單一 IP 的最佳實務。

若要進一步了解，請參閱 the section called “Amazon VPC 和 Apache Airflow 存取模式的範例使用案例”。

私有和公有存取模式的設定

下一節說明根據您為環境選擇的 Apache Airflow 存取模式，您需要的其他設定和組態。

公有網路的設定

如果您為 Apache Airflow Web 伺服器選擇公有網路選項，您可以在建立環境後開始使用 Apache Airflow UI。

您需要採取下列步驟，為您的使用者設定存取權，以及您的環境使用其他 AWS 服務的許可。

1. 新增許可。Amazon MWAA 需要許可才能使用其他 AWS 服務。當您建立環境時，Amazon MWAA 會建立服務連結角色，允許它針對 Amazon Elastic Container Registry (Amazon ECR)、CloudWatch Logs 和 Amazon EC2 使用特定 IAM 動作。

   您可以新增許可，以使用這些服務的其他動作，或透過將許可新增至執行角色 AWS 來使用其他服務。若要進一步了解，請參閱 Amazon MWAA 執行角色。
2. 建立使用者政策。您可能需要為使用者建立多個 IAM 政策，以設定對您環境和 Apache Airflow UI 的存取。若要進一步了解，請參閱 存取 Amazon MWAA 環境。

私有網路的設定

如果您選擇 Apache Airflow Webserver 的私有網路選項，則需要為使用者設定存取權、環境使用其他服務的許可 AWS ，以及建立從電腦存取 Amazon VPC 中資源的機制。

1. 新增許可。Amazon MWAA 需要許可才能使用其他 AWS 服務。當您建立環境時，Amazon MWAA 會建立服務連結角色，允許它針對 Amazon Elastic Container Registry (Amazon ECR)、CloudWatch Logs 和 Amazon EC2 使用特定 IAM 動作。

   您可以新增許可，以使用這些服務的其他動作，或透過將許可新增至執行角色 AWS 來使用其他服務。若要進一步了解，請參閱 Amazon MWAA 執行角色。
2. 建立使用者政策。您可能需要為使用者建立多個 IAM 政策，以設定對您環境和 Apache Airflow UI 的存取。若要進一步了解，請參閱 存取 Amazon MWAA 環境。
3. 啟用網路存取。您需要在 Amazon VPC 中建立機制，才能連線至 Apache Airflow Web 伺服器的 VPC 端點 (AWS PrivateLink)。例如，透過使用 從您的電腦建立 VPN 通道 AWS Client VPN。

存取 Apache Airflow Web 伺服器的 VPC 端點 （私有網路存取）

如果您已選擇私有網路選項，則需要在 Amazon VPC 中建立機制，以存取 Apache Airflow Web 伺服器的 VPC 端點 (AWS PrivateLink)。建議您針對這些資源使用與 Amazon MWAA 環境相同的 Amazon VPC、VPC 安全群組和私有子網路。

若要進一步了解，請參閱管理 VPC 端點的存取。

存取 Apache Airflow

Amazon MWAA 可讓您使用多種方法存取 Apache Airflow 環境：Apache Airflow 使用者介面 (UI) 主控台、Apache Airflow CLI 和 Apache Airflow REST API。您可以使用 Amazon MWAA 主控台來存取和叫用 Apache Airflow UI 中的 DAG，或使用 Amazon MWAA APIs 來取得權杖和叫用 DAG。本節說明存取 Apache Airflow UI 所需的許可、如何在命令 Shell 中直接產生 Amazon MWAA API 呼叫的權杖，以及 Apache Airflow CLI 中支援的命令。

主題

- 先決條件
- 開啟 Apache Airflow UI
- 登入 Apache Airflow
- 建立 Apache Airflow Web 伺服器存取字符
- 設定 Apache Airflow Web 伺服器的自訂網域
- 建立 Apache Airflow CLI 字符
- 使用 Apache Airflow REST API
- Apache Airflow CLI 命令參考

先決條件

下一節說明使用本節中的命令和指令碼所需的初步步驟。

存取

- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess。
- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策 完整 API 和主控台存取政策：AmazonMWAAFullApiAccess。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。

- AWS CLI – 使用 的快速組態`aws configure`。

開啟 Apache Airflow UI

下圖顯示 Amazon MWAA 主控台上 Apache Airflow UI 的連結。

登入 Apache Airflow

您需要 AWS 帳戶 in AWS Identity and Access Management (IAM) 存取 Apache Airflow UI 的Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess許可。

存取您的 Apache Airflow UI

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇開啟氣流使用者介面。

建立 Apache Airflow Web 伺服器存取字符

您可以使用此頁面上的命令來建立 Web 伺服器存取字符。存取權杖可讓您存取 Amazon MWAA 環境。例如，您可以取得權杖，然後使用 Amazon MWAA APIs 以程式設計方式部署 DAGs。下一節包含使用 AWS CLI、Bash 指令碼、POST API 請求或 Python 指令碼建立 Apache Airflow Web 登入字符的步驟。回應中傳回的字符有效期為 60 秒。

> **Important**
>
> 自 2025 年 8 月 19 日起，Amazon MWAA 新增對 IPv6 端點的支援，現在支援 IPv4 和 IPv6 端點。截至此日期，所有新建立的環境都會使用 Airflow 使用者介面 (UI) 的`.on.aws`網域。客戶必須將其 Airflow UI 從 遷移`.amazonaws.com`至這些新建立環境的`.on.aws`網域。Web 伺服器和資料庫的虛擬私有雲端 (VPC) 端點服務會維護其目前的`.amazonaws.com`網域，而不需要變更。

內容

- 先決條件

  - 存取
  - AWS CLI
- 使用 AWS CLI
- 使用 bash 指令碼
- 使用 Python 指令碼
- 後續步驟？

先決條件

下一節說明使用此頁面上的命令和指令碼所需的初步步驟。

存取

- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess。
- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策 完整 API 和主控台存取政策：AmazonMWAAFullApiAccess。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

使用 AWS CLI

下列範例使用 中的 create-web-login-token 命令 AWS CLI 來建立 Apache Airflow Web 登入字符。

```
aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME
```

使用 bash 指令碼

下列範例使用 bash 指令碼呼叫 中的 create-web-login-token 命令 AWS CLI ，以建立 Apache Airflow Web 登入字符。

1. 複製下列程式碼範例的內容，並在本機儲存為 get-web-token.sh。

```
#!/bin/bash
HOST=YOUR_HOST_NAME
YOUR_URL=https://$HOST/aws_mwaa/aws-console-sso?login=true#
WEB_TOKEN=$(aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME --query
 WebToken --output text)
echo $YOUR_URL$WEB_TOKEN
```

2. 以##取代 YOUR_HOST_NAME和 的預留位置YOUR_ENVIRONMENT_NAME。例如，公有網路的主機名稱類似（不含 https：//）：

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. （選用） macOS 和 Linux 使用者可能需要執行下列命令，以確保指令碼可執行。

```
chmod +x get-web-token.sh
```

4. 執行下列指令碼以取得 Web 登入字符。

```
./get-web-token.sh
```

您的命令提示字元會顯示：

```
https://123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com/
aws_mwaa/aws-console-sso?login=true#{your-web-login-token}
```

使用 Python 指令碼

下列範例使用 Python 指令碼中的 boto3 create_web_login_token 方法建立 Apache Airflow Web 登入字符。您可以在 Amazon MWAA 外部執行此指令碼。您唯一需要做的就是安裝 boto3 程式庫。您可能想要建立虛擬環境來安裝程式庫。其假設您已為 帳戶設定身分 AWS 驗證憑證。

1. 複製下列程式碼範例的內容，並在本機儲存為 create-web-login-token.py。

```
import boto3
  mwaa = boto3.client('mwaa')
  response = mwaa.create_web_login_token(
    Name="YOUR_ENVIRONMENT_NAME"
  )
```

```
  webServerHostName = response["WebServerHostname"]
  webToken = response["WebToken"]
  airflowUIUrl = 'https://{0}/aws_mwaa/aws-console-sso?
login=true#{1}'.format(webServerHostName, webToken)
  print("Here is your Airflow UI URL: ")
  print(airflowUIUrl)
```

2. 以##取代 的預留位置YOUR_ENVIRONMENT_NAME。
3. 執行下列指令碼以取得 Web 登入字符。

```
python3 create-web-login-token.py
```

後續步驟？

- 探索用於在 CreateWebLoginToken 建立 Web 登入字符的 Amazon MWAA API 操作。

設定 Apache Airflow Web 伺服器的自訂網域

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) 可讓您為受管 Apache Airflow Web 伺服器設定自訂網域。使用自訂網域，您可以使用 Apache Airflow UI、Apache Airflow CLI 或 Apache Airflow Web 伺服器存取您環境的 Amazon MWAA 受管 Apache Airflow Web 伺服器。

> **Note**
>
> 您只能將自訂網域與沒有網際網路存取的私有 Web 伺服器搭配使用。

Amazon MWAA 上自訂網域的使用案例

1. 在 上跨雲端應用程式共用 Web 伺服器網域 AWS - 使用自訂網域可讓您定義易於存取 Web 伺服器的 URL，而不是產生的服務網域名稱。您可以存放此自訂網域，並將其做為環境變數在應用程式中共用。
2. 存取私有 Web 伺服器 — 如果您想要在 VPC 中為沒有網際網路存取的 Web 伺服器設定存取，使用自訂網域可簡化 URL 重新導向工作流程。

主題

- 設定自訂網域

- 設定網路基礎設施

設定自訂網域

若要設定自訂網域功能，您需要在建立或更新 Amazon MWAA 環境時，透過 webserver.base_url Apache Airflow 組態提供自訂網域值。下列限制條件適用於您的自訂網域名稱：

- 此值必須是不含任何通訊協定或路徑的完整網域名稱 (FQDN)。例如 your-custom-domain.com。
- Amazon MWAA 不允許 URL 中的路徑。例如， your-custom-domain.com/dags/ 不是有效的自訂網域名稱。
- URL 長度限制為 255 個 ASCII 字元。
- 如果您提供空字串，預設會使用 Amazon MWAA 產生的 Web 伺服器 URL 建立環境。

使用以下範例，使用 建立具有自訂 Web 伺服器網域名稱的環境 AWS CLI。

```
aws mwaa create-environment \
--name my-mwaa-env \
--source-bucket-arn arn:aws:s3:::amzn-s3-demo-bucket \
--airflow-configuration-options '{"webserver.base_url":"my-custom-domain.com"}' \
--network-configuration '{"SubnetIds":["subnet-0123456789abcdef","subnet-
fedcba9876543210"]}' \
--execution-role-arn arn:aws:iam::123456789012:role/my-execution-role
```

建立或更新環境後，您需要在 中設定網路基礎設施 AWS 帳戶 ，才能透過自訂網域存取私有 Web 伺服器。

若要還原至預設服務產生的 URL，請更新您的私有環境並移除webserver.base_url組態選項。

設定網路基礎設施

使用下列步驟來設定必要的聯網基礎設施，以與 中的自訂網域搭配使用 AWS 帳戶。

1. 取得 Amazon VPC 端點網路界面 (ENI) 的 IP 地址。若要這樣做，請先使用 get-environment 為您的環境尋找 WebserverVpcEndpointService 。

   ```
   aws mwaa get-environment --name your-environment-name
   ```

如果成功，您會收到類似以下的輸出。

```
{
  "Environment": {
    "AirflowConfigurationOptions": {},
    "AirflowVersion": "latest-version",
    "Arn": "environment-arn",
    "CreatedAt": "2024-06-01T01:00:00-00:00",
    "DagS3Path": "dags",
    .
    .
    .
    "WebserverVpcEndpointService": "web-server-vpc-endpoint-service",
    "WeeklyMaintenanceWindowStart": "TUE:21:30"
  }
}
```

請記下 WebserverVpcEndpointService值，並在下列 Amazon EC2 describe-vpc-endpoints命令web-server-vpc-endpoint-service中將其用於 。在下列命令--filters Name=service-name,Values=*web-server-vpc-endpoint-service-id*中用於 。

2. 擷取 Amazon VPC 端點詳細資訊。此命令會擷取符合特定服務名稱的 Amazon VPC 端點詳細資訊，並以文字格式傳回端點 ID 和相關聯的網路介面 IDs。

```
aws ec2 describe-vpc-endpoints \
 --filters Name=service-name,Values=web-server-vpc-endpoint-service \
 --query 'VpcEndpoints[*].
{EndpointId:VpcEndpointId,NetworkInterfaceIds:NetworkInterfaceIds}' \
 --output text
```

3. 取得網路介面詳細資訊。此命令會擷取與上一個步驟中識別的 Amazon VPC 端點相關聯的每個網路介面的私有 IP 地址。

```
for eni_id in $(
  aws ec2 describe-vpc-endpoints \
   --filters Name=service-name,Values=service-id \
   --query 'VpcEndpoints[*].NetworkInterfaceIds' \
   --output text
 ); do
 aws ec2 describe-network-interfaces \
  --network-interface-ids $eni_id \
```

```
  --query 'NetworkInterfaces[*].PrivateIpAddresses[*].PrivateIpAddress' \
  --output text
      done
```

4. 使用 create-target-group 建立新的目標群組。您將使用此目標群組來註冊 Web 伺服器 Amazon VPC 端點的 IP 地址。

```
aws elbv2 create-target-group \
--name new-target-group-namne \
--protocol HTTPS \
--port 443 \
--vpc-id web-server-vpc-id \
--target-type ip \
--health-check-protocol HTTPS \
--health-check-port 443 \
--health-check-path / \
--health-check-enabled \
--matcher 'HttpCode="200,302"'
```

使用 register-targets命令註冊 IP 地址。

```
aws elbv2 register-targets \
--target-group-arn target-group-arn \
--targets Id=ip-address-1 Id=ip-address-2
```

5. 請求 ACM 憑證。如果您使用現有的憑證，請略過此步驟。

```
aws acm request-certificate \
--domain-name my-custom-domain.com \
--validation-method DNS
```

6. 設定一個 Application Load Balancer。首先，建立負載平衡器，然後建立負載平衡器的接聽程式。指定您在上一個步驟中建立的 ACM 憑證。

```
aws elbv2 create-load-balancer \
--name my-mwaa-lb \
--type application \
--subnets subnet-id-1 subnet-id-2
```

```
aws elbv2 create-listener \
--load-balancer-arn load-balancer-arn \
```

```
--protocol HTTPS \
--port 443 \
--ssl-policy ELBSecurityPolicy-2016-08 \
--certificates CertificateArn=acm-certificate-arn \
--default-actions Type=forward,TargetGroupArn=target-group-arn
```

如果您在私有子網路中使用 Network Load Balancer，請設定堡壘主機或Site-to-Site VPN 通道來存取 Web 伺服器。

7. 使用網域的 Route 53 建立託管區域。

```
aws route53 create-hosted-zone --name my-custom-domain.com \
--caller-reference 1
```

建立網域的 A 記錄。若要使用 執行此操作 AWS CLI，請使用 取得託管區域 ID，`list-hosted-zones-by-name`然後使用 套用記錄`change-resource-record-sets`。

```
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
--dns-name my-custom-domain.com \
--query 'HostedZones[0].Id' --output text)
```

```
aws route53 change-resource-record-sets \
--hosted-zone-id $HOSTED_ZONE_ID \
--change-batch '{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "my-custom-domain.com",
        "Type": "A",
        "AliasTarget": {
          "HostedZoneId": "load-balancer-hosted-zone-id>",
          "DNSName": "load-balancer-dns-name",
          "EvaluateTargetHealth": true
        }
      }
    }
  ]
}'
```

8. 更新 Web 伺服器 Amazon VPC 端點的安全群組規則，以遵循最低權限原則，僅允許來自 Application Load Balancer 所在公有子網路的 HTTPS 流量。在本機儲存下列 JSON。例如， 為 `sg-ingress-ip-permissions.json`。

```
[
  {
    "IpProtocol": "tcp",
    "FromPort": 443,
    "ToPort": 443,
    "UserIdGroupPairs": [
      {
        "GroupId": "load-balancer-security-group-id"
      }
    ],
    "IpRanges": [
      {
        "CidrIp": "public-subnet-1-cidr"
      },
      {
        "CidrIp": "public-subnet-2-cidr"
      }
    ]
  }
]
```

執行下列 Amazon EC2 命令來更新您的輸入安全群組規則。指定 的 JSON 檔案`--ip-permissions`。

```
aws ec2 authorize-security-group-ingress \
--group-id <security-group-id> \
--ip-permissions file://sg-ingress-ip-permissions.json
```

執行下列 Amazon EC2 命令來更新您的輸出規則。

```
aws ec2 authorize-security-group-egress \
--group-id webserver-vpc-endpoint-security-group-id \
--protocol tcp \
--port 443 \
--source-group load-balancer-security-group-id
```

開啟 Amazon MWAA 主控台並導覽至 Apache Airflow UI。如果您要在私有子網路中設定 Network Load Balancer，而不是此處使用的 Application Load Balancer，則必須使用下列其中一個選項存取 Web 伺服器。

- the section called “教學課程：Linux 堡壘主機”
- the section called “教學課程： AWS Client VPN”

建立 Apache Airflow CLI 字符

> **Tip**
>
> REST API 比 CLI 更現代化，旨在與外部系統進行程式設計整合。REST 是與 Apache Airflow 互動的偏好方式。

您可以使用此頁面上的命令來產生 CLI 字符，然後直接在命令 shell 中進行 Amazon Managed Workflows for Apache Airflow API 呼叫。例如，您可以取得權杖，然後使用 Amazon MWAA APIs 以程式設計方式部署 DAGs。下一節包含使用 AWS CLI、curl 指令碼、Python 指令碼或 bash 指令碼建立 Apache Airflow CLI 字符的步驟。回應中傳回的字符有效期為 60 秒。

AWS CLI 字符旨在取代同步 shell 動作，而不是非同步 API 命令。因此，可用的並行會受到限制。為了確保 Web 伺服器對使用者保持回應，我們建議不要在上一個 AWS CLI 請求成功完成之前開啟新請求。

內容

- 先決條件
  - 存取
  - AWS CLI
- 使用 AWS CLI
- 使用 curl 指令碼
- 使用 bash 指令碼
- 使用 Python 指令碼
- 後續步驟？

先決條件

下一節說明使用此頁面上的命令和指令碼所需的初步步驟。

存取

- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess。
- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策 完整 API 和主控台存取政策：AmazonMWAAFullApiAccess。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，可讓您在命令列 Shell 中使用命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

使用 AWS CLI

下列範例使用 中的 create-cli-token 命令 AWS CLI 來建立 Apache Airflow CLI 字符。

```
aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME
```

使用 curl 指令碼

下列範例使用 curl 指令碼呼叫 中的 create-web-login-token 命令 AWS CLI ，透過 Apache Airflow Webserver 上的端點叫用 Apache Airflow CLI。

Apache Airflow v3

1. 從您的文字檔案複製 curl 陳述式，並將其貼到您的命令 shell 中。

> **Note**
>
> 將其複製到剪貼簿後，您可能需要使用 shell 選單中的編輯 > 貼上。

```
CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
&& CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
&& WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
&& CLI_RESULTS=$(curl -L --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/
cli" \
--header "Authorization: Bearer $CLI_TOKEN" \
--header "Content-Type: text/plain" \
--data-raw "dags trigger YOUR_DAG_NAME --logical-date $(date -u +"%Y-%m-%dT%H:
%M:%SZ")") \
&& echo "Output:" \
&& echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
&& echo "Errors:" \
&& echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

2. 將紅色預留位置替換為您的 AWS 區域 環境、 *YOUR_DAG_NAME*和 *YOUR_ENVIRONMENT_NAME*。例如，公有網路的主機名稱類似 （ 不含 https：//）：

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

您的命令提示字元會顯示：

```
{
  "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
  "stdout":"<STDOUT of the CLI execution, base64 encoded>"
}
```

Apache Airflow v2

1. 從您的文字檔案複製 curl 陳述式，並將其貼到您的命令 shell 中。

> **Note**
>
> 將其複製到剪貼簿後，您可能需要使用 shell 選單中的編輯 > 貼上。

```
CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
```

```
&& CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
&& WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
&& CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/cli"
 \
--header "Authorization: Bearer $CLI_TOKEN" \
--header "Content-Type: text/plain" \
--data-raw "dags trigger YOUR_DAG_NAME") \
&& echo "Output:" \
&& echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
&& echo "Errors:" \
&& echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

2. 將紅色預留位置替換為您的 AWS 區域 環境、 YOUR_DAG_NAME和 YOUR_ENVIRONMENT_NAME。例如，公有網路的主機名稱類似 （ 不含 https：//）：

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

您的命令提示字元會顯示：

```
{
  "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
  "stdout":"<STDOUT of the CLI execution, base64 encoded>"
}
```

使用 bash 指令碼

下列範例使用 bash 指令碼呼叫 中的 create-cli-token 命令 AWS CLI 來建立 Apache Airflow CLI 權杖。

Apache Airflow v3

1. 複製下列程式碼範例的內容，並在本機儲存為 get-cli-token.sh。

```
# brew install jq
        aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export
 CLI_TOKEN=$(jq -r .CliToken) && curl -L --request POST "https://YOUR_HOST_NAME/
aws_mwaa/cli" \
        --header "Authorization: Bearer $CLI_TOKEN" \
        --header "Content-Type: text/plain" \
```

```
        --data-raw "dags trigger YOUR_DAG_NAME --logical-date $(date -u +"%Y-%m-
%dT%H:%M:%SZ")"
```

2. 以##取代 YOUR_ENVIRONMENT_NAME、 YOUR_HOST_NAME和 的預留位置YOUR_DAG_NAME。例如，公有網路的主機名稱類似 （ 不含 https：//）：

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. （選用） macOS 和 Linux 使用者可能需要執行下列命令，以確保指令碼可執行。

```
chmod +x get-cli-token.sh
```

4. 執行下列指令碼來建立 Apache Airflow CLI 字符。

```
./get-cli-token.sh
```

Apache Airflow v2

1. 複製下列程式碼範例的內容，並在本機儲存為 get-cli-token.sh。

```
# brew install jq
aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export CLI_TOKEN=$(jq -
r .CliToken) && curl --request POST "https://YOUR_HOST_NAME/aws_mwaa/cli" \
--header "Authorization: Bearer $CLI_TOKEN" \
--header "Content-Type: text/plain" \
--data-raw "dags trigger YOUR_DAG_NAME"
```

2. 以##取代 YOUR_ENVIRONMENT_NAME、 YOUR_HOST_NAME和 的預留位置YOUR_DAG_NAME。例如，公有網路的主機名稱類似 （ 不含 https：//）：

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. （選用） macOS 和 Linux 使用者可以執行下列命令，以確保指令碼可執行。

```
chmod +x get-cli-token.sh
```

4. 執行下列指令碼來建立 Apache Airflow CLI 字符。

```
./get-cli-token.sh
```

使用 Python 指令碼

下列範例使用 Python 指令碼中的 boto3 create_cli_token 方法建立 Apache Airflow CLI 字符並觸發 DAG。您可以在 Amazon MWAA 外部執行此指令碼。您只需要安裝 boto3 程式庫。您可能想要建立虛擬環境來安裝程式庫。其假設您已為 帳戶設定身分 AWS 驗證憑證。

Apache Airflow v3

1. 複製下列程式碼範例的內容，並在本機儲存為 `create-cli-token.py`。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
```

```
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. 取代 YOUR_ENVIRONMENT_NAME和 的預留位置YOUR_DAG_NAME。
3. 執行下列指令碼來建立 Apache Airflow CLI 字符。

```
python3 create-cli-token.py
```

Apache Airflow v2

1. 複製下列程式碼範例的內容，並在本機儲存為 create-cli-token.py。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
```

```
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. 取代 YOUR_ENVIRONMENT_NAME和 的預留位置YOUR_DAG_NAME。

3. 執行下列指令碼來建立 Apache Airflow CLI 字符。

```
python3 create-cli-token.py
```

後續步驟？

- 探索用於在 CreateCliToken 建立 CLI 權杖的 Amazon MWAA API 操作。

使用 Apache Airflow REST API

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) 支援針對執行 Apache Airflow v2.4.3 和更新版本的環境，直接使用 Apache Airflow REST API 與您的 Apache Airflow 環境互動。這可讓您以程式設計方式存取和管理 Amazon MWAA 環境，提供標準化的方式來叫用資料協同運作工作流程、管理您的 DAGs，以及監控各種 Apache Airflow 元件的狀態，例如中繼資料資料庫、觸發器和排程器。

為了在使用 Apache Airflow REST API 時支援可擴展性，Amazon MWAA 可讓您選擇水平擴展 Web 伺服器容量，以處理增加的需求，無論是來自 REST API 請求、命令列界面 (CLI) 用量，還是更多並行 Apache Airflow 使用者介面 (UI) 使用者。如需 Amazon MWAA 如何擴展 Web 伺服器的詳細資訊，請參閱 the section called “設定 Web 伺服器自動擴展”。

您可以使用 Apache Airflow REST API 為您的環境實作下列使用案例：

- 程式設計存取 – 您現在可以啟動 Apache Airflow DAG 執行、管理資料集，以及擷取各種元件的狀態，例如中繼資料資料庫、觸發器和排程器，而無需依賴 Apache Airflow UI 或 CLI。
- 與外部應用程式和微服務整合 – REST API 支援可用來建置自訂解決方案，將您的 Amazon MWAA 環境與其他系統整合。例如，您可以啟動工作流程以回應來自外部系統的事件，例如已完成的資料庫任務或新的使用者註冊。
- 集中式監控 – 您可以建置監控儀表板，彙總多個 Amazon MWAA 環境的 DAGs 狀態，實現集中式監控和管理。

如需 Apache Airflow REST API 的詳細資訊，請參閱 Apache Airflow REST API 參考。

透過使用 `InvokeRestApi`，您可以使用 AWS 登入資料存取 Apache Airflow REST API。或者，您也可以取得 Web 伺服器存取權杖來存取它，然後使用權杖來呼叫它。

如果您在使用 InvokeRestApi操作Update your environment to use InvokeRestApi時遇到訊息錯誤，表示您需要更新 Amazon MWAA 環境。當您的 Amazon MWAA 環境與InvokeRestApi功能相關的最新變更不相容時，就會發生此錯誤。若要解決此問題，請更新您的 Amazon MWAA 環境，以納入InvokeRestApi功能的必要變更。

InvokeRestApi 操作的預設逾時持續時間為 10 秒。如果操作未在此 10 秒內完成，則會自動終止，並引發錯誤。請確定您的 REST API 呼叫旨在在此逾時期間內完成，以避免遇到錯誤。

為了在使用 Apache Airflow REST API 時支援可擴展性，Amazon MWAA 可讓您選擇水平擴展 Web 伺服器容量，以處理增加的需求，無論是來自 REST API 請求、命令列界面 (CLI) 用量，還是更多並行 Apache Airflow 使用者介面 (UI) 使用者。如需 Amazon MWAA 如何擴展 Web 伺服器的詳細資訊，請參閱 the section called “設定 Web 伺服器自動擴展”。

您可以使用 Apache Airflow REST API 為您的環境實作下列使用案例：

- 程式設計存取 – 您現在可以啟動 Apache Airflow DAG 執行、管理資料集，以及擷取各種元件的狀態，例如中繼資料資料庫、觸發器和排程器，而無需依賴 Apache Airflow UI 或 CLI。
- 與外部應用程式和微服務整合 – REST API 支援可用來建置自訂解決方案，將您的 Amazon MWAA 環境與其他系統整合。例如，您可以啟動工作流程以回應來自外部系統的事件，例如已完成的資料庫任務或新的使用者註冊。
- 集中式監控 – 您可以建置監控儀表板，在多個 Amazon MWAA 環境中彙總 DAGs 的狀態，進而實現集中式監控和管理。

如需 Apache Airflow REST API 的詳細資訊，請參閱 Apache Airflow REST API 參考。

透過使用 InvokeRestApi，您可以使用 AWS 登入資料存取 Apache Airflow REST API。或者，您也可以取得 Web 伺服器存取權杖來存取它，然後使用權杖來呼叫它。

- 如果您在使用 InvokeRestApi操作Update your environment to use InvokeRestApi時遇到訊息錯誤，表示您需要更新 Amazon MWAA 環境。當您的 Amazon MWAA 環境與InvokeRestApi功能相關的最新變更不相容時，就會發生此錯誤。若要解決此問題，請更新您的 Amazon MWAA 環境，以納入InvokeRestApi功能的必要變更。
- InvokeRestApi 操作的預設逾時持續時間為 10 秒。如果操作未在此 10 秒內完成，則會自動終止，並引發錯誤。請確定您的 REST API 呼叫旨在在此逾時期間內完成，以避免遇到錯誤。

> **Important**
>
> 回應承載大小不能超過 6 MB。如果超過此限制，您的 `RestApi` 失敗。

使用下列範例對 Apache Airflow REST API 進行 API 呼叫，並啟動新的 DAG 執行：

主題

- 授予 Apache Airflow REST API 的存取權： airflow:InvokeRestApi
- 呼叫 Apache Airflow REST API
- 建立 Web 伺服器工作階段字符並呼叫 Apache Airflow REST API

授予 Apache Airflow REST API 的存取權： `airflow:InvokeRestApi`

若要使用 AWS 登入資料存取 Apache Airflow REST API，您必須在 IAM 政策中授予 `airflow:InvokeRestApi`許可。在下列政策範例中，在 中指定 `Admin`、`Viewer`、、 `Op` `User`或 `Public`角色，`{airflow-role}`以自訂使用者存取層級。如需詳細資訊，請參閱 Apache Airflow 參考指南中的預設角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> 設定私有 Web 伺服器時，無法從 Virtual Private Cloud (VPC) 外部叫用 `InvokeRestApi`動作。您可以使用 `aws:SourceVpc`金鑰，為此操作套用更精細的存取控制。如需詳細資訊，請參閱 aws：SourceVpc。

呼叫 Apache Airflow REST API

以下範例指令碼說明如何使用 Apache Airflow REST API 列出環境中可用的 DAGs，以及如何建立 Apache Airflow 變數：

```
import boto3

  env_name = "MyAirflowEnvironment"

  def list_dags(client):
    request_params = {
      "Name": env_name,
      "Path": "/dags",
      "Method": "GET",
      "QueryParameters": {
        "paused": False
      }
    }
  response = client.invoke_rest_api(
    **request_params
  )

  print("Airflow REST API response: ", response['RestApiResponse'])

  def create_variable(client):
    request_params = {
      "Name": env_name,
      "Path": "/variables",
      "Method": "POST",
      "Body": {
        "key": "test-restapi-key",
        "value": "test-restapi-value",
        "description": "Test variable created by MWAA InvokeRestApi API",
      }
    }
```

```
response = client.invoke_rest_api(
  **request_params
)

print("Airflow REST API response: ", response['RestApiResponse'])

if __name__ == "__main__":
  client = boto3.client("mwaa")
  list_dags(client)
  create_variable(client)
```

建立 Web 伺服器工作階段字符並呼叫 Apache Airflow REST API

若要建立 Web 伺服器存取字符，請使用下列 Python 函數。此函數會先呼叫 Amazon MWAA API 以取得 Web 登入字符。Web 登入字符會在 60 秒後過期，然後交換為 Web 工作階段字符，可讓您存取 Web 伺服器並使用 Apache Airflow REST API。如果您需要每秒超過 10 筆交易 (TPS) 的限流容量，您可以使用此方法存取 Apache Airflow REST API。

工作階段字符會在 12 小時後過期。

Tip

下列程式碼範例從 Apache Airflow v2 到 v3 的主要變更如下：

- REST API 路徑從 變更為 `/api/v1` `/api/v2`
- 登入路徑從 變更為 `/aws_maa/login` `/pluginsv2/aws_mwaa/login`
- 來自登入的回應`response.cookies["_token"]`包含您後續 API 呼叫必須使用的字符資訊
- 對於 REST API 呼叫，您必須在標頭中傳遞`jwt_token`資訊，如下所示：

```
headers = {
  "Authorization": f"Bearer {jwt_token}",
  "Content-Type": "application/json"
}
```

Apache Airflow v3

```
def get_token_info(region, env_name):
  logging.basicConfig(level=logging.INFO)
```

```
try:
  # Initialize MWAA client and request a web login token
  mwaa = boto3.client('mwaa', region_name=region)
  response = mwaa.create_web_login_token(Name=env_name)

  # Extract the web server hostname and login token
  web_server_host_name = response["WebServerHostname"]
  web_token = response["WebToken"]

  # Construct the URL needed for authentication
  login_url = f"https://{web_server_host_name}/pluginsv2/aws_mwaa/login"
  login_payload = {"token": web_token}

  # Make a POST request to the MWAA login url using the login payload
  response = requests.post(
    login_url,
    data=login_payload,
    timeout=10
  )

  # Check if login was successful
  if response.status_code == 200:

  # Return the hostname and the session cookie
  return (
    web_server_host_name,
    response.cookies['_token']
  )
  else:
    # Log an error
    logging.error("Failed to log in: HTTP %d", response.status_code)
    return None
    except requests.RequestException as e:

    # Log any exceptions raised during the request to the MWAA login endpoint
    logging.error("Request failed: %s", str(e))
    return None
    except Exception as e:

    # Log any other unexpected exceptions
    logging.error("An unexpected error occurred: %s", str(e))
    return None
```

Apache Airflow v2

```
def get_session_info(region, env_name):
  logging.basicConfig(level=logging.INFO)

  try:
      # Initialize MWAA client and request a web login token
      mwaa = boto3.client('mwaa', region_name=region)
      response = mwaa.create_web_login_token(Name=env_name)

      # Extract the web server hostname and login token
      web_server_host_name = response["WebServerHostname"]
      web_token = response["WebToken"]

      # Construct the URL needed for authentication
      login_url = f"https://{web_server_host_name}/aws_mwaa/login"
      login_payload = {"token": web_token}

      # Make a POST request to the MWAA login url using the login payload
      response = requests.post(
          login_url,
          data=login_payload,
          timeout=10
      )

      # Check if login was succesfull
      if response.status_code == 200:

          # Return the hostname and the session cookie
          return (
              web_server_host_name,
              response.cookies["session"]
          )
      else:
          # Log an error
          logging.error("Failed to log in: HTTP %d", response.status_code)
          return None
  except requests.RequestException as e:
       # Log any exceptions raised during the request to the MWAA login endpoint
      logging.error("Request failed: %s", str(e))
      return None
  except Exception as e:
      # Log any other unexpected exceptions
      logging.error("An unexpected error occurred: %s", str(e))
```

```
    return None
```

身分驗證完成後，您有登入資料可開始傳送請求至 API 端點。在下一節的範例中，使用端點 dags/{dag_name}/dagRuns。

Apache Airflow v3

```
def trigger_dag(region, env_name, dag_id):
    """
    Triggers a DAG in a specified MWAA environment using the Airflow REST API.

    Args:
    region (str): AWS region where the MWAA environment is hosted.
    env_name (str): Name of the MWAA environment.
    dag_id (str): ID of the DAG to trigger.
    """

    logging.info(f"Attempting to trigger DAG {dag_id} in environment {env_name} at
 region {region}")

    # Retrieve the web server hostname and token for authentication
    try:
    web_server_host_name, jwt_token = get_token_info(region, env_name)
    if not jwt_token:
    logging.error("Authentication failed, no jwt token retrieved.")
    return
    except Exception as e:
    logging.error(f"Error retrieving token info: {str(e)}")
    return

    # Prepare headers and payload for the request
    request_headers = {
    "Authorization": f"Bearer {jwt_token}",
    "Content-Type": "application/json" # Good practice to include, even for GET
    }

    # sample request body input
    json_body = {"logical_date": "2025-09-17T14:15:00Z"}

    # Construct the URL for triggering the DAG
    url = f"https://{web_server_host_name}/api/v2/dags/{dag_id}/dagRuns"
```

```
    # Send the POST request to trigger the DAG
    try:
    response = requests.post(url, headers=request_headers, json=json_body)
    # Check the response status code to determine if the DAG was triggered
successfully
    if response.status_code == 200:
    logging.info("DAG triggered successfully.")
    else:
    logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
{response.text}")
    except requests.RequestException as e:
    logging.error(f"Request to trigger DAG failed: {str(e)}")

    if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 4:
    logging.error("Incorrect usage. Proper format: python script_name.py {region}
{env_name} {dag_id}")
    sys.exit(1)

    region = sys.argv[1]
    env_name = sys.argv[2]
    dag_id = sys.argv[3]

    # Trigger the DAG with the provided arguments
    trigger_dag(region, env_name, dag_id)
```

Apache Airflow v2

```
def trigger_dag(region, env_name, dag_name):
    """
    Triggers a DAG in a specified MWAA environment using the Airflow REST API.

    Args:
    region (str): AWS region where the MWAA environment is hosted.
    env_name (str): Name of the MWAA environment.
    dag_name (str): Name of the DAG to trigger.
    """

    logging.info(f"Attempting to trigger DAG {dag_name} in environment {env_name}
at region {region}")
```

```
    # Retrieve the web server hostname and session cookie for authentication
    try:
    web_server_host_name, session_cookie = get_session_info(region, env_name)
    if not session_cookie:
    logging.error("Authentication failed, no session cookie retrieved.")
    return
    except Exception as e:
    logging.error(f"Error retrieving session info: {str(e)}")
    return

    # Prepare headers and payload for the request
    cookies = {"session": session_cookie}
    json_body = {"conf": {}}

    # Construct the URL for triggering the DAG
    url = f"https://{web_server_host_name}/api/v1/dags/{dag_id}/dagRuns"

    # Send the POST request to trigger the DAG
    try:
    response = requests.post(url, cookies=cookies, json=json_body)
    # Check the response status code to determine if the DAG was triggered
successfully
    if response.status_code == 200:
    logging.info("DAG triggered successfully.")
    else:
    logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
{response.text}")
    except requests.RequestException as e:
    logging.error(f"Request to trigger DAG failed: {str(e)}")

    if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 4:
    logging.error("Incorrect usage. Proper format: python script_name.py {region}
{env_name} {dag_name}")
    sys.exit(1)

    region = sys.argv[1]
    env_name = sys.argv[2]
    dag_name = sys.argv[3]
```

```
    # Trigger the DAG with the provided arguments
    trigger_dag(region, env_name, dag_name)
```

Apache Airflow CLI 命令參考

本主題說明 Amazon Managed Workflows for Apache Airflow 上支援和不支援的 Apache Airflow CLI 命令。

> **Tip**
>
> REST API 比 CLI 更現代化，旨在與外部系統進行程式設計整合。REST 是與 Apache Airflow 互動的偏好方式。

內容

- 先決條件
  - 存取
  - AWS CLI
- 有何變更？
- 支援的 CLI 命令
  - 支援的命令
  - 使用剖析 DAGs命令
- 範本程式碼
  - 設定、取得或刪除 Apache Airflow v2 變數
  - 觸發 DAG 時新增組態
  - 在對堡壘主機的 SSH 通道上執行 CLI 命令

先決條件

下一節說明使用此頁面上命令和指令碼所需的初步步驟。

存取

- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess。
- AWS 帳戶 在 AWS Identity and Access Management (IAM) 中存取 Amazon MWAA 許可政策 完整 API 和主控台存取政策：AmazonMWAAFullApiAccess。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

有何變更？

- v3：氣流架構。Apache Airflow v3 引進突破性的架構變更，以提供更高的安全性和可擴展性，並使維護更容易。若要進一步了解，請參閱升級至氣流 3。
- v2：Airflow CLI 命令結構。Apache Airflow v2 CLI 經過組織，因此相關命令會分組為子命令，這表示如果您想要升級至 Apache Airflow v2，則需要更新 Apache Airflow v1 指令碼。例如，`unpause`在 Apache Airflow v1 `dags unpause`中為 Apache Airflow v2。若要進一步了解，請參閱 2.0 中的 Airflow CLI 變更。

支援的 CLI 命令

下一節列出 Amazon MWAA 上可用的 Apache Airflow CLI 命令。

支援的命令

Apache Airflow v3

次要版本	命令
3.0.6 版	資產詳細資訊

次要版本	命令
3.0.6 版	資產清單
3.0.6 版	資產具體化
3.0.6 版	回填建立
3.0.6 版	cheat-sheet
3.0.6 版	連線新增
3.0.6 版	連線刪除
3.0.6 版	dags 刪除
3.0.6 版	dags 清單
3.0.6 版	dags list-jobs
3.0.6 版	dags list-import-errors
3.0.6 版	dags list-runs
3.0.6 版	dags next-execution
3.0.6 版	dags 暫停
3.0.6 版	dags 報告
3.0.6 版	dags 重新序列化

次要版本	命令
3.0.6 版	dags show
3.0.6 版	dags 狀態
3.0.6 版	dags 測試
3.0.6 版	dags 觸發條件
3.0.6 版	dags unpause
3.0.6 版	db 清除
3.0.6 版	供應商行為
3.0.6 版	供應商取得
3.0.6 版	提供者勾點
3.0.6 版	供應商連結
3.0.6 版	供應商清單
3.0.6 版	供應商通知
3.0.6 版	提供者秘密
3.0.6 版	提供者觸發器

次要版本	命令
3.0.6 版	提供者小工具
3.0.6 版	角色 add-perms
3.0.6 版	角色 del-perms
3.0.6 版	角色建立
3.0.6 版	角色清單
3.0.6 版	任務清除
3.0.6 版	任務故障裝置
3.0.6 版	任務清單
3.0.6 版	任務轉譯
3.0.6 版	任務狀態
3.0.6 版	任務 states-for-dag-run
3.0.6 版	任務測試
3.0.6 版	變數刪除
3.0.6 版	變數取得

次要版本	命令
3.0.6 版	變數集
3.0.6 版	變數清單
3.0.6 版	version

Apache Airflow v2

次要版本	命令
v2.0+	cheat-sheet
v2.0+	連線新增
v2.0+	連線刪除
v2.2+ (備註)	dags 回填
v2.0+	dags 刪除
v2.2+ (備註)	dags 清單
v2.0+	dags list-jobs
v2.6+	dags list-import-errors
v2.2+ (備註)	dags list-runs
v2.2+ (備註)	dags next-execution

次要版本	命令
v2.0+	dags 暫停
v2.0+	dags 報告
v2.4+	dags 重新序列化
v2.0+	dags show
v2.0+	dags 狀態
v2.0+	dags 測試
v2.0+	dags 觸發條件
v2.0+	dags 取消暫停
v2.4+	db 清除
v2.0+	提供者行為
v2.0+	供應商取得
v2.0+	提供者勾點
v2.0+	供應商連結
v2.0+	提供者清單

次要版本	命令
v2.8+	供應商通知
v2.6+	提供者秘密
v2.7+	提供者觸發器
v2.0+	提供者小工具
v2.6+	角色 add-perms
v2.6+	角色 del-perms
v2.6+	角色建立
v2.0+	角色清單
v2.0+	任務清除
v2.0+	任務故障裝置
v2.0+	任務清單
v2.0+	任務轉譯
v2.0+	任務執行
v2.0+	任務狀態

次要版本	命令
v2.0+	任務 states-for-dag-run
v2.0+	任務測試
v2.0+	變數刪除
v2.0+	變數取得
v2.0+	變數集
v2.0+	變數清單
v2.0+	version

使用剖析 DAGs命令

如果您的環境執行 Apache Airflow v2.0.2，如果 DAGs 使用依賴透過 安裝的套件的外掛程式，則剖析 DAG 的 CLI 命令將會失敗`requirements.txt`：

Apache Airflow 2.0.2 版

- `dags backfill`
- `dags list`
- `dags list-runs`
- `dags next-execution`

如果您的 DAGs 不使用依賴透過 安裝的套件的外掛程式，您可以使用這些 CLI 命令`requirements.txt`。

範本程式碼

下一節包含使用 Apache Airflow CLI 的不同方法範例。

設定、取得或刪除 Apache Airflow v2 變數

您可以使用下列範例程式碼來設定、取得或刪除格式為 的變數<script> <mwaa env name> get | set | delete <variable> <variable value> </variable> </variable>。

```
[ $# -eq 0 ] && echo "Usage: $0 MWAA environment name " && exit

if [[ $2 == "" ]]; then
    dag="variables list"

elif  [ $2 == "get" ] ||  [ $2 == "delete" ] ||  [ $2 == "set" ]; then
    dag="variables $2 $3 $4 $5"

else
    echo "Not a valid command"
    exit 1
fi

CLI_JSON=$(aws mwaa --region $AWS_REGION create-cli-token --name $1) \
    && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
    && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
    && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/cli" \
    --header "Authorization: Bearer $CLI_TOKEN" \
    --header "Content-Type: text/plain" \
    --data-raw "$dag" ) \
    && echo "Output:" \
    && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
    && echo "Errors:" \
    && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

觸發 DAG 時新增組態

您可以在觸發 DAG 時使用下列範例程式碼搭配 Apache Airflow v2 來新增組態，例如 airflow trigger_dag 'dag_name' —conf '{"key":"value"}'。

```
import boto3
import json
import requests
import base64

  mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
  dag_name = 'YOUR_DAG_NAME'
  key = "YOUR_KEY"
```

```
  value = "YOUR_VALUE"
  conf = "{\"" + key + "\":\"" + value + "\"}"

  client = boto3.client('mwaa')

  mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
  )

  mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
  mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
  raw_data = "trigger_dag {0} -c '{1}'".format(dag_name, conf)

  mwaa_response = requests.post(
    mwaa_webserver_hostname,
    headers={
      'Authorization': mwaa_auth_token,
      'Content-Type': 'text/plain'
    },
    data=raw_data
  )

  mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
  mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

  print(mwaa_response.status_code)
  print(mwaa_std_err_message)
  print(mwaa_std_out_message)
```

在對堡壘主機的 SSH 通道上執行 CLI 命令

使用以下範例，使用 SSH 通道代理對 Linux 堡壘主機執行 Airflow CLI 命令。

使用 curl

1. ```
   ssh -D 8080 -f -C -q -N YOUR_USER@YOUR_BASTION_HOST
   ```
2. ```
   curl -x socks5h://0:8080 --request POST https://YOUR_HOST_NAME/aws_mwaa/cli --
   header YOUR_HEADERS --data-raw YOUR_CLI_COMMAND
   ```

管理 Apache Airflow 的連線

本章說明如何為 Amazon Managed Workflows for Apache Airflow 環境設定 Apache Airflow 連線。

主題

- Apache Airflow 變數和連線概觀
- 安裝在 Amazon MWAA 環境上的 Apache Airflow 提供者套件
- 連線類型概觀
- 使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線

Apache Airflow 變數和連線概觀

在某些情況下，您可能想要指定環境的其他連線或變數，例如 AWS 設定檔，或在 Apache Airflow 中繼存放區中的連線物件中新增執行角色，然後參考來自 DAG 內的連線。

- 自我管理的 Apache Airflow。在自我管理的 Apache Airflow 安裝上，您可以在 中設定 Apache Airflow 組態選項`airflow.cfg`。

```
[secrets]
backend = airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
backend_kwargs = {"connections_prefix" : "airflow/connections", "variables_prefix" :
 "airflow/variables"}
```

- Amazon MWAA 上的 Apache Airflow。在 Amazon MWAA 上，您需要將這些組態設定新增為 Amazon MWAA 主控台上的 Apache Airflow 組態選項。Apache Airflow 組態選項會寫入環境變數至您的環境，並覆寫相同設定的所有其他現有組態。

安裝在 Amazon MWAA 環境上的 Apache Airflow 提供者套件

此頁面列出 Amazon MWAA 為所有支援的 Apache Airflow 環境安裝的 Apache Airflow 提供者套件。如需這些套件的詳細資訊，請參閱套件額外項目的 Apache Airflow 參考。

> **Note**
>
> 為了確保其他 Python 程式庫安裝不會覆寫與 CloudWatch 記錄的相容性，Amazon MWAA 會在執行 之後安裝 Watchtower 2.0.1 版。 `pip3 install -r requirements.txt`

主題

- 限制條件檔案
- 特定版本提供者套件

限制條件檔案

從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制，Amazon MWAA 會為您指定一個，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。

Apache Airflow 限制條件檔案會指定 Apache Airflow 發行時可用的提供者版本。不過，在許多情況下，較新的供應商與該版本的 Apache Airflow 相容。由於您必須使用限制條件，若要指定較新版本的提供者套件，您可以修改特定提供者版本的限制條件檔案：

1. 從 GitHub 下載版本特定的限制條件檔案，例如 https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt （將 '2.7.2' 取代為您要使用的版本）。
2. 將修改後的限制檔案儲存至 Amazon MWAA 環境的 Amazon S3 dags 資料夾，例如 `constraints-3.11-updated.txt`。
3. 指定您的需求，如下所列。

```
--constraint "/usr/local/airflow/dags/constraints-3.11-updated.txt"
apache-airflow-providers-amazon==version-number
```

> **Note**
>
> 如果您使用的是私有 Web 伺服器，建議您使用 aws-mwaa-docker-images 將所需的程式庫封裝為 WHL 檔案。

特定版本提供者套件

安裝提供者套件，您可以使用 存取 Apache Airflow UI 中的連線類型。這也表示您不需要將這些套件指定為`requirements.txt`檔案中的 Python 相依性。此頁面列出 Amazon MWAA 為所有支援的 Apache Airflow 環境安裝的 Apache Airflow 提供者套件。

> **Note**
>
> 對於 Apache Airflow v2 和更新版本，Amazon MWAA 在執行 之後安裝 Watchtower 2.0.1 版`pip3 install -r requirements.txt`，以確保其他 Python 程式庫安裝不會覆寫與 CloudWatch 記錄的相容性。

您可以指定 的最新支援版本`apache-airflow-providers-amazon`，以升級此供應商。

支援的 Apache Airflow 版本：

v3.0.6

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】==9.9.0
Postgres 連線	apache-airflow-providers-postgres==6.2.1
FTP 連線	apache-airflow-providers-ftp==3.13.1
Fab 連線	apache-airflow-providers-fab==2.3.0
Celery 連線	apache-airflow-providers-celery==3.12.1
HTTP 連線	apache-airflow-providers-http=5.3.2
IMAP 連線	apache-airflow-providers-imap==3.9.1
常見 SQL	apache-airflow-providers-common-sql=1.27.3
SQLite 連線	apache-airflow-providers-sqlite==4.1.1

v2.11.0

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】==9.8.0
Postgres 連線	apache-airflow-providers-postgres==6.2.0
FTP 連線	apache-airflow-providers-ftp=3.13.0
Fab 連線	apache-airflow-providers-fab==1.5.3
Celery 連線	apache-airflow-providers-celery==3.11.0
HTTP 連線	apache-airflow-providers-http=5.3.0
IMAP 連線	apache-airflow-providers-imap==3.9.0
常見 SQL	apache-airflow-providers-common-sql==1.27.1
SQLite 連線	apache-airflow-providers-sqlite==4.1.0
SMTP 連線	apache-airflow-providers-smtp==2.1.0

v2.10.3

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】==9.0.0

連線類型	套件
Postgres 連線	apache-airflow-providers-postgres=5.13.1
FTP 連線	apache-airflow-providers-ftp==3.11.1
Fab 連線	apache-airflow-providers-fab==1.5.0
Celery 連線	apache-airflow-providers-celery==3.8.3
HTTP 連線	apache-airflow-providers-http=4.13.2
IMAP 連線	apache-airflow-providers-imap=3.7.0
常見 SQL	apache-airflow-providers-common-sql=1.19.0
SQLite 連線	apache-airflow-providers-sqlite==3.9.0
SMTP 連線	apache-airflow-providers-smtp==1.8.0

v2.10.1

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】=8.28.0
Postgres 連線	apache-airflow-providers-postgres==5.12.0
FTP 連線	apache-airflow-providers-ftp==3.11.0

連線類型	套件
Fab 連線	apache-airflow-providers-fab==1.3.0
Celery 連線	apache-airflow-providers-celery==3.8.1
HTTP 連線	apache-airflow-providers-http=4.13.0
IMAP 連線	apache-airflow-providers-imap==3.7.0
常見 SQL	apache-airflow-providers-common-sql=1.16.0
SQLite 連線	apache-airflow-providers-sqlite==3.9.0
SMTP 連線	apache-airflow-providers-smtp==1.8.0

v2.9.2

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】=8.24.0
Postgres 連線	apache-airflow-providers-postgres==5.11.1
FTP 連線	apache-airflow-providers-ftp==3.9.1
Fab 連線	apache-airflow-providers-fab==1.1.1
Celery 連線	apache-airflow-providers-celery==3.7.2
HTTP 連線	apache-airflow-providers-http=4.11.1

連線類型	套件
IMAP 連線	apache-airflow-providers-imap==3.6.1
常見 SQL	apache-airflow-providers-common-sql=1.14.0
SQLite 連線	apache-airflow-providers-sqlite==3.8.1
SMTP 連線	apache-airflow-providers-smtp==1.7.1

v2.8.1

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】=8.16.0
Postgres 連線	apache-airflow-providers-postgres=5.10.0
FTP 連線	apache-airflow-providers-ftp==3.7.0
Celery 連線	apache-airflow-providers-celery==3.5.1
HTTP 連線	apache-airflow-providers-http=4.8.0
IMAP 連線	apache-airflow-providers-imap=3.5.0
常見 SQL	apache-airflow-providers-common-sql=1.10.0
SQLite 連線	apache-airflow-providers-sqlite==3.7.0

v2.7.2

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】=8.7.1
Postgres 連線	apache-airflow-providers-postgres==5.6.1
FTP 連線	apache-airflow-providers-ftp==3.5.2
Celery 連線	apache-airflow-providers-celery==3.3.4
HTTP 連線	apache-airflow-providers-http=4.5.2
IMAP 連線	apache-airflow-providers-imap==3.3.2
常見 SQL	apache-airflow-providers-common-sql=1.7.2
SQLite 連線	apache-airflow-providers-sqlite==3.4.3

v2.6.3

連線類型	套件
AWS 連線	apache-airflow-providers-amazon【aiobotocore】=8.2.0
Postgres 連線	apache-airflow-providers-postgres==5.5.1
FTP 連線	apache-airflow-providers-ftp==3.4.2

連線類型	套件
Celery 連線	apache-airflow-providers-celery==3.2.1
HTTP 連線	apache-airflow-providers-http=4.4.2
IMAP 連線	apache-airflow-providers-imap==3.2.2
常見 SQL	apache-airflow-providers-common-sql=1.5.2
SQLite 連線	apache-airflow-providers-sqlite==3.4.2

v2.5.1

連線類型	套件
AWS 連線	apache-airflow-providers-amazon==7.1.0
Postgres 連線	apache-airflow-providers-postgres=5.4.0
FTP 連線	apache-airflow-providers-ftp==3.3.0
Celery 連線	apache-airflow-providers-celery==3.1.0
HTTP 連線	apache-airflow-providers-http=4.1.1
IMAP 連線	apache-airflow-providers-imap==3.1.1
常見 SQL	apache-airflow-providers-common-sql=1.3.3
SQLite 連線	apache-airflow-providers-sqlite==3.3.1

v2.4.3

連線類型	套件
AWS 連線	apache-airflow-providers-amazon==6.0.0
Postgres 連線	apache-airflow-providers-postgres==5.2.2
FTP 連線	apache-airflow-providers-ftp==3.1.0
Celery 連線	apache-airflow-providers-celery==3.0.0
HTTP 連線	apache-airflow-providers-http=4.0.0
IMAP 連線	apache-airflow-providers-imap==3.0.0
常見 SQL	apache-airflow-providers-common-sql=1.2.0
SQLite 連線	apache-airflow-providers-sqlite==3.2.1

連線類型概觀

Apache Airflow 將連線儲存為連線 URI 字串。它在 Apache Airflow UI 中提供連線範本來產生連線 URI 字串，無論連線類型為何。如果 Apache Airflow UI 中無法使用連線範本，則可以使用替代連線範本來產生此連線 URI 字串，例如使用 HTTP 連線範本。主要差異是 URI 字首，例如 `my-conn-type://`，Apache Airflow 供應商通常會忽略此字首的連線。此頁面說明如何針對不同的連線類型交替使用 Apache Airflow UI 中的連線範本。

> **Warning**
>
> 請勿覆寫 Amazon MWAA 中的`aws_default`連線。Amazon MWAA 使用此連線來執行各種關鍵任務，例如收集任務日誌。覆寫此連線可能會導致資料遺失和環境可用性中斷。

主題

- 連線 URI 字串範例
- 連線範本範例
- 使用 HTTP 連線範本進行 Jdbc 連線的範例

連線 URI 字串範例

下列範例顯示 MySQL 連線類型的連線 URI 字串。

```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role%2FAmazonMWAA-
MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

連線範本範例

下列範例說明 Apache Airflow UI 中的 HTTP 連線範本。

Apache Airflow v3

Apache Airflow v2

使用 HTTP 連線範本進行 Jdbc 連線的範例

使用下列範例，在 Apache Airflow UI 中套用 Jdbc 連線類型的 HTTP 連線範本。

Apache Airflow v3

下列範例顯示 Apache Airflow 針對本節中的範例產生的連線 URI 字串。

```
http://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

使用下列範例，在 Apache Airflow UI 中為 Apache Airflow v3 的 Jdbc 連線套用 HTTP 連線範本。

Apache Airflow v2

下列範例顯示 Apache Airflow 針對本節中的範例產生的連線 URI 字串。

```
http://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

使用下列範例，在 Apache Airflow UI 中為 Apache Airflow v2 的 Jdbc 連線套用 HTTP 連線範本。

使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線

AWS Secrets Manager 是 Amazon Managed Workflows for Apache Airflow 環境上支援的替代 Apache Airflow 後端。本主題說明如何使用 在 Amazon Managed Workflows for Apache Airflow 上 AWS Secrets Manager 安全地存放 Apache Airflow 變數和 Apache Airflow 連線的秘密。

> **Note**
>
> - 您需要為您建立的秘密付費。如需 Secrets Manager 定價的詳細資訊，請參閱 AWS 定價。
> - AWS Systems Manager 參數存放區也支援做為 Amazon MWAA 中的秘密後端。如需詳細資訊，請參閱 Amazon Provider Package 文件。

內容

- 步驟一：提供 Amazon MWAA 存取 Secrets Manager 私密金鑰的許可
- 步驟二：建立 Secrets Manager 後端做為 Apache Airflow 組態選項
- 步驟三：產生 Apache Airflow AWS 連線 URI 字串
- 步驟四：在 Secrets Manager 中新增變數
- 步驟五：在 Secrets Manager 中新增連線
- 範本程式碼
- Resources
- 後續步驟？

步驟一：提供 Amazon MWAA 存取 Secrets Manager 私密金鑰的許可

Amazon MWAA 環境的執行角色需要秘密金鑰的讀取存取權 AWS Secrets Manager。下列 IAM 政策允許使用 AWS受管 SecretsManagerReadWrite 政策進行讀寫存取。

將政策連接至您的執行角色

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在許可窗格中選擇您的執行角色。
4. 選擇連接政策。
5. `SecretsManagerReadWrite` 在篩選政策文字欄位中輸入 。
6. 選擇連接政策。

如果您不想使用 AWS受管許可政策，您可以直接更新環境的執行角色，以允許任何層級的 Secrets Manager 資源存取權。例如，下列政策陳述式會授予您在 Secrets Manager 中特定 AWS 區域 中建立的所有秘密的讀取存取權。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:111122223333:secret:*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        }
    ]
```

```
}
```

步驟二：建立 Secrets Manager 後端做為 Apache Airflow 組態選項

下一節說明如何在 AWS Secrets Manager 後端的 Amazon MWAA 主控台上建立 Apache Airflow 組態選項。如果您在 中使用相同名稱的組態設定`airflow.cfg`，則在下列步驟中建立的組態會優先並覆寫組態設定。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。
5. 在 Airflow 組態選項窗格中選擇新增自訂組態。 新增下列鍵/值對：

   a. **`secrets.backend`**: **`airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBacken`**

   b. **`secrets.backend_kwargs`**：**`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`**這會設定 Apache Airflow 在 和 `airflow/variables/*` 路徑搜尋連線字串`airflow/connections/*`和變數。

      您可以使用查詢模式來減少 Amazon MWAA 代表您對 Secrets Manager 進行的 API 呼叫次數。如果您未指定查詢模式，Apache Airflow 會搜尋所設定後端中的所有連線和變數。透過指定模式，您可以縮小 Apache Airflow 搜尋的可能路徑。這可降低將 Secrets Manager 與 Amazon MWAA 搭配使用時的成本。

      若要指定查詢模式，請指定 `connections_lookup_pattern`和 `variables_lookup_pattern` 參數。這些參數接受 RegEx 字串做為輸入。例如，若要搜尋開頭為 的秘密`test`，請在 中輸入下列內容`secrets.backend_kwargs`：

      ```
      {
        "connections_prefix": "airflow/connections",
        "connections_lookup_pattern": "^test",
        "variables_prefix" : "airflow/variables",
        "variables_lookup_pattern": "^test"
      }
      ```

> **Note**
>
> 若要使用 connections_lookup_pattern和 variables_lookup_pattern，您必須安裝 7apache-airflow-providers-amazon.3.0 版或更新版本。如需將的 provder pacakge 更新至較新版本的詳細資訊，請參閱 限制條件檔案。

6. 選擇儲存。

步驟三：產生 Apache Airflow AWS 連線 URI 字串

若要建立連線字串，請使用鍵盤上的「tab」鍵來縮排連線物件中的鍵值對。我們也建議您在 shell 工作階段中為extra物件建立變數。下一節會逐步解說使用 Apache Airflow 或 Python 指令碼為 Amazon MWAA 環境產生 Apache Airflow 連線 URI 字串的步驟。

Apache Airflow CLI

下列 shell 工作階段使用本機 Airflow CLI 來產生連線字串。如果您沒有安裝 CLI，建議您使用 Python 指令碼。

1. 開啟 Python shell 工作階段：

```
python3
```

2. 輸入以下命令：

```
>>> import json
```

3. 輸入以下命令：

```
>>> from airflow.models.connection import Connection
```

4. 在extra物件的 shell 工作階段中建立變數。將 *YOUR_EXECUTION_ROLE_ARN* 中的範例值替換為執行角色 ARN，以及 *us-east-1* 中的區域（例如 us-east-1)。

```
>>> extra=json.dumps({'role_arn': 'YOUR_EXECUTION_ROLE_ARN', 'region_name': 'us-east-1'})
```

5. 建立連線物件。以 Apache Airflow 連線myconn的名稱取代 中的範例值。

```
>>> myconn = Connection(
```

6. 使用鍵盤上的「tab」鍵來縮排連線物件中的下列每個鍵值對。以##取代範例值。

   a. 指定 AWS 連線類型：

   ```
   ... conn_id='aws',
   ```

   b. 指定 Apache Airflow 資料庫選項：

   ```
   ... conn_type='mysql',
   ```

   c. 在 Amazon MWAA 上指定 Apache Airflow UI URL：

   ```
   ... host='288888a0-50a0-888-9a88-1a111aaa0000.a1.us-
   east-1.airflow.amazonaws.com/home',
   ```

   d. 指定要登入 Amazon MWAA 的 AWS 存取金鑰 ID （使用者名稱）：

   ```
   ... login='YOUR_AWS_ACCESS_KEY_ID',
   ```

   e. 指定要登入 Amazon MWAA 的 AWS 私密存取金鑰 （密碼）：

   ```
   ... password='YOUR_AWS_SECRET_ACCESS_KEY',
   ```

   f. 指定 `extra` shell 工作階段變數：

   ```
   ... extra=extra
   ```

   g. 關閉連線物件。

   ```
   ... )
   ```

7. 列印連線 URI 字串：

   ```
   >>> myconn.get_uri()
   ```

   請參閱回應中的連線 URI 字串：

```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
%2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

Python script

下列 Python 指令碼不需要 Apache Airflow CLI。

1. 複製下列程式碼範例的內容，並在本機儲存為 mwaa_connection.py。

```
import urllib.parse

        conn_type = 'YOUR_DB_OPTION'
        host = 'YOUR_MWAA_AIRFLOW_UI_URL'
        port = 'YOUR_PORT'
        login = 'YOUR_AWS_ACCESS_KEY_ID'
        password = 'YOUR_AWS_SECRET_ACCESS_KEY'
        role_arn = urllib.parse.quote_plus('YOUR_EXECUTION_ROLE_ARN')
        region_name = 'us-east-1'

        conn_string = '{0}://{1}:{2}@{3}:{4}?
role_arn={5}&region_name={6}'.format(conn_type, login, password, host, port,
 role_arn, region_name)
        print(conn_string)
```

2. 以##取代預留位置。
3. 執行下列指令碼來產生連線字串。

```
python3 mwaa_connection.py
```

步驟四：在 Secrets Manager 中新增變數

下一節說明如何在 Secrets Manager 中建立變數的秘密。

建立秘密

1. 開啟 AWS Secrets Manager 主控台。
2. 選擇儲存新機密。

3. 選擇其他類型的秘密。
4. 在指定要存放在此秘密窗格中的金鑰/值對上，選擇純文字。
5. 以下列格式將變數值新增為純文字。

```
"YOUR_VARIABLE_VALUE"
```

例如，若要指定整數：

```
14
```

例如，若要指定字串：

```
"mystring"
```

6. 對於加密金鑰，從下拉式清單中選擇 AWS KMS 金鑰選項。
7. 在秘密名稱的文字欄位中輸入名稱，格式如下。

```
airflow/variables/YOUR_VARIABLE_NAME
```

例如：

```
airflow/variables/test-variable
```

8. 選擇下一步。
9. 在設定秘密頁面上的秘密名稱和描述窗格中，執行下列動作。
   a. 針對秘密名稱，請提供秘密的名稱。
   b. （選用）針對描述，提供秘密的描述。

   選擇下一步。
10. 在設定輪換 - 選用中保留預設選項，然後選擇下一步。
11. 針對您要新增的任何其他變數，在 Secrets Manager 中重複這些步驟。
12. 在檢閱頁面上，檢閱您的秘密，然後選擇儲存。

步驟五：在 Secrets Manager 中新增連線

下一節說明如何在 Secrets Manager 中建立連線字串 URI 的秘密。

建立秘密

1. 開啟 AWS Secrets Manager 主控台。
2. 選擇儲存新機密。
3. 選擇其他類型的秘密。
4. 在指定要存放在此秘密窗格中的金鑰/值對上，選擇純文字。
5. 以下列格式將連線 URI 字串新增為純文字。

```
YOUR_CONNECTION_URI_STRING
```

例如：

```
mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
%2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1
```

> **Warning**
>
> Apache Airflow 會剖析連線字串中的每個值。您不得使用單引號或雙引號，或將連線剖析為單一字串。

6. 對於加密金鑰，請從下拉式清單中選擇 AWS KMS 金鑰選項。
7. 在秘密名稱的文字欄位中輸入名稱，格式如下。

```
airflow/connections/YOUR_CONNECTION_NAME
```

例如：

```
airflow/connections/myconn
```

8. 選擇下一步。
9. 在設定秘密頁面上的秘密名稱和描述窗格中，執行下列動作。

a. 針對秘密名稱，請提供秘密的名稱。

b. （選用）針對描述，提供秘密的描述。

選擇下一步。

10. 在設定輪換 - 選用保留預設選項，然後選擇下一步。
11. 針對您要新增的任何其他變數，在 Secrets Manager 中重複這些步驟。
12. 在檢閱頁面上，檢閱您的秘密，然後選擇儲存。

範本程式碼

- 了解如何使用位於 的範例程式碼，在此頁面上使用 Apache Airflow 連線 (`myconn`) 的私密金鑰在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線。
- 了解如何使用位於 的範例程式碼，在此頁面上使用 Apache Airflow 變數 (`test-variable`) 的私密金鑰針對 AWS Secrets Manager Apache Airflow 變數在 中使用私密金鑰。

Resources

- 如需使用主控台和 設定 Secrets Manager 秘密的詳細資訊 AWS CLI，請參閱AWS Secrets Manager《使用者指南》中的建立秘密。
- 使用 Python 指令碼將大量 Apache Airflow 變數和連線遷移至 Secrets Manager，並將您的 Apache Airflow 連線和變數移至其中 AWS Secrets Manager。

後續步驟？

- 了解如何在 中產生權杖以存取 Apache Airflow UI存取 Apache Airflow。

管理 Amazon MWAA 環境

Amazon Managed Workflows for Apache Airflow 主控台包含內建選項，可設定 Apache Airflow UI 的私有或公有存取權。它還包含用於設定環境大小、何時擴展工作者的內建選項，以及可用於覆寫通常只能在 中存取的 Apache Airflow 組態的 Apache Airflow 組態選項`airflow.cfg`。本章說明如何在 Amazon MWAA 主控台上使用這些組態。

主題

- 設定 Amazon MWAA 環境類別
- 設定 Amazon MWAA 工作者自動擴展
- 設定 Amazon MWAA Webserver 自動擴展
- 在 Amazon MWAA 上使用 Apache Airflow 組態選項
- 更新 Amazon MWAA 環境
- 變更 Apache Airflow 版本
- 搭配 Amazon MWAA 使用啟動指令碼

設定 Amazon MWAA 環境類別

您為 Amazon MWAA 環境選擇的環境類別會決定 Celery Executor 執行所在的 AWS受管 AWS Fargate 容器大小，以及 Apache Airflow 排程器建立任務執行個體所在的 AWS受管 Amazon Aurora PostgreSQL 中繼資料資料庫。本主題說明每個 Amazon MWAA 環境類別，以及如何更新 Amazon MWAA 主控台上的環境類別。

章節

- 環境功能
- Apache Airflow 排程器

環境功能

下一節包含每個環境類別的預設並行 Apache Airflow 任務、隨機存取記憶體 (RAM) 和虛擬集中式處理單元 (vCPUs)。列出的並行任務假設任務並行不超過環境中的 Apache Airflow 工作者容量。

在下表中，DAG 容量是指 DAG 定義，而不是執行，並假設您的 DAGs是動態的，並使用 Apache Airflow 最佳實務撰寫。

任務執行取決於同時排程的數目，並假設設定為同時啟動的 DAG 執行數目不超過預設 `max_dagruns_per_loop_to_schedule`，以及本主題中詳述的工作者大小和數目。

mw1.micro

- 高達 25 DAG 容量
- 3 個並行任務（預設）
- 元件：
  - Web 伺服器：1 個 vCPU、3GB RAM
  - 工作者和排程器：1 個 vCPU、3GB RAM
  - 資料庫：2 個 vCPU、4GB RAM

> **Note**
>
> mw1.micro 不支援自動擴展。

mw1.small

- 高達 50 DAG 容量
- 5 個並行任務（預設）
- 元件：
  - Web 伺服器：1 個 vCPU，每個 2GB RAM
  - 工作者：1 個 vCPU，每個 2GB RAM
  - 排程器：1 個 vCPU，每個 2GB RAM
  - 資料庫：2 個 vCPU、4GB RAM

mw1.medium

- 高達 250 DAG 容量
- 10 個並行任務（預設）
- 元件：
  - Web 伺服器：每個 1 個 vCPU 2GB RAM
  - 工作者：每個 2 個 vCPU 4GB RAM
  - 排程器：每個 2 個 vCPU 4GB RAM

- 資料庫：2 個 vCPU 8GB RAM

mw1.large

- 高達 1000 DAG 容量
- 20 個並行任務（預設）
- 元件：
  - Web 伺服器：每個 2 個 vCPU 4GB RAM
  - 工作者：每個 4 個 vCPU 8GB RAM
  - 排程器：每個 4 個 vCPU 8GB RAM
  - 資料庫：2 個 vCPU 8GB RAM

mw1.xlarge

- 高達 2000 DAG 容量
- 40 個並行任務（預設）
- 元件：
  - Web 伺服器：4 個 vCPU，每個 12GB RAM
  - 工作者：每個 8 個 vCPU 24GB RAM
  - 排程器：每個 8 個 vCPU 24GB RAM
  - 資料庫：4 個 vCPU 32GB RAM

mw1.2xlarge

- 高達 4000 DAG 容量
- 80 個並行任務（預設）
- Componenets：
  - Web 伺服器：每個 8 個 vCPU 24GB RAM
  - 工作者：每個 16 個 vCPU 48GB RAM
  - 排程器：每個 16 個 vCPU 48GB RAM
  - 資料庫：8 個 vCPU 64GB RAM

您可以使用 `celery.worker_autoscale`來增加每個工作者的任務。如需詳細資訊，請參閱 the section called “高效能使用案例範例”。

Apache Airflow 排程器

下一節包含 Amazon MWAA 上可用的 Apache Airflow 排程器選項，以及排程器數目如何影響觸發器數目。

在 Apache Airflow 中，觸發器會管理其延遲的任務，直到符合使用觸發指定的特定條件為止。在 Amazon MWAA 中，觸發器會與排程器一起在相同的 Fargate 任務上執行。相應地增加排程器計數會增加可用觸發器的數量，最佳化環境管理延遲任務的方式。這可確保有效處理任務，並在滿足條件時立即排定任務以執行。

Apache Airflow v3

- v3 - 對於大於 mw1.micro 的環境， 接受從 2到 的值。 5除了預設為 的 mw12.micro 之外，所有環境大小預設為 。 1

Apache Airflow v2

- v2 - 對於大於 mw1.micro 的環境， 接受從 2到 的值。 5除了預設為 的 mw12.micro 之外，所有環境大小預設為 。 1

設定 Amazon MWAA 工作者自動擴展

自動擴展機制會自動增加 Apache Airflow 工作者的數量，以回應在 Amazon Managed Workflows for Apache Airflow 環境中執行和排入佇列的任務，並在沒有更多任務排入佇列或執行時處置額外的工作者。本主題說明如何透過指定使用 Amazon MWAA 主控台在環境中執行的 Apache Airflow 工作者數量上限，來設定自動擴展。

> **Note**
>
> Amazon MWAA 使用 Apache Airflow 指標來判斷何時需要額外的 Celery Executor 工作者，並視需要將 Fargate 工作者的數量增加到 指定的值`max-workers`。當額外的工作者完成工作和工作量減少時，Amazon MWAA 會將其移除，因此會縮減為 設定的值`min-workers`。
> 如果工作者在縮減規模時取得新任務，Amazon MWAA 會保留 Fargate 資源，而不會移除工作者。如需詳細資訊，請參閱 Amazon MWAA 自動擴展的運作方式。

章節

- 工作者擴展的運作方式
- 使用 Amazon MWAA 主控台
- 高效能使用案例範例
- 對卡在執行中狀態的任務進行故障診斷
- 後續步驟？

工作者擴展的運作方式

Amazon MWAA 使用 RunningTasks和 QueuedTasks指標，其中（執行中的任務 + 已排入佇列的任務）/（每個工作者的任務) =（必要的工作者）。如果所需的工作者數量大於目前的工作者數量，Amazon MWAA 會將 Fargate 工作者容器新增至該值，直到 指定的最大值為止max-workers。

隨著工作負載減少且 RunningTasks和 QueuedTasks 指標總和減少，Amazon MWAA 會請求 Fargate 縮減環境的工作者。仍然完成工作的任何工作者在縮減規模期間都會受到保護，直到他們完成工作為止。視工作負載而定，任務可能會在工作者縮減規模時排入佇列。

使用 Amazon MWAA 主控台

您可以選擇可在 Amazon MWAA 主控台上同時在環境中執行的工作者數量上限。根據預設，您最多可以指定 25 個值。

設定工作者數量

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。
5. 在環境類別窗格中，在工作者計數上限中輸入值。
6. 選擇儲存。

> **Note**
>
> 變更對您的環境生效可能需要幾分鐘的時間。

高效能使用案例範例

下一節說明您可用來在 環境中啟用高效能和平行處理的組態類型。

內部部署 Apache Airflow

一般而言，在內部部署 Apache Airflow 平台中，您可以在 `airflow.cfg` 檔案中設定任務平行處理、自動擴展和並行設定：

- `core.parallelism` – 每個排程器可同時執行的任務執行個體數目上限。
- `core.dag_concurrency` – DAGs（非工作者）。
- `celery.worker_autoscale` – 可在任何工作者上同時執行的任務數量上限和下限。

例如，如果 `core.parallelism` 設為 `100` 且 `core.dag_concurrency` 設為 `7`，則只有在您有 2 個 DAGs 時，您才能同時執行總共 個14任務。有鑑於，即使整體平行處理設定為（在 中`core.dag_concurrency`），每個 DAG 都會設定為僅同時執行七個任務 `100`（在中`core.parallelism`）。

> **Note**
>
> `core.dag_concurrency` Apache Airflow v3 中無法使用 。

在 Amazon MWAA 環境中

在 Amazon MWAA 環境中，您可以使用 在 Amazon MWAA 上使用 Apache Airflow 組態選項、 和工作者計數上限自動擴展機制設定 Amazon MWAA 環境類別，直接在 Amazon MWAA 主控台上設定這些設定。雖然 `core.dag_concurrency` Amazon MWAA 主控台的 Apache Airflow 組態選項無法在下拉式清單中使用，但您可以將它新增為自訂 Apache Airflow 組態選項。

假設您在建立環境時選擇下列設定：

1. mw1.small 環境類別，可控制每個工作者預設可執行的並行任務數量上限，以及容器的 vCPU。
2. `10` 工作者計數上限中的工作者預設設定。
3. 每個工作者`celery.worker_autoscale5,5`任務的 Apache Airflow 組態選項。

這表示您可以在環境中執行 50 個並行任務。任何超過 50 個的任務都會排入佇列，並等待執行中的任務完成。

執行更多並行任務。您可以使用下列組態來修改環境，以同時執行更多任務：

1. 根據預設，透過選擇 mw1.medium（依預設 10 個並行任務）環境類別，增加每個工作者可執行的並行任務數量上限和容器的 vCPU。
2. 新增 celery.worker_autoscale做為 Apache Airflow 組態選項。
3. 增加工作者計數上限。在此範例中，將工作者上限從 增加10為 ，使環境可執行的並行任務數量20加倍。

指定最小工作者。您也可以使用 AWS Command Line Interface () 指定在您環境中執行的 Apache Airflow 工作者數量下限和上限AWS CLI。例如：

```
aws mwaa update-environment --max-workers 10 --min-workers 10 --
name YOUR_ENVIRONMENT_NAME
```

若要進一步了解，請參閱 中的 update-environment 命令 AWS CLI。

對卡在執行中狀態的任務進行故障診斷

在極少數情況下，Apache Airflow 可能會認為仍有任務仍在執行中。若要解決此問題，您需要清除 Apache Airflow UI 中的分層任務。如需詳細資訊，請參閱對 Amazon Managed Workflows for Apache Airflow 進行故障診斷疑難排解主題。

後續步驟？

- 進一步了解建議您在 中調整環境效能的最佳實務Amazon MWAA 上 Apache Airflow 的效能調校。

設定 Amazon MWAA Webserver 自動擴展

對於執行 Apache Airflow v2.2.2 和更新版本的環境，Amazon MWAA 會動態擴展 Web 伺服器以處理波動的工作負載，進而防止尖峰負載期間發生效能問題。透過根據 CPU 使用率和作用中連線計數自動擴展 Web 伺服器數量，Amazon MWAA 可確保您的 Apache Airflow 環境可以順暢地適應增加的需求，無論是來自 REST API 請求、CLI 用量還是更多並行 Apache Airflow 使用者界面使用者。

章節

- Web 伺服器擴展的運作方式
- 使用 Amazon MWAA 主控台

Web 伺服器擴展的運作方式

Amazon MWAA 使用容器指標 CPUUtilization和負載平衡器指標 ActiveConnectionCount，根據流量判斷是否需要擴展 Web 伺服器。如果 CPUUtilization 大於 70 或ActiveConnectionCount大於 15，Amazon MWAA 會新增額外的 Fargate Web 伺服器容器，直到 指定的最大值為止MaxWebservers。

隨著流量減少且 CPUUtilization和 ActiveConnectionCount值減少，Amazon MWAA 會請求 Fargate 將環境的 Web 伺服器容器縮減至 設定的最小值MinimumWebservers。

使用 Amazon MWAA 主控台

您可以選擇可在 Amazon MWAA 主控台上同時在環境中執行的 Web 伺服器數量。根據預設，Web 伺服器數量下限為 2，Web 伺服器數量上限為 5。

設定 Web 伺服器的數量

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。
5. 在環境類別窗格中，在 Web 伺服器計數上限中輸入值。
6. 接著，在最低 Web 伺服器計數中輸入值。
7. 選擇儲存。

> **Note**
>
> 變更對您的環境生效可能需要幾分鐘的時間。

在 Amazon MWAA 上使用 Apache Airflow 組態選項

Apache Airflow 組態選項可以做為環境變數連接至 Amazon Managed Workflows for Apache Airflow 環境。您可以從建議的下拉式清單中選擇，或在 Amazon MWAA 主控台上為您的 Apache Airflow 版本指定自訂組態選項。本主題說明可用的 Apache Airflow 組態選項，以及如何使用這些選項來覆寫您環境中的 Apache Airflow 組態設定。

內容

- 先決條件
- 運作方式
- 使用組態選項載入外掛程式
- 組態選項概觀
  - Apache Airflow 組態選項
  - Apache Airflow 參考
  - 使用 Amazon MWAA 主控台
- 組態參考
  - 電子郵件組態
  - 任務組態
  - 排程器組態
  - 工作者組態
  - Web 伺服器組態
  - 觸發器組態
- 範例和範例程式碼
  - 範例 DAG
  - 電子郵件通知設定範例
- 後續步驟？

先決條件

您將需要下列項目，才能完成此頁面上的步驟。

- 許可 — 您的管理員 AWS 帳戶 必須已授予您環境的 AmazonMWAAFullConsoleAccess 存取控制政策的存取權。此外，您的執行角色必須允許您的 Amazon MWAA 環境，才能存取您環境所使用的 AWS 資源。
- 存取 — 如果您需要存取公有儲存庫，才能直接在 Web 伺服器上安裝相依性，您的環境必須設定公有網路 Web 伺服器存取。如需詳細資訊，請參閱 the section called “Apache Airflow 存取模式”。
- Amazon S3 組態 — 用於在 中存放 DAGs、自訂外掛程式`plugins.zip`和 Python 相依性的 Amazon S3 儲存貯體`requirements.txt`必須設定為啟用公開存取封鎖和版本控制。

運作方式

當您建立環境時，Amazon MWAA 會將您在 Airflow 組態選項中的 Amazon MWAA 主控台上指定的組態設定，做為環境變數連接至您環境的 AWS Fargate 容器。如果您在 中使用相同名稱的設定`airflow.cfg`，則您在 Amazon MWAA 主控台上指定的選項會覆寫 中的值`airflow.cfg`。

雖然我們預設不會在 Amazon MWAA 環境的 Apache Airflow UI `airflow.cfg`中公開 ，但您可以直接在 Amazon MWAA 主控台上變更 Apache Airflow 組態選項，包括`webserver.expose_config`公開組態的設定。

使用組態選項載入外掛程式

在 Apache Airflow v2 和更新版本中，外掛程式預設為使用 `core.lazy_load_plugins : True`設定「延遲」載入。如果您使用的是自訂外掛程式，則必須將 新增`core.lazy_load_plugins : False`為 Apache Airflow 組態選項，才能在每個 Airflow 程序開始時載入外掛程式，以覆寫預設設定。

組態選項概觀

當您在 Amazon MWAA 主控台上新增組態時，Amazon MWAA 會將組態寫入環境變數。

- 列出的選項。您可以在下拉式清單中選擇適用於 Apache Airflow 版本的其中一個組態設定。例如，`dag_concurrency : 16`。組態設定會轉譯為您環境的 Fargate 容器，做為 `AIRFLOW__CORE__DAG_CONCURRENCY : 16`
- 自訂選項。您也可以在下拉式清單中指定未針對 Apache Airflow 版本列出的 Airflow 組態選項。例如，`foo.user : YOUR_USER_NAME`。組態設定會翻譯為您環境的 Fargate 容器，做為 `AIRFLOW__FOO__USER : YOUR_USER_NAME`

Apache Airflow 組態選項

下圖說明您可以在 Amazon MWAA 主控台上自訂 Apache Airflow 組態選項的位置。

Apache Airflow 參考

如需 Apache Airflow 支援的組態選項清單，請參閱《Apache Airflow 參考指南》中的組態參考。 若要存取您在 Amazon MWAA 上執行的 Apache Airflow 版本選項，請從下拉式清單中選取版本。

使用 Amazon MWAA 主控台

下列程序會逐步解說將 Airflow 組態選項新增至環境的步驟。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。
5. 在 Airflow 組態選項窗格中選擇新增自訂組態。
6. 從下拉式清單中選擇組態並輸入值，或輸入自訂組態並輸入值。
7. 針對您要新增的每個組態，選擇新增自訂組態。
8. 選擇儲存。

組態參考

下一節包含 Amazon MWAA 主控台下拉式清單中可用 Apache Airflow 組態的清單。

電子郵件組態

以下清單顯示適用於 Apache Airflow v2 和 v3 的 Amazon MWAA 上可用的 Airflow 電子郵件通知組態選項。

我們建議將連接埠 587 用於 SMTP 流量。根據預設， 會 AWS 封鎖所有 Amazon EC2 執行個體連接埠 25 上的傳出 SMTP 流量。如果您想要在連接埠 25 上傳送傳出流量，您可以請求移除此限制。

氣流組態選項	說明	範例值
email.email_backend	用於 email_backend 中電子郵件通知的 Apache Airflow 公用程式。	airflow.utils.email.send_email_smtp
smtp.smtp_host	用於 smtp_host 中電子郵件地址的傳出伺服器名稱。	localhost
smtp.smtp_starttls	Transport Layer Security (TLS) 用於在 smtp_starttls 中透過網際網路加密電子郵件。	False

氣流組態選項	說明	範例值
smtp.smtp_ssl	Secure Sockets Layer (SSL) 用於連接 smtp_ssl 中的伺服器和電子郵件用戶端。	True
smtp.smtp_port	在 smtp_port 中指定給伺服器的傳輸控制通訊協定 (TCP) 連接埠。	587
smtp.smtp_mail_from	smtp_mail_from 中的傳出電子郵件地址。	myemail@domain.com

任務組態

以下清單顯示 Amazon MWAA for Apache Airflow v2 和 v3 上 Airflow 任務下拉式清單中可用的組態。

氣流組態選項	說明	範例值
core.default_task_retries	在 default_task_retries 中重試 Apache Airflow 任務的次數。	3
core.parallelism	可在整個環境中平行同時執行的任務執行個體數量上限 (平行處理)。	40

排程器組態

下列清單顯示在適用於 Apache Airflow v2 和 v3 的 Amazon MWAA 下拉式清單中可用的 Apache Airflow 排程器組態。

氣流組態選項	說明	範例值

氣流組態選項	說明	範例值
scheduler.catchup_by_default	指示排程器建立 DAG 執行，以在 catchup_by_default 中將 DAG 執行「趕上」至特定時間間隔。	False
scheduler.scheduler_zombie_task_threshold **Note** 不適用於 Apache Airflow v3。	告知排程器是否要將任務執行個體標記為失敗，並在 scheduler_zombie_task_threshold 中重新排程任務。	300

工作者組態

下列清單顯示在適用於 Apache Airflow v2 和 v3 的 Amazon MWAA 下拉式清單中可用的 Airflow 工作者組態。

氣流組態選項	說明	範例值
celery.worker_autoscale	可在任何工作者上使用 worker_autoscale 中的 Celery Executor 同時執行的任務數量上限和下限。值必須以逗號分隔，順序如下：`max_concurrency,min_concurrency` 。	16，12

Web 伺服器組態

下列清單顯示 Amazon MWAA for Apache Airflow v2 和 v3 下拉式清單中可用的 Apache Airflow Web 伺服器組態。

氣流組態選項	說明	範例值
webserver.default_ui_timezone Note 不適用於 Apache Airflow v3。	default_ui_timezone 中的預設 Apache Airflow UI 日期時間設定。 Note 設定 `default_ui_timezone` 選項不會變更 DAGs排程執行的時區。若要變更 DAGs的時區，您可以使用自訂外掛程式。如需詳細資訊，請參閱 the section called “變更 DAG 的時區”。	America/New_York

觸發器組態

以下清單顯示適用於 Apache Airflow v2 和 v3 的 Amazon MWAA 上可用的 Apache Airflow 觸發器組態。

氣流組態選項	說明	範例值
mwaa.trigger_enabled	用於在 Amazon MWAA 上啟用和停用觸發器。依預設，此值是設為 `True`。如果設為 `False`，Amazon MWAA 將不會在排程器上啟動任何觸發程序。	True
triggerer.default_capacity （在 v2 中）	定義每個觸發器可以平行執行的觸發次數。在 Amazon	125

氣流組態選項	說明	範例值
triggerer.capacity （在 v3 中）	MWAA 上，隨著兩個元件一起執行，每個觸發器和每個排程器都會設定此容量。對於1000小型60、中型和大型500、xlarge 和 2xlarge 125250執行個體，每個排程器的預設值分別設定為 、、、和 。	

範例和範例程式碼

範例 DAG

您可以使用下列 DAG 來列印 email_backend Apache Airflow 組態選項。若要執行 以回應 Amazon MWAA 事件，請將程式碼複製到 Amazon S3 儲存貯體上的環境 DAGs 資料夾。

```
from airflow.decorators import dag
    from datetime import datetime

    def print_var(**kwargs):
    email_backend = kwargs['conf'].get(section='email', key='email_backend')
    print("email_backend")
    return email_backend

    @dag(
    dag_id="print_env_variable_example",
    schedule_interval=None,
    start_date=datetime(yyyy, m, d),
    catchup=False,
    )
    def print_variable_dag():
    email_backend_test = PythonOperator(
    task_id="email_backend_test",
    python_callable=print_var,
    provide_context=True
    )

    print_variable_test = print_variable_dag()
```

電子郵件通知設定範例

下列 Apache Airflow 組態選項可用於使用應用程式密碼的 Gmail.com 電子郵件帳戶。如需詳細資訊，請參閱 Gmail 說明參考指南中的使用應用程式密碼登入。

後續步驟？

- 了解如何將 DAG 資料夾上傳至 中的 Amazon S3 儲存貯體新增或更新 DAGs。

更新 Amazon MWAA 環境

> **Note**
>
> 加拿大西部（卡加利）和亞太區域（馬來西亞）區域尚未支援 Amazon MWAA 正常更新。

Amazon MWAA 環境更新會套用最新的變更和安全性修補程式。您也可以編輯現有的組態並升級 Apache Airflow 版本。本指南說明更新 Amazon MWAA 環境的步驟。

內容

- 開始之前
- 工作者替換策略
- 更新環境資源
- 更新環境
  - 步驟一：指定詳細資訊
  - 步驟二：設定進階設定
  - 步驟三：檢閱和更新

開始之前

- 建立環境後，無法修改您為環境指定的 VPC 網路。
- 您需要將 Amazon S3 儲存貯體設定為封鎖所有公有存取，並啟用儲存貯體版本控制。

- 您需要 AWS 帳戶 具有使用 Amazon MWAA 許可的 ，以及 AWS Identity and Access Management (IAM) 中建立 IAM 角色的許可。如果您選擇 Apache Airflow Webserver 的私有網路存取模式，這會限制 Amazon VPC 內的 Apache Airflow 存取，則需要 IAM 中的許可才能建立 Amazon VPC 端點。
- 若要啟用 Graceful 環境更新，您需要升級至 Apache Airflow 2.4.3 版或更新版本。若要升級 Airflow 版本，請參閱 變更 Apache Airflow 版本。

工作者替換策略

您可以選擇工作者替換策略，以控制 Amazon MWAA 在環境更新期間處理作用中工作者的方式。您可以選取以下其中一個策略：

強制更新

強制更新是預設的工作者替換策略。強制更新會立即停止所有活躍工作者，導致執行中任務在更新期間失敗。

優雅更新

從容更新允許工作者在關機前繼續執行任務達 12 小時。它可以防止任務因為更新中斷而失敗，只要它們在 12 小時內完成即可。新任務會轉送給更新的工作者。

若要在現有環境上啟用 Graceful 更新，您必須完成一個強制更新，並確保環境位於 Apache Airflow 2.4.3 版或更新版本。

> **Note**
>
> 如果您在環境處於 `MAINTENANCE` 狀態時執行更新，任何進行中環境更新的工作者替換策略會從 切換`GRACEFUL`到 `FORCED`。您的更新會在維護完成後執行。

更新環境資源

根據預設，Amazon MWAA 環境更新會使用現有的環境組態。若要在不變更目前組態的情況下更新環境：

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 從環境清單中，選擇您要更新的環境。
3. 在環境頁面上，選擇編輯以編輯環境。

4. 選擇下一步，直到您進入檢閱和儲存頁面為止。
5. 在檢閱和儲存頁面上，檢閱您的變更，然後選擇儲存。

更新環境

下一節說明更新 Amazon MWAA 環境的步驟。

步驟一：指定詳細資訊

指定環境的詳細資訊

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 從環境清單中，選擇您要更新的環境。
3. 在環境頁面上，選擇編輯以編輯環境。
4. 在環境詳細資訊區段中，對於 Airflow 版本，從下拉式清單中選擇您要將環境升級到的新 Apache Airflow 版本編號。

> **Note**
>
> 升級之前，請確定您的 DAGs和其他工作流程資源與新的 Apache Airflow 版本相容。如需詳細資訊，請參閱 變更 Apache Airflow 版本。

5. 在 Amazon S3 中的 DAG 程式碼下，指定下列項目：
   a. S3 儲存貯體。選擇瀏覽 S3，然後選取您的 Amazon S3 儲存貯體，或輸入 Amazon S3 URI。
   b. DAGs。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體中的`dags`資料夾，或輸入 Amazon S3 URI。
   c. 外掛程式檔案 - 選用。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體上的`plugins.zip`檔案，或輸入 Amazon S3 URI。
   d. 需求檔案 - 選用。選擇瀏覽 S3，然後選取 Amazon S3 儲存貯體上的`requirements.txt`檔案，或輸入 Amazon S3 URI。
   e. 啟動指令碼檔案 - 選用，選擇瀏覽S3並選取 Amazon S3 儲存貯體上的指令碼檔案，或輸入 Amazon S3 URI。
6. 選擇下一步。

步驟二：設定進階設定

設定進階設定

1. 在 Web 伺服器存取下，選取您偏好的 Apache Airflow 存取模式：

   a. 私有網路。這會將 Apache Airflow UI 的存取權限制在 Amazon VPC 中已授予您環境 IAM 政策存取權的使用者。您需要許可才能建立此步驟的 Amazon VPC 端點。

   > **Note**
   >
   > 如果您的 Apache Airflow UI 僅在公司網路內存取，而且您不需要存取公有儲存庫以進行 Web 伺服器需求安裝，請選擇私有網路選項。如果您選擇此存取模式選項，則需要建立機制來存取 Amazon VPC 中的 Apache Airflow Web 伺服器。如需詳細資訊，請參閱 存取 Apache Airflow Webserver 的 VPC 端點（私有網路存取）。

   b. 公有網路。這可讓獲授予環境 IAM 政策存取權的使用者透過網際網路存取 Apache Airflow UI。

2. 在安全群組下，選擇用來保護 Amazon VPC 的安全群組：

   a. 根據預設，Amazon MWAA 會在您的 Amazon VPC 中建立安全群組，並在建立新的安全群組中使用特定傳入和傳出規則。

   b. 「選用」。取消選取建立新安全群組中的核取方塊，以選取最多 5 個安全群組。

   > **Note**
   >
   > 現有的 Amazon VPC 安全群組必須設定特定的傳入和傳出規則，以允許網路流量。若要進一步了解，請參閱 Amazon MWAA 上 VPC 的安全性。

3. 在環境類別下，選擇環境類別。

   我們建議您選擇支援工作負載所需的最小大小。您可以隨時變更環境類別。

4. 針對工作者計數上限，指定要在環境中執行的 Apache Airflow 工作者數目上限。

   如需詳細資訊，請參閱 高效能使用案例範例。

5. 指定 Web 伺服器計數上限和 Web 伺服器計數下限，以設定 Amazon MWAA 如何擴展您環境中的 Apache Airflow Web 伺服器。

   如需 Web 伺服器自動擴展的詳細資訊，請參閱 the section called “設定 Web 伺服器自動擴展”。

6. 在加密下，選擇資料加密選項：

   a. 根據預設，Amazon MWAA 會使用 擁有 AWS的金鑰來加密您的資料。

   b. 「選用」。選擇自訂加密設定（進階）以選擇不同的 AWS KMS 金鑰。如果您選擇在此步驟中指定客戶受管金鑰，則必須指定 AWS KMS 金鑰 ID 或 ARN。AWS KMS Amazon MWAA 不支援別名和多區域金鑰。如果您在 Amazon S3 儲存貯體上為伺服器端加密指定了 Amazon S3 金鑰，則必須為 Amazon MWAA 環境指定相同的金鑰。

   > **Note**
   >
   > 您必須擁有 金鑰的許可，才能在 Amazon MWAA 主控台上選取金鑰。您還必須連接中所述的政策，授予 Amazon MWAA 使用金鑰的許可連接金鑰政策。

7. 建議使用。在監控下，為 Airflow 記錄組態選擇一或多個日誌類別，將 Apache Airflow 日誌傳送至 CloudWatch Logs：

   a. Airflow 任務日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 任務日誌類型。

   b. Airflow Web 伺服器日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow Webserver 日誌類型。

   c. Airflow 排程器日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 排程器日誌類型。

   d. 氣流工作者日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow 工作者日誌類型。

   e. Airflow DAG 處理日誌。在日誌層級選擇要傳送至 CloudWatch Logs 的 Apache Airflow DAG 處理日誌類型。

8. 「選用」。針對 Airflow 組態選項，選擇新增自訂組態選項。

   您可以從建議的 Apache Airflow 組態選項下拉式清單中選擇 Apache Airflow 版本，或指定自訂組態選項。例如，`core.default_task_retries : 3`。

9. 在許可下，選擇執行角色：

   a. 根據預設，Amazon MWAA 會在建立新角色中建立執行角色。 您必須擁有建立 IAM 角色的許可，才能使用此選項。

   b. 「選用」。選擇輸入角色 ARN 以輸入現有執行角色的 Amazon Resource Name (ARN)。

10. 在更新規格下，選擇 工作者替換策略以控制在更新期間處理作用中工作者的方式。

11. 選擇下一步。

步驟三：檢閱和更新

若要檢閱環境摘要

- 檢閱環境摘要，選擇儲存。

> **Note**
>
> 使用強制更新更新環境大約需要二十到三十分鐘。寬限期更新最多可能需要 12 小時才能完成，因為它會等待您的進行中任務完成。

變更 Apache Airflow 版本

Amazon MWAA 支援次要版本升級和降級。這表示您可以將環境從 版本更新`x.4.z`為 `x.5.z`，或從更新`x.5.z`為 `x.4.z`。若要執行主要版本升級，例如從 版本`1.y.z`升級至 `2.y.z`，您必須建立新的環境並遷移資源。如需有關升級至 Apache Airflow 新主要版本的詳細資訊，請參閱《Amazon MWAA 遷移指南》中的遷移至新的 Amazon MWAA 環境。

在升級或降級程序期間，Amazon MWAA 會擷取環境中繼資料的快照、將工作者、排程器、Web 伺服器升級或降級至新的 Apache Airflow 版本，最後使用快照還原中繼資料資料庫。

升級或降級之前，請確定您的 DAGs和其他工作流程資源與您升級到的新 Apache Airflow 版本相容。如果您使用 `requirements.txt` 來管理相依性，您還必須確保您在需求中指定的相依性與新版本相容。

主題

- 升級或降級您的工作流程資源
- 指定新版本

升級或降級您的工作流程資源

每當您變更 Apache Airflow 版本時，請務必在 中參考正確的 `--constraint` URL`requirements.txt`。

> **Warning**
>
> 在升級或降級期間，指定與您的目標 Apache Airflow 版本不相容的需求可能會導致舊版 Apache Airflow 與先前需求版本進行冗長的轉返程序。

遷移您的工作流程資源

1. 建立 aws-mwaa-docker-images 儲存庫的分支，並複製 Amazon MWAA 本機執行器的副本。
2. 前往符合您要升級或降級之版本的 aws-mwaa-docker-images 儲存庫分支。
3. 若要更新您的 `requirements.txt`，請遵循《Amazon MWAA 使用者指南》中的管理 Python 相依性中建議的最佳實務。
4. （選用）若要加速升級或降級程序，請清除環境的中繼資料資料庫。具有大量中繼資料的環境可能需要更長的時間才能升級。
5. 成功測試工作流程資源後，請將 DAGs、 `requirements.txt`和 外掛程式複製到您環境的 Amazon S3 儲存貯體。

您現在已準備好編輯環境、指定新的 Apache Airflow 版本，並開始更新程序。

指定新版本

完成更新工作流程資源以確保與新 Apache Airflow 版本的相容性後，請執行下列動作來編輯環境詳細資訊，並指定您要升級的 Apache Airflow 版本。

> **Note**
>
> 當您執行升級或降級時，目前在環境上執行的所有任務都會在程序期間終止。更新程序最多可能需要兩個小時，在此期間您的環境將無法使用。

使用主控台指定新版本

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 從環境清單中，選擇您要升級或降級的環境。
3. 在環境頁面上，選擇編輯以編輯環境。
4. 在環境詳細資訊區段中，針對 Airflow 版本，從下拉式清單中選擇您要升級或降級環境的 Apache Airflow 版本編號。

5. 選擇下一步，直到您進入檢閱和儲存頁面為止。
6. 在檢閱和儲存頁面上，檢閱您的變更，然後選擇儲存。

當您套用變更時，您的環境會開始升級或降級程序。在此期間，您的環境狀態會指出 Amazon MWAA 正在採取的動作，以及程序是否成功。

在升級或降級成功的情況下，狀態為 UPDATING，然後CREATING_SNAPSHOT當 Amazon MWAA 擷取中繼資料的備份時。最後，狀態會先傳回至 UPDATING，然後在程序完成AVAILABLE時傳回至 。

如果環境無法向上或降級，您的環境狀態將為 ROLLING_BACK。如果轉返成功，狀態會先出現 UPDATE_FAILED，表示更新失敗，但環境可用。如果轉返失敗，狀態將為 UNAVAILABLE，表示您無法存取環境。

搭配 Amazon MWAA 使用啟動指令碼

啟動指令碼是您在環境 Amazon S3 儲存貯體中託管的 shell (`.sh`) 指令碼，類似於您的 DAGs、需求和外掛程式。在安裝需求和初始化 Apache Airflow 程序之前，Amazon MWAA 會在每個個別 Apache Airflow 元件 （ 工作者、排程器和 Web 伺服器 ） 的啟動期間執行此指令碼。使用啟動指令碼執行下列動作：

- 安裝執行時間 – 安裝工作流程和連線所需的 Linux 執行時間。
- 設定環境變數 – 為每個 Apache Airflow 元件設定環境變數。覆寫常見的變數，例如 PATH、PYTHONPATH和 LD_LIBRARY_PATH。
- 管理金鑰和字符 – 將自訂儲存庫的存取字符傳遞給 `requirements.txt`並設定安全金鑰。

下列主題說明如何設定啟動指令碼來安裝 Linux 執行期、設定環境變數，以及使用 CloudWatch Logs 疑難排解相關問題。

主題

- 設定啟動指令碼
- 使用啟動指令碼安裝 Linux 執行期
- 使用啟動指令碼設定環境變數

設定啟動指令碼

若要將啟動指令碼與現有的 Amazon MWAA 環境搭配使用，請將.sh檔案上傳至您環境的 Amazon S3 儲存貯體。然後，若要將指令碼與環境建立關聯，請在環境詳細資訊中指定下列項目：

- 指令碼的 Amazon S3 URL 路徑 – 儲存貯體中託管指令碼的相對路徑，例如， `s3://mwaa-environment/startup.sh`
- 指令碼的 Amazon S3 版本 ID – Amazon S3 儲存貯體中啟動 shell 指令碼的版本。每次更新指令碼時，您必須指定 Amazon S3 指派給檔案的版本 ID。版本 IDs為 Unicode、UTF-8 編碼、URL 就緒、不透明字串，長度不超過 1，024 個位元組，例如 `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`。

若要完成本節中的步驟，請使用下列範例指令碼。指令碼會輸出指派給 的值`MWAA_AIRFLOW_COMPONENT`。此環境變數可識別指令碼執行所在的每個 Apache Airflow 元件。

複製程式碼並在本機儲存為 `startup.sh`。

```
#!/bin/sh

echo "Printing Apache Airflow component"
echo $MWAA_AIRFLOW_COMPONENT
```

接著，將指令碼上傳至您的 Amazon S3 儲存貯體。

AWS 管理主控台

上傳 shell 指令碼（主控台）

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
2. 從儲存貯體清單中，選擇與您的環境相關聯的儲存貯體名稱。
3. 在 Objects (物件)標籤上，選擇 Upload (上傳)。
4. 在上傳頁面上，拖放您建立的 shell 指令碼。
5. 選擇上傳。

指令碼會列在物件清單中。Amazon S3 會為 檔案建立新的版本 ID。如果您更新指令碼並使用相同的檔案名稱再次上傳，則會將新的版本 ID 指派給檔案。

AWS CLI

建立和上傳 shell 指令碼 (CLI)

1. 開啟新的命令提示，並執行 Amazon S3 `ls`命令來列出和識別與您的環境相關聯的儲存貯體。

```
aws s3 ls
```

2. 導覽至您儲存 shell 指令碼的資料夾。`cp` 在新的提示視窗中使用 ，將指令碼上傳至您的儲存貯體。將 *amzn-s3-demo-bucket* 取代為您的資訊。

```
aws s3 cp startup.sh s3://amzn-s3-demo-bucket/startup.sh
```

如果成功，Amazon S3 會輸出物件的 URL 路徑：

```
upload: ./startup.sh to s3://amzn-s3-demo-bucket/startup.sh
```

3. 使用下列命令來擷取指令碼的最新版本 ID。

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix startup --
query 'Versions[?IsLatest].[VersionId]' --output text
```

```
BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

當您將指令碼與 環境建立關聯時，您可以指定此版本 ID。

現在，將指令碼與您的環境建立關聯。

AWS 管理主控台

將指令碼與環境建立關聯 （主控台）

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選取您要更新的環境資料列，然後選擇編輯。
3. 在指定詳細資訊頁面上，針對啟動指令碼檔案 - 選用，輸入指令碼的 Amazon S3 URL，例如：`s3://amzn-s3-demo-bucket/startup-sh.`。
4. 從下拉式清單中選擇最新版本，或瀏覽S3以尋找指令碼。

5. 選擇下一步，然後前往檢閱並儲存頁面。
6. 檢閱變更，然後選擇儲存。

環境更新可能需要 10 到 30 分鐘。Amazon MWAA 會在環境中的每個元件重新啟動時執行啟動指令碼。

AWS CLI

將指令碼與環境建立關聯 (CLI)

- 開啟命令提示，並使用 `update-environment` 指定指令碼的 Amazon S3 URL 和版本 ID。

```
aws mwaa update-environment \
--name your-mwaa-environment \
--startup-script-s3-path startup.sh \
--startup-script-s3-object-version BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

如果成功，Amazon MWAA 會傳回環境的 Amazon Resource Name (ARN)：

```
arn:aws:airflow:us-west-2:123456789012:environment/your-mwaa-environment
```

環境更新可能需要 10 到 30 分鐘。Amazon MWAA 會在環境中的每個元件重新啟動時執行啟動指令碼。

最後，擷取日誌事件，以驗證指令碼是否如預期般運作。當您為每個 Apache Airflow 元件啟用記錄時，Amazon MWAA 會建立新的日誌群組和日誌串流。如需詳細資訊，請參閱 Apache Airflow 日誌類型。

AWS 管理主控台

取得 Apache Airflow 日誌串流 （主控台）

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇您的環境。
3. 在監控窗格中，選擇您要存取日誌的日誌群組，例如 Airflow 排程器日誌群組 。
4. 在 CloudWatch 主控台的日誌串流清單中，選擇具有下列字首的串流：`startup_script_execution_ip`。

5. 在日誌事件窗格中，您將參考列印 值的命令輸出MWAA_AIRFLOW_COMPONENT。例如，對於排程器日誌，您將執行下列操作：

```
Printing Apache Airflow component
        scheduler
        Finished running startup script. Execution time: 0.004s.
        Running verification
        Verification completed
```

您可以重複上述步驟來存取工作者和 Web 伺服器日誌。

使用啟動指令碼安裝 Linux 執行期

使用啟動指令碼來更新 Apache Airflow 元件的作業系統，並安裝其他執行時間程式庫以搭配工作流程使用。例如，執行下列指令碼yum update來更新作業系統。

在啟動指令碼yum update中執行 時，您必須使用 排除 Python，--exclude=python*如範例所列。為了讓您的環境執行，Amazon MWAA 會安裝與您的環境相容的 Python 特定版本。因此，您無法使用啟動指令碼更新環境的 Python 版本。

```
#!/bin/sh

echo "Updating operating system"
sudo yum update -y --exclude=python*
```

若要在特定 Apache Airflow 元件上安裝執行時間，請使用 MWAA_AIRFLOW_COMPONENT和 if和 fi 條件式陳述式。此範例會執行單一命令，以在排程器和工作者上安裝程式libaio庫，但不會在 Web 伺服器上安裝程式庫。

> **Important**
>
> - 如果您已設定私有 Web 伺服器，則必須使用下列條件或在本機提供所有安裝檔案，以避免安裝逾時。
> - 使用 sudo執行需要管理權限的操作。

```
#!/bin/sh
```

```
if [[ "${MWAA_AIRFLOW_COMPONENT}" != "webserver" ]]
  then
    sudo yum -y install libaio

fi
```

您可以使用啟動指令碼來取得 Python 版本。

```
#!/bin/sh

export PYTHON_VERSION_CHECK=`python -c 'import sys; version=sys.version_info[:3];
 print("{0}.{1}.{2}".format(*version))'`
echo "Python version is $PYTHON_VERSION_CHECK"
```

Amazon MWAA 不支援覆寫預設 Python 版本，因為這可能會導致與已安裝的 Apache Airflow 程式庫不相容。

使用啟動指令碼設定環境變數

使用啟動指令碼來設定環境變數和修改 Apache Airflow 組態。以下定義了新的變數 `ENVIRONMENT_STAGE`。您可以在 DAG 或自訂模組中參考此變數。

```
#!/bin/sh

export ENVIRONMENT_STAGE="development"
echo "$ENVIRONMENT_STAGE"
```

使用啟動指令碼覆寫常見的 Apache Airflow 或系統變數。例如，您設定 `LD_LIBRARY_PATH` 指示 Python 在您指定的路徑中搜尋二進位檔。這可讓您使用外掛程式為工作流程提供自訂二進位檔：

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/local/airflow/plugins/your-custom-binary
```

預留環境變數

Amazon MWAA 會保留一組重要的環境變數。如果您覆寫預留變數，Amazon MWAA 會將其還原為其預設值。下列列出預留變數：

- MWAA__AIRFLOW__COMPONENT – 用來識別具有下列其中一個值的 Apache Airflow 元件：scheduler、 worker或 webserver。
- AIRFLOW__WEBSERVER__SECRET_KEY – 用於在 Apache Airflow Web 伺服器中安全簽署工作階段 Cookie 的私密金鑰。
- AIRFLOW__CORE__FERNET_KEY – 用於加密和解密存放在中繼資料資料庫中之敏感資料的金鑰，例如連線密碼。
- AIRFLOW_HOME – Apache Airflow 主目錄的路徑，其中組態檔案和 DAG 檔案存放在本機。
- AIRFLOW__CELERY__BROKER_URL – 用於 Apache Airflow 排程器與 Celery 工作者節點之間通訊的訊息中介裝置 URL。
- AIRFLOW__CELERY__RESULT_BACKEND – 用來存放 Celery 任務結果的資料庫 URL。
- AIRFLOW__CORE__EXECUTOR – Apache Airflow 必須使用的執行器類別。在 Amazon MWAA 中，這是 CeleryExecutor
- AIRFLOW__CORE__LOAD_EXAMPLES – 用來啟用或停用範例 DAGs的載入。
- AIRFLOW__METRICS__METRICS_BLOCK_LIST – 用來管理 Amazon MWAA 在 CloudWatch 中發出和擷取哪些 Apache Airflow 指標。
- SQL_ALCHEMY_CONN – RDS for PostgreSQL 資料庫的連線字串，用於將 Apache Airflow 中繼資料存放在 Amazon MWAA 中。
- AIRFLOW__CORE__SQL_ALCHEMY_CONN – 用於與 相同的用途SQL_ALCHEMY_CONN，但遵循新的 Apache Airflow 命名慣例。
- AIRFLOW__CELERY__DEFAULT_QUEUE – Apache Airflow 中 Celery 任務的預設佇列。
- AIRFLOW__OPERATORS__DEFAULT_QUEUE – 使用特定 Apache Airflow 運算子的任務的預設佇列。
- AIRFLOW_VERSION – 安裝在 Amazon MWAA 環境中的 Apache Airflow 版本。
- AIRFLOW_CONN_AWS_DEFAULT – 用來與其他 AWS 服務整合的預設 AWS 登入資料。
- AWS_DEFAULT_REGION – 設定預設 搭配預設登入資料 AWS 區域 使用，以與其他 AWS 服務整合。
- AWS_REGION – 如果已定義，此環境變數會覆寫環境變數AWS_DEFAULT_REGION和設定檔設定區域中的值。
- PYTHONUNBUFFERED – stderr 用來傳送stdout和串流至容器日誌。
- AIRFLOW__METRICS__STATSD_ALLOW_LIST – 用來設定允許逗號分隔字首清單，以傳送以清單元素開頭的指標。
- AIRFLOW__METRICS__STATSD_ON – 啟用傳送指標至 StatsD。

- AIRFLOW__METRICS__STATSD_HOST – 用來連線至StatSD協助程式。
- AIRFLOW__METRICS__STATSD_PORT – 用來連線至StatSD協助程式。
- AIRFLOW__METRICS__STATSD_PREFIX – 用來連線至StatSD協助程式。
- AIRFLOW__CELERY__WORKER_AUTOSCALE – 設定最大和最小並行。
- AIRFLOW__CORE__DAG_CONCURRENCY – 設定排程器在一個 DAG 中可同時執行的任務執行個體數目。
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG – 設定每個 DAG 的作用中任務數目上限。
- AIRFLOW__CORE__PARALLELISM – 定義可同時執行的任務執行個體數量上限。
- AIRFLOW__SCHEDULER__PARSING_PROCESSES – 設定排程器剖析以排程 DAGs 的程序數目上限。
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT – 定義重新傳遞訊息給另一個工作者之前，工作者等待確認任務的秒數。
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION – 設定基礎 Celery 傳輸 AWS 區域 的 。
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES – 設定基礎 Celery 傳輸的佇列。
- AIRFLOW_SCHEDULER_ALLOWED_RUN_ID_PATTERN – 用於在觸發 DAG run_id時驗證參數輸入的有效性。
- AIRFLOW__WEBSERVER__BASE_URL – 用來託管 Apache Airflow UI 的 Web 伺服器 URL。
- PYTHONPATH （僅適用於 Apache Airflow v2.9 和更新版本） – 由 Amazon MWAA 預留，以確保所有基本環境功能都能正常運作。

> **Note**
>
> 對於 v2.9 之前的 Apache Airflow 版本， PYTHONPATH 是未保留的環境變數。

未預留的環境變數

您可以使用啟動指令碼來覆寫未保留的環境變數。以下列出其中一些常見變數：

- PATH – 指定目錄清單，其中作業系統會搜尋可執行檔和指令碼。當命令在命令列中執行時，系統會檢查 中的目錄PATH，以尋找並執行命令。當您在 Apache Airflow 中建立自訂運算子或任務時，您可能需要依賴外部指令碼或可執行檔。如果包含這些檔案的目錄不在 PATH變數中指定的 中，則當系統

找不到任務時，任務會無法執行。透過將適當的目錄新增至 PATH，Apache Airflow 任務可以尋找並執行所需的可執行檔。

- PYTHONPATH – Python 解譯器用來判斷要搜尋匯入模組和套件的目錄。這是您可以新增至預設搜尋路徑的目錄清單。這可讓解譯器尋找和載入標準程式庫中未包含或安裝在系統目錄中的 Python 程式庫。使用此變數來新增您的模組和自訂 Python 套件，並將其與 DAGs搭配使用。

  > **Note**
  >
  > 對於 Apache Airflow v2.9 和更新版本， PYTHONPATH 是預留環境變數。

- LD_LIBRARY_PATH – 動態連結器和載入器在 Linux 中用來尋找和載入共用程式庫的環境變數。它會指定包含共用程式庫的目錄清單，這些程式庫會在預設系統程式庫目錄之前搜尋。使用此變數來指定您的自訂二進位檔。
- CLASSPATH – Java 執行期環境 (JRE) 和 Java 開發套件 (JDK) 用來在執行期尋找和載入 Java 類別、程式庫和資源。這是目錄、JAR 檔案和 ZIP 封存的清單，其中包含編譯的 Java 程式碼。

在 Amazon MWAA 上使用 DAG

若要在 Amazon Managed Workflows for Apache Airflow 環境上執行定向無環圖 (DAGs)，請將檔案複製到連接至您環境的 Amazon S3 儲存貯體，然後讓 Amazon MWAA 知道您DAGs 和支援檔案位於 Amazon MWAA 主控台的何處。Amazon MWAA 負責同步工作者、排程器和 Web 伺服器之間的 DAGs。本指南說明如何新增或更新 DAGs並在 Amazon MWAA 環境上安裝自訂外掛程式和 Python 相依性。

主題

- Amazon S3 儲存貯體概觀
- 新增或更新 DAGs
- 安裝自訂外掛程式
- 安裝 Python 相依性
- 刪除 Amazon S3 上的檔案

Amazon S3 儲存貯體概觀

Amazon MWAA 環境的 Amazon S3 儲存貯體必須封鎖公開存取。根據預設，所有 Amazon S3 資源 - 儲存貯體、物件和相關的子資源 （ 例如生命週期組態 ） 都是私有的。

- 只有建立儲存貯 AWS 帳戶 體的資源擁有者，才能存取資源。資源擁有者 （ 例如，您的管理員 ） 可以透過撰寫存取控制政策，將存取許可授予其他人。
- 您設定的存取政策必須具有將 DAGs、 中的自訂外掛程式`plugins.zip`和 中的 Python 相依性`requirements.txt`新增至 Amazon S3 儲存貯體的許可。如需包含必要許可的範例政策，請參閱 AmazonMWAAFullConsoleAccess。

Amazon MWAA 環境的 Amazon S3 儲存貯體必須啟用版本控制。啟用 Amazon S3 儲存貯體版本控制時，每當建立新版本時，就會建立新的複本。

- 為 中的自訂外掛程式啟用版本控制`plugins.zip`，並在 Amazon S3 儲存貯`requirements.txt`體的 中啟用 Python 相依性。
- 每次在 Amazon S3 儲存貯`requirements.txt`體上更新這些檔案時`plugins.zip`，您必須在 Amazon MWAA 主控台上指定 和 的版本。

新增或更新 DAGs

定向無環圖形 (DAGs) 是在將 DAG 結構定義為程式碼的 Python 檔案中定義。您可以使用 AWS CLI 或 Amazon S3 主控台將 DAGs上傳至您的環境。本主題說明使用 Amazon S3 儲存貯體中的 dags 資料夾，在 Amazon Managed Workflows for Apache Airflow 環境中新增或更新 Apache Airflow DAGs 的步驟。

章節

- 先決條件
- 運作方式
- 有哪些變更？
- 使用 Amazon MWAA CLI 公用程式測試 DAGs
- 將 DAG 程式碼上傳至 Amazon S3
- 在 Amazon MWAA 主控台上指定 DAGs路徑 （第一次）
- 存取 Apache Airflow UI 上的變更
- 後續步驟？

先決條件

您將需要下列項目，才能完成此頁面上的步驟。

- 許可 — 您的管理員 AWS 帳戶 必須已授予您環境的 AmazonMWAAFullConsoleAccess 存取控制政策的存取權。此外，您的執行角色必須允許 Amazon MWAA 環境，才能存取環境所使用的 AWS 資源。
- 存取 — 如果您需要存取公有儲存庫，才能直接在 Web 伺服器上安裝相依性，您的環境必須設定公有網路 Web 伺服器存取。如需詳細資訊，請參閱 the section called “Apache Airflow 存取模式”。
- Amazon S3 組態 — 用於在 中存放 DAGs、自訂外掛程式plugins.zip和 Python 相依性的 Amazon S3 儲存貯體requirements.txt必須設定為啟用公開存取封鎖和版本控制。

運作方式

導向無環圖形 (DAG) 是在將 DAG 結構定義為程式碼的單一 Python 檔案中定義。它包含下列項目：

- DAG 定義。
- 描述如何執行 DAG 和要執行之任務的運算子。

- 描述執行任務順序的運算子關係。

若要在 Amazon MWAA 環境上執行 Apache Airflow 平台，您需要將 DAG 定義複製到儲存貯體中的 dags 資料夾。例如，儲存貯體中的 DAG 資料夾應該類似：

Example DAG 資料夾

```
dags/
# dag_def.py
```

Amazon MWAA 每 30 秒會自動將新的和變更的物件從 Amazon S3 儲存貯體同步至 Amazon MWAA 排程器和工作者容器的/usr/local/airflow/dags資料夾，無論檔案類型為何，都會保留 Amazon S3 來源的檔案階層。新 DAGs 在 Apache Airflow UI 中列出所需的時間由 控制scheduler.dag_dir_list_interval。對現有 DAGs的變更將在下一個 DAG 處理迴圈上收取。

> **Note**
>
> 您不需要在 DAG 資料夾中包含airflow.cfg組態檔案。您可以從 Amazon MWAA 主控台覆寫預設 Apache Airflow 組態。如需詳細資訊，請參閱 在 Amazon MWAA 上使用 Apache Airflow 組態選項。

有哪些變更？

若要檢閱特定 Apache Airflow 版本的變更，請參閱版本備註頁面。

- Apache Airflow v3 組態：組態參考
- Apache Airflow v2 公有界面資訊：Airflow 的公有界面

使用 Amazon MWAA CLI 公用程式測試 DAGs

- 命令列界面 (CLI) 公用程式會在本機複寫 Amazon Managed Workflows for Apache Airflow 環境。
- CLI 會在本機建置類似於 Amazon MWAA 生產映像的 Docker 容器映像。您可以使用它來執行本機 Apache Airflow 環境，以在部署到 Amazon MWAA 之前開發和測試 DAGs、自訂外掛程式和相依性。
- 若要執行 CLI，請參閱 GitHub 上的 aws-mwaa-docker-images。

將 DAG 程式碼上傳至 Amazon S3

您可以使用 Amazon S3 主控台或 AWS Command Line Interface (AWS CLI) 將 DAG 程式碼上傳至 Amazon S3 儲存貯體。下列步驟假設您正在將程式碼 (.py) 上傳到 Amazon S3 儲存貯體dags中名為的資料夾。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，可讓您在命令列 Shell 中使用命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 進行快速組態aws configure。

使用 上傳 AWS CLI

1. 使用下列命令列出所有 Amazon S3 儲存貯體。

```
aws s3 ls
```

2. 使用下列命令列出您環境的 Amazon S3 儲存貯體中的檔案和資料夾。

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. 下列命令會將dag_def.py檔案上傳至dags資料夾。

```
aws s3 cp dag_def.py s3://amzn-s3-demo-bucket/dags/
```

如果您的 Amazon S3 儲存貯體上尚dags不存在名為 的資料夾，此命令會建立 dags 資料夾，並將名為 的檔案上傳dag_def.py到新資料夾。

使用 Amazon S3 主控台

Amazon S3 主控台是以 Web 為基礎的使用者介面，可用來建立和管理 Amazon S3 儲存貯體中的資源。下列步驟假設您有一個名為 DAGs 資料夾dags。

使用 Amazon S3 主控台上傳

1. 在 Amazon MWAA 主控台上開啟環境頁面。

2. 選擇環境。
3. 在 S3 窗格中的 DAG 程式碼中選取 S3 儲存貯體連結，以在主控台中開啟儲存貯體。 S3
4. 選擇 dags 資料夾。
5. 選擇上傳。
6. 選擇新增檔案。
7. 選取您 的本機副本dag_def.py，然後選擇上傳。

在 Amazon MWAA 主控台上指定 DAGs路徑 （第一次）

下列步驟假設您正在指定名為 的 Amazon S3 儲存貯體上資料夾的路徑dags。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇您要執行 DAGs的環境。
3. 選擇編輯。
4. 在 Amazon S3 窗格中的 DAG 程式碼上，選擇 DAG 資料夾欄位旁的瀏覽 S3。
5. 選取您的dags資料夾。
6. 選擇 Choose (選擇)。
7. 選擇下一步，更新環境。

存取 Apache Airflow UI 上的變更

您需要 AWS 帳戶 in AWS Identity and Access Management (IAM) 存取 Apache Airflow UI 的Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess許可。

存取您的 Apache Airflow UI

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇開啟氣流使用者介面。

後續步驟？

在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。

安裝自訂外掛程式

Amazon Managed Workflows for Apache Airflow 支援 Apache Airflow 的內建外掛程式管理員，可讓您使用自訂 Apache Airflow 運算子、勾點、感應器或介面。此頁面說明使用 `plugins.zip` 檔案在 Amazon MWAA 環境上安裝 Apache Airflow 自訂外掛程式的步驟。

內容

- 先決條件
- 運作方式
- 何時使用外掛程式
- 自訂外掛程式概觀
  - 自訂外掛程式目錄和大小限制
- 自訂外掛程式的範例
  - plugins.zip 中使用平面目錄結構的範例
  - 在 plugins.zip 中使用巢狀目錄結構的範例
- 建立 plugins.zip 檔案
  - 步驟一：使用 Amazon MWAA CLI 公用程式測試自訂外掛程式
  - 步驟二：建立 plugins.zip 檔案
- plugins.zip 上傳至 Amazon S3
  - 使用 AWS CLI
  - 使用 Amazon S3 主控台
- 在您的環境上安裝自訂外掛程式
  - 在 Amazon MWAA 主控台plugins.zip上指定 的路徑 （ 第一次 ）
  - 在 Amazon MWAA 主控台上指定plugins.zip版本
- plugins.zip 的範例使用案例
- 後續步驟？

先決條件

您將需要下列項目，才能完成此頁面上的步驟。

- 許可 — 您的管理員 AWS 帳戶 必須已授予您環境的 AmazonMWAAFullConsoleAccess 存取控制政策的存取權。此外，您的執行角色必須允許 Amazon MWAA 環境存取您的環境所使用的 AWS 資源。
- 存取 — 如果您需要存取公有儲存庫，才能直接在 Web 伺服器上安裝相依性，您的環境必須設定公有網路 Web 伺服器存取。如需詳細資訊，請參閱 the section called “Apache Airflow 存取模式”。
- Amazon S3 組態 — 用於在 中存放 DAGs、自訂外掛程式`plugins.zip`和 Python 相依性的 Amazon S3 儲存貯體`requirements.txt`必須設定為啟用公開存取封鎖和版本控制。

運作方式

若要在您的環境中執行自訂外掛程式，您必須執行三項操作：

1. 在本機建立`plugins.zip`檔案。
2. 將本機`plugins.zip`檔案上傳至您的 Amazon S3 儲存貯體。
3. 在 Amazon MWAA 主控台的外掛程式檔案欄位中指定此檔案的版本。

> **Note**
>
> 如果這是您第一次將 上傳至 Amazon S3 `plugins.zip`儲存貯體，您也需要在 Amazon MWAA 主控台上指定 檔案的路徑。您只需要完成此步驟一次。

何時使用外掛程式

外掛程式僅適用於擴展 Apache Airflow 使用者介面，如 Apache Airflow 文件中所述。自訂運算子可以直接與您的DAG程式碼放在 `/dags` 資料夾中。

如果您需要建立與外部系統的整合，請將它們放在 `/dags` 資料夾或其中的子資料夾，而不是資料夾中`plugins.zip`。在 Apache Airflow 2.x 中，外掛程式主要用於擴展 UI。

同樣地，其他相依性無法放置在 中`plugins.zip`。相反地，它們可以存放在 Amazon S3 `/dags`資料夾中的位置，在 Apache Airflow 啟動之前，它們將同步到每個 Amazon MWAA 容器。

> **Note**
>
> /dags 資料夾中或 中plugins.zip未明確定義 Apache Airflow DAG 物件的任何檔案都必須列在 .airflowignore 檔案中。

自訂外掛程式概觀

Apache Airflow 的內建外掛程式管理員只需在 $AIRFLOW_HOME/plugins 資料夾中捨棄檔案，即可將外部功能整合至其核心。您可以使用它來使用自訂 Apache Airflow 運算子、勾點、感應器或界面。下一節提供本機開發環境中平面和巢狀目錄結構的範例，以及產生的匯入陳述式，其決定 plugins.zip 內的目錄結構。

自訂外掛程式目錄和大小限制

Apache Airflow 排程器和工作者會在 環境的 AWS受管 Fargate 容器上啟動期間搜尋自訂外掛程式/usr/local/airflow/plugins/*。

- 目錄結構。目錄結構（位於 /*) 是以plugins.zip檔案的內容為基礎。例如，如果您的plugins.zip包含 operators目錄做為主要層級目錄，則會將目錄解壓縮至您環境中/usr/local/airflow/plugins/*operators*的 。
- 大小限制。我們建議plugins.zip的檔案小於 1 GB。檔案的大小越大plugins.zip，環境的啟動時間就越長。雖然 Amazon MWAA 不會明確限制plugins.zip檔案大小，但如果無法在十分鐘內安裝相依性，則 Fargate 服務將會逾時，並嘗試將環境復原至穩定狀態。

> **Note**
>
> 對於使用 Apache Airflow v2.0.2 的環境，Amazon MWAA 會限制 Apache Airflow Web 伺服器上的傳出流量，不允許直接在 Web 伺服器上安裝外掛程式或 Python 相依性。從 Apache Airflow 2.2.2 版開始，Amazon MWAA 可以直接在 Web 伺服器上安裝外掛程式和相依性。

自訂外掛程式的範例

下一節使用 Apache Airflow 參考指南中的範例程式碼來說明如何建構本機開發環境。

plugins.zip 中使用平面目錄結構的範例

Apache Airflow v3

下列範例顯示 Apache Airflow v3 的平面目錄結構`plugins.zip`檔案。

Example具有 PythonVirtualenvOperator plugins.zip 的平面目錄

下列範例顯示 中`PythonVirtualenvOperator`自訂外掛程式之 plugins.zip 檔案的主要層級樹狀目錄為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式。

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

下列範例顯示`PythonVirtualenvOperator`自訂外掛程式。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
```

```
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v2

下列範例顯示 Apache Airflow v2 的平面目錄結構`plugins.zip`檔案。

Example具有 PythonVirtualenvOperator plugins.zip 的平面目錄

下列範例顯示 中`PythonVirtualenvOperator`自訂外掛程式之 plugins.zip 檔案的主要層級樹狀目錄為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式。

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

下列範例顯示`PythonVirtualenvOperator`自訂外掛程式。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
```

```
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

在 plugins.zip 中使用巢狀目錄結構的範例

Apache Airflow v3

下列範例顯示 `plugins.zip` 檔案，其中包含 `hooks`、 `operators`和 `sensors`目錄的個別目錄。

Example plugins.zip

```
__init__.py
my_airflow_plugin.py
 hooks/
  |-- __init__.py
  |-- my_airflow_hook.py
 operators/
  |-- __init__.py
  |-- my_airflow_operator.py
  |-- hello_operator.py
 sensors/
  |-- __init__.py
  |-- my_airflow_sensor.py
```

下列範例顯示使用自訂外掛程式的 DAG (DAGs 資料夾) 中的匯入陳述式。

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_airflow_operator import MyOperator
from sensors.my_airflow_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
  task_id='taskB',
  my_field='some text'
 )

 hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



 sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
```

```
from operators.my_airflow_operator import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

下列範例顯示自訂外掛程式檔案中所需的每個匯入陳述式。

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensor/my_airflow_sensor.py

```
from airflow.sensors.base import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example Operator/my_airflow_operator.py

```
from airflow.operators.bash import BaseOperator
```

```
from airflow.utils.decorators import apply_defaults
from hooks.my_airflow_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example Operator/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

請遵循使用 Amazon MWAA CLI 公用程式測試自訂外掛程式中的步驟，然後建立 plugins.zip 檔案以壓縮plugins目錄中的內容。 例如 `cd plugins`。

Apache Airflow v2

下列範例顯示 plugins.zip 檔案，其中包含 hooks、 operators和 sensors目錄的個別目錄。

Example plugins.zip

```
__init__.py
 my_airflow_plugin.py
 hooks/
  |-- __init__.py
  |-- my_airflow_hook.py
 operators/
  |-- __init__.py
  |-- my_airflow_operator.py
  |-- hello_operator.py
 sensors/
  |-- __init__.py
  |-- my_airflow_sensor.py
```

下列範例顯示使用自訂外掛程式的 DAG (DAGs 資料夾) 中的匯入陳述式。

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_airflow_operator import MyOperator
from sensors.my_airflow_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
```

```
  schedule_interval='@once',
  default_args=default_args) as dag:

sens = MySensor(
 task_id='taskA'
)

op = MyOperator(
 task_id='taskB',
 my_field='some text'
)

hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
from operators.my_airflow_operator import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

下列範例顯示自訂外掛程式檔案中所需的每個匯入陳述式。

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensor/my_airflow_sensor.py

```
from airflow.sensors.base import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example Operator/my_airflow_operator.py

```
from airflow.operators.bash import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_airflow_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example Operator/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults
```

```
class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

請遵循使用 Amazon MWAA CLI 公用程式測試自訂外掛程式中的步驟，然後建立 plugins.zip 檔案以壓縮plugins目錄中的內容。 例如 cd plugins。

建立 plugins.zip 檔案

下列步驟說明我們建議在本機建立 plugins.zip 檔案的步驟。

步驟一：使用 Amazon MWAA CLI 公用程式測試自訂外掛程式

- 命令列界面 (CLI) 公用程式會在本機複寫 Amazon Managed Workflows for Apache Airflow 環境。
- CLI 會在本機建置類似於 Amazon MWAA 生產映像的 Docker 容器映像。您可以使用它來執行本機 Apache Airflow 環境，在部署到 Amazon MWAA 之前開發和測試 DAGs、自訂外掛程式和相依性。
- 若要執行 CLI，請參閱 GitHub 上的 aws-mwaa-docker-images。

步驟二：建立 plugins.zip 檔案

您可以使用內建的 ZIP 封存公用程式，或任何其他 ZIP 公用程式 （例如 7zip) 來建立 .zip 檔案。

> **Note**
>
> 當您建立 .zip 檔案時，Windows OS 的內建 zip 公用程式可能會新增子資料夾。建議您先驗證 plugins.zip 檔案的內容，再上傳至 Amazon S3 儲存貯體，以確保沒有新增其他目錄。

1. 將目錄變更為本機 Airflow 外掛程式目錄。例如：

```
myproject$ cd plugins
```

2. 執行下列命令，以確保內容具有可執行的許可（僅限 macOS 和 Linux)。

```
plugins$ chmod -R 755 .
```

3. 壓縮plugins資料夾中的內容。

```
plugins$ zip -r plugins.zip .
```

plugins.zip 上傳至 Amazon S3

您可以使用 Amazon S3 主控台或 AWS Command Line Interface (AWS CLI) 將plugins.zip檔案上傳至 Amazon S3 儲存貯體。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態aws configure。

使用 上傳 AWS CLI

1. 在命令提示中，導覽至儲存plugins.zip檔案的目錄。例如：

```
cd plugins
```

2. 使用下列命令列出所有 Amazon S3 儲存貯體。

```
aws s3 ls
```

3. 使用下列命令列出您環境的 Amazon S3 儲存貯體中的檔案和資料夾。

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

4. 使用下列命令，將plugins.zip檔案上傳至您環境的 Amazon S3 儲存貯體。

```
aws s3 cp plugins.zip s3://amzn-s3-demo-bucket/plugins.zip
```

使用 Amazon S3 主控台

Amazon S3 主控台是以 Web 為基礎的使用者介面，可用來建立和管理 Amazon S3 儲存貯體中的資源。

使用 Amazon S3 主控台上傳

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 S3 窗格中的 DAG 程式碼中選取 S3 儲存貯體連結，以在主控台中開啟您的儲存貯體。 S3
4. 選擇上傳。
5. 選擇新增檔案。
6. 選取您 的本機副本plugins.zip，然後選擇上傳。

在您的環境上安裝自訂外掛程式

本節說明如何透過指定 plugins.zip 檔案的路徑，以及在每次更新 zip 檔案時指定 plugins.zip 檔案的版本，來安裝您上傳至 Amazon S3 儲存貯體的自訂外掛程式。

在 Amazon MWAA 主控台**plugins.zip**上指定 的路徑 （第一次）

如果這是您第一次將 上傳至 Amazon S3 plugins.zip儲存貯體，您也需要在 Amazon MWAA 主控台上指定 檔案的路徑。您只需要完成此步驟一次。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 在 Amazon S3 窗格中的 DAG 程式碼上，選擇與外掛程式檔案 - 選用欄位相鄰的瀏覽 S3。
5. 選取 Amazon S3 儲存貯體上的 plugins.zip 檔案。
6. 選擇 Choose (選擇)。
7. 選擇下一步，更新環境。

在 Amazon MWAA 主控台上指定**plugins.zip**版本

每次在 Amazon S3 儲存貯plugins.zip體中上傳新版本的 時，您需要在 Amazon MWAA 主控台上指定plugins.zip檔案的版本。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 在 Amazon S3 窗格中的 DAG 程式碼上，在下拉式清單中選擇plugins.zip版本。
5. 選擇下一步。

plugins.zip 的範例使用案例

- 了解如何在 中建立自訂外掛程式使用 Apache Hive 和 Hadoop 的自訂外掛程式。
- 了解如何在 中建立自訂外掛程式自訂外掛程式以修補 PythonVirtualenvOperator 。
- 了解如何在 中建立自訂外掛程式搭配 Oracle 的自訂外掛程式。
- 了解如何在 中建立自訂外掛程式the section called “變更 DAG 的時區”。

後續步驟？

在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。

安裝 Python 相依性

Python 相依性是 Amazon Managed Workflows for Apache Airflow 環境上 Apache Airflow 版本 Apache Airflow 基本安裝中未包含的任何套件或分佈。本主題說明使用 Amazon S3 儲存貯體中的requirements.txt檔案在 Amazon MWAA 環境上安裝 Apache Airflow Python 相依性的步驟。

內容

- 先決條件
- 運作方式
- Python 相依性概觀
  - Python 相依性位置和大小限制
- 建立 requirements.txt 檔案

  - 步驟一：使用 Amazon MWAA CLI 公用程式測試 Python 相依性
  - 步驟二：建立 requirements.txt
- requirements.txt 上傳至 Amazon S3
  - 使用 AWS CLI
  - 使用 Amazon S3 主控台
- 在您的環境上安裝 Python 相依性
  - 在 Amazon MWAA 主控台requirements.txt上指定 的路徑 （第一次）
  - 在 Amazon MWAA 主控台上指定requirements.txt版本
- 存取 的日誌 requirements.txt
- 後續步驟？

先決條件

您將需要下列項目，才能完成此頁面上的步驟。

- 許可 — 您的管理員 AWS 帳戶 必須已授予您環境的 AmazonMWAAFullConsoleAccess 存取控制政策的存取權。此外，您的執行角色必須允許您的 Amazon MWAA 環境，才能存取您的環境所使用的 AWS 資源。
- 存取 — 如果您需要存取公有儲存庫，才能直接在 Web 伺服器上安裝相依性，您的環境必須設定公有網路 Web 伺服器存取。如需詳細資訊，請參閱 the section called “Apache Airflow 存取模式”。
- Amazon S3 組態 — 用於在 中存放 DAGs、自訂外掛程式`plugins.zip`和 Python 相依性的 Amazon S3 儲存貯體`requirements.txt`必須設定為啟用公開存取封鎖和版本控制。

運作方式

在 Amazon MWAA 上，您可以將`requirements.txt`檔案上傳到 Amazon S3 儲存貯體，然後在每次更新檔案時，在 Amazon MWAA 主控台上指定檔案的版本，以安裝所有 Python 相依性。Amazon MWAA 會執行 `pip3 install -r requirements.txt` 在 Apache Airflow 排程器和每個工作者上安裝 Python 相依性。

若要在您的環境上執行 Python 相依性，您必須執行三項作業：

1. 在本機建立`requirements.txt`檔案。
2. 將本機 上傳至您的 Amazon S3 `requirements.txt`儲存貯體。

3. 在 Amazon MWAA 主控台的要求檔案欄位中指定此檔案的版本。

> **Note**
>
> 如果這是您第一次建立 並將其上傳至 Amazon S3 `requirements.txt`儲存貯體，您也需要在 Amazon MWAA 主控台上指定 檔案的路徑。您只需要完成此步驟一次。

Python 相依性概觀

您可以從 Python 套件索引 (PyPi.org)，`.whl` PyPi

Python 相依性位置和大小限制

Apache Airflow 排程器和工作者會搜尋 `requirements.txt` 檔案中的套件，而套件會安裝在 的環境。 `/usr/local/airflow/.local/bin`

- 大小限制。我們建議`requirements.txt`檔案參考合併大小小於 1 GB 的程式庫。Amazon MWAA 需要安裝的程式庫越多，環境的啟動時間就越長。雖然 Amazon MWAA 不會明確限制已安裝程式庫的大小，但如果無法在十分鐘內安裝相依性，則 Fargate 服務將會逾時，並嘗試將環境復原至穩定狀態。

建立 requirements.txt 檔案

下列步驟說明我們建議在本機建立 requirements.txt 檔案的步驟。

步驟一：使用 Amazon MWAA CLI 公用程式測試 Python 相依性

- 命令列界面 (CLI) 公用程式會在本機複寫 Amazon Managed Workflows for Apache Airflow 環境。
- CLI 會在本機建置類似於 Amazon MWAA 生產映像的 Docker 容器映像。您可以使用它來執行本機 Apache Airflow 環境，以在部署到 Amazon MWAA 之前開發和測試 DAGs、自訂外掛程式和相依性。
- 若要執行 CLI，請參閱 GitHub 上的 aws-mwaa-docker-images。

步驟二：建立 **`requirements.txt`**

下一節說明如何從`requirements.txt`檔案中的 Python 套件索引指定 Python 相依性。

Apache Airflow v3

1. 在本機測試。在建立`requirements.txt`檔案之前，反覆新增其他程式庫，以尋找套件及其版本的正確組合。若要執行 Amazon MWAA CLI 公用程式，請參閱 GitHub 上的 aws-mwaa-docker-images。
2. 檢閱 Apache Airflow 套件額外項目。若要存取 Amazon MWAA 上為 Apache Airflow v3 安裝的套件清單，請參閱 GitHub 網站上的 aws-mwaa-docker-images`requirements.txt`。
3. 新增限制條件陳述式。在檔案頂端新增 Apache Airflow v3 環境的限制條件`requirements.txt`檔案。Apache Airflow 限制條件檔案會指定 Apache Airflow 發行時可用的提供者版本。

   在下列範例中，將 *{environment-version}* 取代為您環境的版本編號，並將 *{Python-version}* 取代為您環境相容的 Python 版本。

   如需有關與 Apache Airflow 環境相容的 Python 版本的資訊，請參閱 Apache Airflow 版本。

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   如果限制條件檔案判斷`xyz==1.0`套件與您環境中的其他套件不相容， `pip3 install` 將無法防止不相容的程式庫安裝到您的環境。如果任何套件的安裝失敗，您可以在 CloudWatch Logs 的對應日誌串流中存取每個 Apache Airflow 元件 （排程器、工作者和 Web 伺服器） 的錯誤日誌。如需日誌類型的詳細資訊，請參閱 the section called “存取 Airflow 日誌”。
4. Apache Airflow 套件。新增套件額外項目和版本 (==)。這有助於防止相同名稱但不同版本的套件安裝在您的環境中。

   ```
   apache-airflow[package-extra]==2.5.1
   ```
5. Python 程式庫。在 `requirements.txt` 檔案中新增套件名稱和版本 (==)。這有助於防止 PyPi.org 未來的重大更新自動套用。

   ```
   library == version
   ```

   Example Boto3 和 psycopg2-binary

   此範例僅供示範之用。boto 和 psycopg2-binary 程式庫包含在 Apache Airflow v3 的基本安裝中，不需要在`requirements.txt`檔案中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的套件沒有版本，Amazon MWAA 會從 PyPi.org 安裝最新版本的套件。此版本可能與 中的其他套件衝突`requirements.txt`。

Apache Airflow v2

1. 在本機測試。在建立`requirements.txt`檔案之前，反覆新增其他程式庫，以尋找套件及其版本的正確組合。若要執行 Amazon MWAA CLI 公用程式，請參閱 GitHub 上的 aws-mwaa-docker-images。
2. 檢閱 Apache Airflow 套件額外項目。若要存取 Amazon MWAA 上為 Apache Airflow v2 安裝的套件清單，請存取 GitHub 網站上的 aws-mwaa-docker-images`requirements.txt`。
3. 新增限制條件陳述式。在檔案頂端新增 Apache Airflow v2 環境的限制條件`requirements.txt`檔案。Apache Airflow 限制條件檔案會指定 Apache Airflow 發行時可用的提供者版本。

   從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制條件，Amazon MWAA 會為您指定一個限制條件，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。

   在下列範例中，將 *{environment-version}* 取代為您環境的版本編號，並將 *{Python-version}* 取代為您環境相容的 Python 版本。

   如需有關與 Apache Airflow 環境相容的 Python 版本的資訊，請參閱 Apache Airflow 版本。

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   如果限制條件檔案判斷`xyz==1.0`套件與您環境中的其他套件不相容， `pip3 install` 將無法防止不相容的程式庫安裝到您的環境。如果任何套件的安裝失敗，您可以在 CloudWatch Logs 的對應日誌串流中存取每個 Apache Airflow 元件 （ 排程器、工作者和 Web 伺服器 ） 的錯誤日誌。如需日誌類型的詳細資訊，請參閱 the section called “存取 Airflow 日誌”。
4. Apache Airflow 套件。新增套件額外項目和版本 (==)。這有助於防止相同名稱但不同版本的套件安裝在您的環境中。

```
apache-airflow[package-extra]==2.5.1
```

5. Python 程式庫。在 `requirements.txt` 檔案中新增套件名稱和版本 (==)。這有助於防止 PyPi.org 未來的重大更新自動套用。

```
library == version
```

Example Boto3 和 psycopg2-binary

此範例僅供示範之用。boto 和 psycopg2-binary 程式庫包含在 Apache Airflow v2 基本安裝中，不需要在`requirements.txt`檔案中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的套件沒有版本，Amazon MWAA 會從 PyPi.org 安裝最新版本的套件。此版本可能與 中的其他套件衝突`requirements.txt`。

`requirements.txt` 上傳至 Amazon S3

您可以使用 Amazon S3 主控台或 AWS Command Line Interface (AWS CLI) 將`requirements.txt`檔案上傳至 Amazon S3 儲存貯體。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一種開放原始碼工具，您可以使用命令列 shell 中的 命令與 AWS 服務互動。若要完成此頁面上的步驟，您需要下列項目：

- AWS CLI – 安裝第 2 版。
- AWS CLI – 使用 的快速組態`aws configure`。

使用 上傳 AWS CLI

1. 使用下列命令列出所有 Amazon S3 儲存貯體。

```
aws s3 ls
```

2. 使用下列命令列出您環境的 Amazon S3 儲存貯體中的檔案和資料夾。

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. 下列命令會將requirements.txt檔案上傳至 Amazon S3 儲存貯體。

```
aws s3 cp requirements.txt s3://amzn-s3-demo-bucket/requirements.txt
```

使用 Amazon S3 主控台

Amazon S3 主控台是以 Web 為基礎的使用者介面，可用來建立和管理 Amazon S3 儲存貯體中的資源。

使用 Amazon S3 主控台上傳

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 S3 窗格中的 DAG 程式碼中選取 S3 儲存貯體連結，以在主控台中開啟您的儲存貯體。 S3
4. 選擇上傳。
5. 選擇新增檔案。
6. 選取 的本機副本requirements.txt，然後選擇上傳。

在您的環境上安裝 Python 相依性

本節說明如何透過指定 requirements.txt 檔案的路徑，以及在每次更新時指定 requirements.txt 檔案的版本，來安裝您上傳至 Amazon S3 儲存貯體的相依性。

在 Amazon MWAA 主控台**requirements.txt**上指定 的路徑 （第一次）

如果這是您第一次建立 並將其上傳至 Amazon S3 requirements.txt儲存貯體，您也需要在 Amazon MWAA 主控台上指定 檔案的路徑。您只需要完成此步驟一次。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。

3. 選擇編輯。
4. 在 Amazon S3 窗格中的 DAG 程式碼上，選擇需求檔案 - 選用欄位旁的瀏覽 S3。
5. 選取 Amazon S3 儲存貯體上的 `requirements.txt` 檔案。
6. 選擇 Choose (選擇)。
7. 選擇下一步，更新環境。

您可以在環境完成更新之後立即開始使用新套件。

在 Amazon MWAA 主控台上指定**requirements.txt**版本

每次在 Amazon S3 儲存貯`requirements.txt`體中上傳新版本的 時，您需要在 Amazon MWAA 主控台上指定 `requirements.txt` 檔案的版本。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 在 Amazon S3 窗格中的 DAG 程式碼上，在下拉式清單中選擇`requirements.txt`版本。
5. 選擇下一步，更新環境。

您可以在環境完成更新之後立即開始使用新套件。

存取 的日誌 **requirements.txt**

您可以檢視排程器的 Apache Airflow 日誌，以排程工作流程和剖析`dags`資料夾。下列步驟說明如何在 Amazon MWAA 主控台上開啟排程器的日誌群組，以及在 CloudWatch Logs 主控台上存取 Apache Airflow 日誌。

存取 的日誌 **requirements.txt**

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在監控窗格中選擇 Airflow 排程器日誌群組。
4. 在`requirements_install_ip`日誌串流中選擇日誌。
5. 請參閱 環境上安裝的套件清單`/usr/local/airflow/.local/bin`。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 檢閱套件清單，以及是否有任何套件在安裝期間發生錯誤。如果發生錯誤，您會收到類似以下的錯誤：

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

後續步驟？

在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。

刪除 Amazon S3 上的檔案

此頁面說明版本控制如何在 Amazon Managed Workflows for Apache Airflow 環境的 Amazon S3 儲存貯體中運作，以及刪除 DAG`plugins.zip`、 或 `requirements.txt` 檔案的步驟。

內容

- 先決條件
- 版本控制概觀
- 運作方式
- 在 Amazon S3 上刪除 DAG
- 從環境移除 "current" requirements.txt 或 plugins.zip
- 刪除「非目前」（先前的）requirements.txt 或 plugins.zip 版本
- 使用生命週期自動刪除 "non-current"（先前的）版本和刪除標記
- 自動刪除 requirements.txt "non-current" 版本和刪除標記的生命週期政策範例
- 後續步驟？

先決條件

您將需要下列項目，才能完成此頁面上的步驟。

- 許可 — 您的管理員 AWS 帳戶 必須已授予您環境的 AmazonMWAAFullConsoleAccess 存取控制政策的存取權。此外，您的執行角色必須允許 Amazon MWAA 環境，才能存取環境所使用的 AWS 資源。
- 存取 — 如果您需要存取公有儲存庫，才能直接在 Web 伺服器上安裝相依性，您的環境必須設定公有網路 Web 伺服器存取。如需詳細資訊，請參閱 the section called “Apache Airflow 存取模式”。
- Amazon S3 組態 — 用於在 中存放 DAGs、自訂外掛程式`plugins.zip`和 Python 相依性的 Amazon S3 儲存貯體`requirements.txt`必須設定為啟用公開存取封鎖和版本控制。

版本控制概觀

Amazon S3 儲存貯`plugins.zip`體中的 `requirements.txt`和 已進行版本控制。為物件啟用 Amazon S3 儲存貯體版本控制，並從 Amazon S3 儲存貯體刪除成品（例如 plugins.zip) 時，檔案不會完全刪除。每當在 Amazon S3 上刪除成品時，就會建立新的檔案複本，該檔案為 404（找不到物件）錯誤/0k 檔案，其中顯示 `I'm not here`。Amazon S3 將此稱為刪除標記。刪除標記是檔案的「null」版本，其金鑰名稱（或金鑰）和版本 ID 與任何其他物件相同。

建議您定期刪除檔案版本和刪除標記，以降低 Amazon S3 儲存貯體的儲存成本。若要完全刪除「非目前」（先前的）檔案版本，您必須刪除檔案版本，然後刪除該版本的刪除標記。

運作方式

Amazon MWAA 每 30 秒在您的 Amazon S3 儲存貯體上執行一次同步操作。這會導致 Amazon S3 儲存貯體中的任何 DAG 刪除同步到 Fargate 容器的 Airflow 映像。

對於 `plugins.zip`和 `requirements.txt` 檔案，只有當 Amazon MWAA 使用自訂外掛程式和 Python 相依性建置 Fargate 容器的新 Airflow 映像時，才會在環境更新之後發生變更。如果您刪除任何 `requirements.txt`或 `plugins.zip` 檔案的目前版本，然後更新環境而不為已刪除的檔案提供新版本，則更新會失敗並顯示錯誤訊息，例如 `Unable to read version {version number} of file {file name}`。

在 Amazon S3 上刪除 DAG

DAG 檔案 (`.py`) 未進行版本控制，可直接在 Amazon S3 主控台上刪除。下列步驟說明如何刪除 Amazon S3 儲存貯體上的 DAG。

刪除 DAG

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 S3 窗格中的 DAG 程式碼中選取 S3 儲存貯體連結，以在主控台中開啟儲存貯體。
4. 選擇 dags 資料夾。
5. 選取 DAG、刪除。
6. 在刪除物件？下，輸入 delete。
7. 選擇 Delete objects (刪除物件)。

> **Note**
>
> Apache Airflow 會保留歷史 DAG 執行。在 Apache Airflow 中執行 DAG 之後，無論檔案狀態為何，它都會保留在 Airflow DAGs清單中，直到您在 Apache Airflow 中刪除它為止。若要刪除 Apache Airflow 中的 DAG，請選擇連結欄中的紅色「刪除」按鈕。

從環境移除 "current" requirements.txt 或 plugins.zip

目前，您無法在環境新增 plugins.zip 或 requirements.txt 之後將其移除，但我們正在處理此問題。在此期間，解決方法是分別指向空白文字或 zip 檔案。

刪除「非目前」（先前的）requirements.txt 或 plugins.zip 版本

Amazon S3 儲存貯體中的 `requirements.txt`和 `plugins.zip` 檔案會在 Amazon MWAA 上進行版本控制。如果您想要完全刪除 Amazon S3 儲存貯體上的這些檔案，您必須擷取物件的目前版本 (121212)（例如 plugins.zip)、刪除版本，然後移除檔案版本的刪除標記。

您也可以在 Amazon S3 主控台中刪除「非目前」（先前的）檔案版本；不過，您仍然需要使用下列其中一個選項刪除刪除標記。

- 若要擷取物件版本，請參閱《Amazon S3 指南》中的從已啟用版本控制的儲存貯體擷取物件版本。
- 若要刪除物件版本，請參閱《Amazon S3 指南》中的從已啟用版本控制的儲存貯體刪除物件版本。
- 若要移除刪除標記，請參閱《Amazon S3 指南》中的管理刪除標記。

使用生命週期自動刪除 "non-current"（先前的）版本和刪除標記

您可以設定 Amazon S3 儲存貯體的生命週期政策，在特定天數後刪除 Amazon S3 儲存貯體中的 Plugins.zip 和 requirements.txt 檔案的「非目前」版本，或移除過期物件的刪除標記。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 Amazon S3 中的 DAG 程式碼下，選擇您的 Amazon S3 儲存貯體。
4. 選擇建立生命週期規則。

自動刪除 requirements.txt "non-current" 版本和刪除標記的生命週期政策範例

使用下列範例建立生命週期規則，該規則會在三十天後永久刪除 requirements.txt 檔案的「非最新版本」版本及其刪除標記。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 Amazon S3 中的 DAG 程式碼下，選擇您的 Amazon S3 儲存貯體。
4. 選擇建立生命週期規則。
5. 在生命週期規則名稱中，輸入 `Delete previous requirements.txt versions and delete markers after thirty days`。
6. 在字首中，要求。
7. 在生命週期規則動作中，選擇永久刪除舊版本的物件，並刪除過期的刪除標記或未完成的分段上傳。
8. 在物件成為先前版本的天數中，輸入 `30`。
9. 在過期物件刪除標記中，選擇刪除過期物件刪除標記，物件會在 30 天後永久刪除。

後續步驟？

- 進一步了解管理刪除標記中的 Amazon S3 刪除標記。 https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-lifecycle.html
- 進一步了解即將過期物件中的 Amazon S3 生命週期。

聯網

本指南說明 Amazon MWAA 環境所需的 Amazon VPC 網路設定。

章節

- 關於 Amazon MWAA 上的聯網
- Amazon MWAA 上 VPC 的安全性
- 在 Amazon MWAA 上管理對服務特定 Amazon VPC 端點的存取
- 在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點
- 在 Amazon MWAA 上管理您自己的 Amazon VPC 端點

關於 Amazon MWAA 上的聯網

Amazon VPC 是與您的 連結的虛擬網路 AWS 帳戶。它可讓您透過提供對虛擬基礎設施和網路流量分段的精細控制，獲得雲端安全性和動態擴展的能力。此頁面說明支援 Amazon Managed Workflows for Apache Airflow 環境所需的具有公有路由或私有路由的 Amazon VPC 基礎設施。

內容

- 條款
- 支援的內容
- VPC 基礎設施概觀
  - 透過網際網路的公有路由
  - 沒有網際網路存取的私有路由
- Amazon VPC 和 Apache Airflow 存取模式的範例使用案例
  - 允許網際網路存取 - 新的 Amazon VPC 網路
  - 不允許網際網路存取 - 新的 Amazon VPC 網路
  - 不允許網際網路存取 - 現有的 Amazon VPC 網路

條款

公有路由

可存取網際網路的 Amazon VPC 網路。

私有路由

無法存取網際網路的 Amazon VPC 網路。

支援的內容

下表說明 Amazon VPCsAmazon MWAA 支援的類型。

Amazon VPC 類型	支援	
正在嘗試建立環境的帳戶所擁有的 Amazon VPC。	是	
共用的 Amazon VPC，其中多個 會 AWS 帳戶 建立其 AWS 資源。	是	

VPC 基礎設施概觀

當您建立 Amazon MWAA 環境時，Amazon MWAA 會根據您為環境選擇的 Apache Airflow 存取模式，為您的環境建立一到兩個 VPC 端點。這些端點會在 Amazon VPC 中顯示為具有私有 IPs彈性網路界面 (ENIs)。建立這些端點之後，任何目的地為這些 IPs流量都會私下或公開路由至您環境所使用的對應 AWS 服務。

下節說明透過網際網路公開路由流量所需的 Amazon VPC 基礎設施，或在 Amazon VPC 內私下路由流量。

透過網際網路的公有路由

本節說明具有公有路由之環境的 Amazon VPC 基礎設施。您需要下列 VPC 基礎設施：

- 一個 VPC 安全群組。VPC 安全群組充當虛擬防火牆，以控制執行個體上的傳入（傳入）和傳出（傳出）網路流量。
  - 最多可指定 5 個安全群組。
  - 安全群組必須指定自我參考的傳入規則給自己。
  - 安全群組必須為所有流量指定傳出規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。

  - 安全群組必須允許自我參考規則中的所有流量。例如 （建議） 所有存取自我參考安全群組的範例。
  - 安全群組可以透過指定 HTTPS 連接埠範圍443和 TCP 連接埠範圍 來選擇性地進一步限制流量5432。例如，（選用） 限制連接埠 5432 傳入存取的安全群組範例 和 （選用） 限制連接埠 443 傳入存取的安全群組範例。
- 兩個公有子網路。公有子網路是一種子網路，其與具有網際網路閘道路由的路由表相關聯。
  - 需要兩個公有子網路。這可讓 Amazon MWAA 在某個容器故障時，為其他可用區域中的環境建立新的容器映像。
  - 子網路必須位於不同的可用區域。例如 `us-east-1a` 和 `us-east-1b`。
  - 子網路必須使用彈性 IP 地址 (EIP) 路由至 NAT 閘道 （或 NAT 執行個體）。
  - 子網路必須具有路由表，將網際網路繫結流量導向網際網路閘道。
- 兩個私有子網路。私有子網路是與具有網際網路閘道路由的路由表沒有關聯的子網路。
  - 需要兩個私有子網路。這可讓 Amazon MWAA 在某個容器故障時，為其他可用區域中的環境建立新的容器映像。
  - 子網路必須位於不同的可用區域。例如 `us-east-1a` 和 `us-east-1b`。
  - 子網路必須具有 NAT 裝置 （閘道或執行個體） 的路由表。
  - 子網路不得路由至網際網路閘道。
  - 將 IPv6 子網路`true`的 `assignIpV6AddressOnCreation`設定為 。
  - 對於 IPv6 私有子網路，您必須擁有輸出限定網際網路閘道 (EIGW) 的連線。
- 網路存取控制清單 (ACL)。NACL 會在子網路層級管理 （透過允許或拒絕規則） 傳入和傳出流量。
  - NACL 必須具有允許所有流量的傳入規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。
  - NACL 必須具有允許所有流量的傳出規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。
  - 例如 （建議） 範例 ACLs。
- 兩個 NAT 閘道 （或 NAT 執行個體）。NAT 裝置會將流量從私有子網路中的執行個體轉送到網際網路或其他 AWS 服務，然後將回應路由回執行個體。
  - NAT 裝置必須連接到公有子網路。（每個公有子網路一個 NAT 裝置。）
  - NAT 裝置必須具有連接到每個公有子網路的彈性 IPv4 地址 (EIP)。
- 網際網路閘道。網際網路閘道會將 Amazon VPC 連線至網際網路和其他 AWS 服務。
  - 網際網路閘道必須連接到 Amazon VPC。

沒有網際網路存取的私有路由

本節說明具有私有路由之環境的 Amazon VPC 基礎設施。您需要下列 VPC 基礎設施：

- 一個 VPC 安全群組。VPC 安全群組充當虛擬防火牆，以控制執行個體上的傳入（傳入）和傳出（傳出）網路流量。
  - 最多可指定 5 個安全群組。
  - 安全群組必須指定自我參考的傳入規則給自己。
  - 安全群組必須為所有流量指定傳出規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。
  - 安全群組必須允許自我參考規則中的所有流量。例如（建議）所有存取自我參考安全群組的範例。
  - 安全群組可以透過指定 HTTPS 連接埠範圍`443`和 TCP 連接埠範圍 來選擇性地進一步限制流量`5432`。例如，（選用）限制連接埠 5432 傳入存取的安全群組範例 和 （選用）限制連接埠 443 傳入存取的安全群組範例。
- 兩個私有子網路。私有子網路是與具有網際網路閘道路由的路由表沒有關聯的子網路。
  - 需要兩個私有子網路。這可讓 Amazon MWAA 在某個容器故障時，為其他可用區域中的環境建立新的容器映像。
  - 子網路必須位於不同的可用區域。例如 `us-east-1a` 和 `us-east-1b`。
  - 子網路必須具有 VPC 端點的路由表。
  - 子網路必須具有 EIGW 的路由表，才能從網際網路下載做為 DAG 的一部分。
  - 子網路不得具有 NAT 裝置（閘道或執行個體）的路由表，也不得具有網際網路閘道。
- 網路存取控制清單 (ACL)。NACL 會在子網路層級管理（透過允許或拒絕規則）傳入和傳出流量。
  - NACL 必須具有允許所有流量的傳入規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。
  - NACL 必須具有拒絕所有流量的傳出規則 (`0.0.0.0/0`對於 IPv6，請使用 `::/0`)。
  - 例如（建議）範例 ACLs。
- 本機路由表。本機路由表是 VPC 內通訊的預設路由。
  - 本機路由表必須與私有子網路相關聯。
  - 本機路由表必須啟用 VPC 中的執行個體，才能與您自己的網路通訊。例如，如果您使用 AWS Client VPN 存取 Apache Airflow Web 伺服器的 VPC 介面端點，路由表必須路由至 VPC 端點。
- 您環境所使用的每個 AWS 服務的 VPC 端點，以及與 Amazon MWAA 環境位於相同 AWS 區域 和 Amazon VPC 中的 Apache Airflow VPC 端點。
  - Apache Airflow 的環境和 VPC 端點所使用的每個 AWS 服務的 VPC 端點。例如（必要）VPC 端點。

- VPC 端點必須啟用私有 DNS。
- VPC 端點必須與您環境的兩個私有子網路相關聯。
- VPC 端點必須與環境的安全群組相關聯。
- 每個端點的 VPC 端點政策必須設定為允許存取環境使用 AWS 的服務。例如 （建議） 允許所有存取的 VPC 端點政策範例。
- Amazon S3 的 VPC 端點政策必須設定為允許儲存貯體存取。例如 （建議） 允許儲存貯體存取的 Amazon S3 閘道端點政策範例。

Amazon VPC 和 Apache Airflow 存取模式的範例使用案例

本節說明 Amazon VPC 中網路存取的不同使用案例，以及 Amazon MWAA 主控台上的 Apache Airflow Web 伺服器存取模式選擇。

允許網際網路存取 - 新的 Amazon VPC 網路

如果您的組織允許 VPC 中的網際網路存取，而且您希望使用者透過網際網路存取 Apache Airflow Web 伺服器：

1. 建立具有網際網路存取的 Amazon VPC 網路。
2. 為您的 Apache Airflow Web 伺服器建立具有公有網路存取模式的環境。
3. 建議的內容：我們建議您同時使用建立 Amazon VPC 基礎設施、Amazon S3 儲存貯體和 Amazon MWAA 環境的 CloudFormation 快速入門範本。若要進一步了解，請參閱 Amazon Managed Workflows for Apache Airflow 的快速入門教學課程。

如果您的組織允許 VPC 中的網際網路存取，而且您想要將 Apache Airflow Webserver 存取限制在 VPC 內的使用者：

1. 建立具有網際網路存取的 Amazon VPC 網路。
2. 建立從電腦存取 Apache Airflow Webserver VPC 介面端點的機制。
3. 為您的 Apache Airflow Web 伺服器建立具有私有網路存取模式的環境。
4. 我們建議的內容：
    a. 建議您在 中使用 Amazon MWAA 主控台選項一：在 Amazon MWAA 主控台上建立 VPC 網路，或在 中使用 CloudFormation 範本選項二：建立具有網際網路存取的 Amazon VPC 網路。

b. 建議您在 中使用 AWS Client VPN 設定對 Apache Airflow Webserver 的存取教學課程：使用設定私有網路存取 AWS Client VPN。

不允許網際網路存取 - 新的 Amazon VPC 網路

如果您的組織不允許 VPC 中的網際網路存取：

1. 建立沒有網際網路存取的 Amazon VPC 網路。
2. 建立從電腦存取 Apache Airflow Webserver VPC 介面端點的機制。
3. 為您的環境使用的每個 AWS 服務建立 VPC 端點。
4. 為您的 Apache Airflow Web 伺服器建立具有私有網路存取模式的環境。
5. 我們建議的內容：

   a. 我們建議您使用 CloudFormation 範本來建立沒有網際網路存取的 Amazon VPC，以及 Amazon MWAA 在 中使用之每個 AWS 服務的 VPC 端點選項三：在沒有網際網路存取的情況下建立 Amazon VPC 網路。

   b. 建議您在 中使用 AWS Client VPN 設定對 Apache Airflow Webserver 的存取教學課程：使用設定私有網路存取 AWS Client VPN。

不允許網際網路存取 - 現有的 Amazon VPC 網路

如果您的組織不允許 VPC 中的網際網路存取，且您已擁有沒有網際網路存取的必要 Amazon VPC 網路：

1. 為您的環境使用的每個 AWS 服務建立 VPC 端點。
2. 為 Apache Airflow 建立 VPC 端點。
3. 建立從電腦存取 Apache Airflow Webserver VPC 介面端點的機制。
4. 為您的 Apache Airflow Web 伺服器建立具有私有網路存取模式的環境。
5. 我們建議的內容：

   a. 建議您建立和連接 Amazon MWAA 使用的每個 AWS 服務所需的 VPC 端點，以及 中 Apache Airflow 所需的 VPC 端點在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點。

   b. 建議您在 中使用 AWS Client VPN 設定對 Apache Airflow Webserver 的存取教學課程：使用設定私有網路存取 AWS Client VPN。

Amazon MWAA 上 VPC 的安全性

此頁面說明用於保護 Amazon Managed Workflows for Apache Airflow 環境的 Amazon VPC 元件，以及這些元件所需的組態。

內容

- 條款
- 安全性概觀
- 網路存取控制清單 (ACL)
  - （建議）範例 ACLs
- VPC security groups (VPC 安全群組)
  - （建議）所有存取自我參考安全群組的範例
  - （選用）限制連接埠 5432 傳入存取的安全群組範例
  - （選用）限制連接埠 443 傳入存取的安全群組範例
- VPC 端點政策（僅限私有路由）
  - （建議）允許所有存取的 VPC 端點政策範例
  - （建議）允許儲存貯體存取的 Amazon S3 閘道端點政策範例

條款

公有路由

可存取網際網路的 Amazon VPC 網路。

私有路由

無法存取網際網路的 Amazon VPC 網路。

安全性概觀

安全群組和存取控制清單 (ACLs) 提供使用您指定的規則來控制 Amazon VPC 中子網路和執行個體之間網路流量的方法。

- 往返子網路的網路流量可由存取控制清單 (ACLs) 控制。您只需要一個 ACL，而且相同的 ACL 可用於多個環境。

- 往返執行個體的網路流量可由 Amazon VPC 安全群組控制。您可以在每個環境使用一到五個安全群組。
- 往返執行個體的網路流量也可以由 VPC 端點政策控制。如果您的組織不允許 Amazon VPC 內的網際網路存取，而且您使用具有私有路由的 Amazon VPC 網路，則 VPC 端點AWS 和 Apache Airflow VPC 端點需要 VPC 端點政策。

網路存取控制清單 (ACL)

網路存取控制清單 (ACL) 可以在子網路層級管理（透過允許或拒絕規則）傳入和傳出流量。ACL 無狀態，這表示傳入和傳出規則必須分別明確指定。它用於指定允許進出 VPC 網路中執行個體的網路流量類型。

每個 Amazon VPC 都有預設 ACL，允許所有傳入和傳出流量。您可以編輯預設 ACL 規則，或建立自訂 ACL 並將其連接至子網路。子網路隨時只能連接一個 ACL，但一個 ACL 可以連接到多個子網路。

（建議）範例 ACLs

下列範例顯示傳入和傳出 ACL 規則，可用於具有公有路由或私有路由的 Amazon VPC。

規則編號	Type	通訊協定	連接埠範圍	來源	允許/拒絕
100	所有 IPv4 流量	全部	全部	0.0.0.0/0	允許
*	所有 IPv4 流量	全部	全部	0.0.0.0/0	拒絕

VPC security groups (VPC 安全群組)

VPC 安全群組可做為虛擬防火牆，控制執行個體層級的網路流量。安全群組具有狀態，這表示允許傳入連線時，允許其回覆。它用於指定 VPC 網路中執行個體在 中允許的網路流量類型。

每個 Amazon VPC 都有預設的安全群組。根據預設，它沒有傳入規則。它具有允許所有傳出流量的傳出規則。您可以編輯預設安全群組規則，或建立自訂安全群組並將其連接至您的 Amazon VPC。在 Amazon MWAA 上，您需要設定傳入和傳出規則，以引導 NAT 閘道上的流量。

（建議）所有存取自我參考安全群組的範例

下列範例顯示傳入安全群組規則，允許具有公有路由或私有路由的 Amazon VPC 的所有流量。此範例中的安全群組是自我參考規則。

Type	通訊協定	來源類型	來源		
所有流量	全部	全部	sg-0909e8e81919 / my-mwaa-vpc-security-group		

下列範例顯示傳出安全群組規則。

Type	通訊協定	來源類型	來源		
所有流量	全部	全部	0.0.0.0/0		

（選用）限制連接埠 5432 傳入存取的安全群組範例

下列範例顯示傳入安全群組規則，允許 Amazon Aurora PostgreSQL 中繼資料資料庫 (Amazon MWAA 擁有）在連接埠 5432 上的所有 HTTPS 流量用於您的環境。

> **Note**
>
> 如果您選擇使用此規則限制流量，則需要新增另一個規則，以允許連接埠 443 上的 TCP 流量。

Type	通訊協定	連接埠範圍	Source type (來源類型)	來源	
自訂 TCP	TCP	5432	Custom		

Type	通訊協定	連接埠範圍	Source type (來源類型)	來源	
				sg-0909e8e81919 / my-mwaa-vpc-security-group	

（選用）限制連接埠 443 傳入存取的安全群組範例

下列範例顯示傳入安全群組規則，允許連接埠 443 上 Apache Airflow Web 伺服器的所有 TCP 流量。

Type	通訊協定	連接埠範圍	Source type (來源類型)	來源	
HTTPS	TCP	443	Custom	sg-0909e8e81919 / my-mwaa-vpc-security-group	

VPC 端點政策（僅限私有路由）

VPC 端點 (AWS PrivateLink) 政策控制從私有子網路存取 AWS 服務的權限。VPC 端點政策是您連接到 VPC 閘道或介面端點的 IAM 資源政策。本節說明每個 VPC 端點的 VPC 端點政策所需的許可。

建議您針對您建立的每個 VPC 端點使用 VPC 介面端點政策，以允許完整存取所有 AWS 服務，並僅針對 AWS 許可使用您的執行角色。

（建議）允許所有存取的 VPC 端點政策範例

下列範例顯示具有私有路由之 Amazon VPC 的 VPC 介面端點政策。

```
{
  "Statement": [
```

```
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

（建議）允許儲存貯體存取的 Amazon S3 閘道端點政策範例

下列範例提供 VPC 閘道端點政策，可讓您存取具有私有路由之 Amazon VPC 的 Amazon ECR 操作所需的 Amazon S3 儲存貯體。除了存放 DAGs和支援檔案的儲存貯體之外，這是擷取 Amazon ECR 映像的必要項目。

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-us-east-1-starport-layer-bucket/*"]
    }
  ]
}
```

在 Amazon MWAA 上管理對服務特定 Amazon VPC 端點的存取

您可以使用 VPC 端點 (AWS PrivateLink) 將 VPC 私下連線至 上託管的服務， AWS 而不需要網際網路閘道、NAT 裝置、VPN 或防火牆代理。這些端點是水平可擴展且高度可用的虛擬裝置，允許 VPC 和 AWS 服務中的執行個體之間進行通訊。此頁面說明 Amazon MWAA 建立的 VPC 端點，以及如果您已在 Amazon Managed Workflows for Apache Airflow 上選擇私有網路存取模式，如何存取 Apache Airflow Webserver 的 VPC 端點。

內容

- 定價
- VPC 端點概觀

  - 公有網路存取模式
  - 私有網路存取模式
- 使用其他服務的許可 AWS
- 存取 VPC 端點
  - 在 Amazon VPC 主控台上存取 VPC 端點
  - 識別 Apache Airflow Web 伺服器及其 VPC 端點的私有 IP 地址
- 存取 Apache Airflow Webserver 的 VPC 端點（私有網路存取）
  - 使用 AWS Client VPN
  - 使用 Linux 堡壘主機
  - 使用Load Balancer（進階）

定價

- AWS PrivateLink 定價

VPC 端點概觀

當您建立 Amazon MWAA 環境時，Amazon MWAA 會為您的環境建立一到兩個 VPC 端點。這些端點會在您的 Amazon VPC 中顯示為具有私有 IPs彈性網路界面 (ENIs)。建立這些端點之後，任何目的地為這些 IPs流量都會私下或公開路由至您環境所使用的對應 AWS 服務。

公有網路存取模式

如果您為 Apache Airflow Web 伺服器選擇公有網路存取模式，網路流量會透過網際網路公開路由。

- Amazon MWAA 會為您的 Amazon Aurora PostgreSQL 中繼資料資料庫建立 VPC 介面端點。端點是在映射到您的私有子網路的可用區域中建立的，並且獨立於其他子網路 AWS 帳戶。
- 然後，Amazon MWAA 會將私有子網路的 IP 地址繫結至介面端點。這旨在支援從 Amazon VPC 的每個可用區域繫結單一 IP 的最佳實務。

私有網路存取模式

如果您為 Apache Airflow Web 伺服器選擇私有網路存取模式，網路流量會在 Amazon VPC 內私下路由。

- Amazon MWAA 會為您的 Apache Airflow Web 伺服器建立 VPC 介面端點，並為 Amazon Aurora PostgreSQL 中繼資料資料庫建立介面端點。端點會在映射到您的私有子網路的可用區域中建立，並獨立於其他子網路 AWS 帳戶。
- 然後，Amazon MWAA 會將私有子網路的 IP 地址繫結至介面端點。這旨在支援從 Amazon VPC 的每個可用區域繫結單一 IP 的最佳實務。

使用其他服務的許可 AWS

介面端點使用 AWS Identity and Access Management (IAM) 中您環境的執行角色來管理您環境所用 AWS 資源的許可。隨著環境開啟更多 AWS 服務，每個服務都需要您使用環境的執行角色來設定許可。若要新增許可，請參閱 Amazon MWAA 執行角色。

如果您已為 Apache Airflow Web 伺服器選擇私有網路存取模式，您還必須在每個端點的 VPC 端點政策中允許許可。若要進一步了解，請參閱 the section called “VPC 端點政策 （僅限私有路由）”。

存取 VPC 端點

本節說明如何存取 Amazon MWAA 建立的 VPC 端點，以及如何識別 Apache Airflow VPC 端點的私有 IP 地址。

在 Amazon VPC 主控台上存取 VPC 端點

下一節顯示存取 Amazon MWAA 所建立 VPC 端點的步驟，以及如果您使用 Amazon VPC 的私有路由，您可能已建立的任何 VPC 端點。

存取 VPC 端點

1. 在 Amazon VPC 主控台上開啟端點頁面。
2. 選取您的 AWS 區域。
3. 請參閱 Amazon MWAA 建立的 VPC 介面端點，以及您在 Amazon VPC 中使用私有路由時可能已建立的任何 VPC 端點。

若要進一步了解具有私有路由的 Amazon VPC 所需的 VPC 服務端點，請參閱 在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點。

識別 Apache Airflow Web 伺服器及其 VPC 端點的私有 IP 地址

下列步驟說明如何擷取 Apache Airflow Web 伺服器及其 VPC 介面端點的主機名稱，以及其私有 IP 地址。

1. 使用 following AWS Command Line Interface (AWS CLI) 命令來擷取 Apache Airflow Webserver 的主機名稱。

```
aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query
 'Environment.WebserverUrl'
```

您得到類似下列回應的內容：

```
"99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
```

2. 在上一個命令的回應中傳回的主機名稱上執行 dig 命令。例如：

```
dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-
west-2.airflow.amazonaws.com
```

您得到類似下列回應的內容：

```
vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com.
```

3. 使用 following AWS Command Line Interface (AWS CLI) 命令擷取上一個命令回應中傳回的 VPC 端點 DNS 名稱。例如：

```
aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-
svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

您得到類似下列回應的內容：

```
"DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com",
```

4. 在 Apache Airflow 主機名稱及其 VPC 端點 DNS 名稱上執行 nslookup 或 dig 命令，以擷取 IP 地址。例如：

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

您得到類似下列回應的內容：

```
192.0.5.1
```

```
192.0.6.1
```

存取 Apache Airflow Webserver 的 VPC 端點（私有網路存取）

如果您已為 Apache Airflow Web 伺服器選擇私有網路存取模式，則需要建立機制來存取 Apache Airflow Web 伺服器的 VPC 介面端點。您必須針對這些資源使用與 Amazon MWAA 環境相同的 Amazon VPC、VPC 安全群組和私有子網路。

使用 AWS Client VPN

AWS Client VPN 是一種受管的用戶端型 VPN 服務，可用來安全地存取內部部署網路中的 AWS 資源和資源。它使用 OpenVPN 用戶端從任何位置提供安全的 TLS 連線。

建議您遵循 Amazon MWAA 教學課程來設定 Client VPN：教學課程：使用 設定私有網路存取 AWS Client VPN。

使用 Linux 堡壘主機

堡壘主機是一種伺服器，其目的是從外部網路提供私有網路的存取權，例如透過網際網路從電腦存取。Linux 執行個體位於公有子網路中，並使用安全群組進行設定，允許從連接到執行堡壘主機之基礎 Amazon EC2 執行個體的安全群組進行 SSH 存取。

我們建議您遵循 Amazon MWAA 教學課程來設定 Linux 堡壘主機：教學課程：使用 Linux 堡壘主機設定私有網路存取。

使用Load Balancer（進階）

下一節顯示您需要套用至 Application Load Balancer 的組態。

1. 目標群組。您需要使用指向 Apache Airflow Web 伺服器及其 VPC 介面端點之私有 IP 地址的目標群組。我們建議您指定兩個私有 IP 地址做為已註冊的目標，因為只有使用一個地址可以降低可用性。如需如何識別私有 IP 地址的詳細資訊，請參閱 the section called “識別 Apache Airflow Web 伺服器及其 VPC 端點的私有 IP 地址”。
2. 狀態碼。建議您在目標群組設定中使用 `200`和 `302` 狀態碼。否則，如果 Apache Airflow Webserver 的 VPC 端點回應`302 Redirect`錯誤，目標可能會標記為運作狀態不良。
3. HTTPS 接聽程式。您需要指定 Apache Airflow Web 伺服器的目標連接埠。例如：

通訊協定	連線埠

通訊協定	連線埠
HTTPS	443

4. ACM 新網域。如果您想要在 中關聯 SSL/TLS 憑證 AWS Certificate Manager，則需要為負載平衡器的 HTTPS 接聽程式建立新的網域。
5. ACM 憑證區域。如果您想要在 中關聯 SSL/TLS 憑證 AWS Certificate Manager，則需要上傳到與環境 AWS 區域 相同的 。例如：

    - Example要上傳憑證的區域

    ```
    aws acm import-certificate --certificate fileb://Certificate.pem --certificate-
    chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --
    region us-west-2
    ```

在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點

沒有網際網路存取的現有 Amazon VPC 網路需要額外的 VPC 服務端點 (AWS PrivateLink)，才能在 Amazon Managed Workflows for Apache Airflow 上使用 Apache Airflow。此頁面說明 Amazon MWAA 所使用的 AWS 服務所需的 VPC 端點、Apache Airflow 所需的 VPC 端點，以及如何建立 VPC 端點並將其連接至具有私有路由的現有 Amazon VPC。

內容

- 定價
- 私有網路和私有路由
- （必要）VPC 端點
- 連接所需的 VPC 端點
    - AWS 服務所需的 VPC 端點
    - Apache Airflow 所需的 VPC 端點
- （選用）為您的 Amazon S3 VPC 介面端點啟用私有 IP 地址
    - 使用 Route 53
    - 具有自訂 DNS VPCs

定價

- AWS PrivateLink 定價

私有網路和私有路由

私有網路存取模式會將對 Apache Airflow UI 的存取限制在 Amazon VPC 中已獲授予環境 IAM 政策存取權的使用者。

當您建立具有私有 Web 伺服器存取權的環境時，您必須在 Python wheel 封存檔 (`.whl`) 中封裝所有相依性，然後在 `.whl`中參考 `requirements.txt`。如需使用 wheel 封裝和安裝相依性的說明，請參閱使用 Python wheel 管理相依性。

下圖說明在 Amazon MWAA 主控台上尋找私有網路選項的位置。

- 私有路由。沒有網際網路存取的 Amazon VPC 會限制 VPC 內的網路流量。此頁面假設您的 Amazon VPC 沒有網際網路存取，並且需要您環境所使用的每個 AWS 服務的 VPC 端點，以及與 Amazon MWAA 環境相同的 AWS 區域 和 Amazon VPC 中 Apache Airflow 的 VPC 端點。

（必要）VPC 端點

下一節顯示沒有網際網路存取的 Amazon VPC 所需的必要 VPC 端點。它列出 Amazon MWAA 使用的每個 AWS 服務的 VPC 端點，包括 Apache Airflow 所需的 VPC 端點。

```
com.amazonaws.us-east-1.s3
com.amazonaws.us-east-1.monitoring
com.amazonaws.us-east-1.logs
com.amazonaws.us-east-1.sqs
com.amazonaws.us-east-1.kms
```

> **Note**
>
> 使用 Transit Gateway 或任何其他未直接前往 AWS API 端點的路由時，建議您將 AWS PrivateLink 新增至下列服務的 Amazon MWAA 私有子網路：
>
> - Amazon S3

- Amazon SQS
- CloudWatch Logs
- CloudWatch 指標
- AWS KMS （如適用）

這可確保您的 Amazon MWAA 環境可以安全有效地與這些服務通訊，而無需透過公有網際網路路由流量，從而提高安全性和效能。

連接所需的 VPC 端點

本節說明為具有私有路由的 Amazon VPC 連接所需 VPC 端點的步驟。

AWS 服務所需的 VPC 端點

下一節顯示將 環境所用 AWS 服務的 VPC 端點連接到現有 Amazon VPC 的步驟。

將 VPC 端點連接至私有子網路

1. 在 Amazon VPC 主控台上開啟端點頁面。
2. 選取您的 AWS 區域。
3. 建立 Amazon S3 的端點：
   a. 選擇建立端點。
   b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**`.s3`**，然後按下鍵盤上的 Enter。
   c. 建議您選擇針對閘道類型列出的服務端點。

      例如 **`com.amazonaws.us-west-2.s3 amazon Gateway`**
   d. 在 VPC 中選擇您環境的 Amazon VPC。
   e. 確定已選取您在不同可用區域中的兩個私有子網路，且已選取啟用 DNS 名稱來啟用該私有 DNS。
   f. 選擇您環境的 Amazon VPC 安全群組。
   g. 在政策中選擇完整存取。
   h. 選擇建立端點。
4. 建立 CloudWatch Logs 的端點：

a. 選擇建立端點。

b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.logs**，然後按下鍵盤上的 Enter。

c. 選取服務端點。

d. 在 VPC 中選擇您環境的 Amazon VPC。

e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。

f. 選擇您環境的 Amazon VPC 安全群組。

g. 在政策中選擇完整存取。

h. 選擇建立端點。

5. 建立 CloudWatch 監控的端點：

a. 選擇建立端點。

b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.monitoring**，然後按下鍵盤上的 Enter。

c. 選取服務端點。

d. 在 VPC 中選擇您環境的 Amazon VPC。

e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。

f. 選擇您環境的 Amazon VPC 安全群組。

g. 在政策中選擇完整存取。

h. 選擇建立端點。

6. 建立 Amazon SQS 的端點：

a. 選擇建立端點。

b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.sqs**，然後按下鍵盤上的 Enter。

c. 選取服務端點。

d. 在 VPC 中選擇您環境的 Amazon VPC。

e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。

f. 選擇您環境的 Amazon VPC 安全群組。

g. 在政策中選擇完整存取。

h. 選擇建立端點。

7. 建立 的端點 AWS KMS：

a. 選擇建立端點。

b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.kms**，然後按下鍵盤上的 Enter。
c. 選取服務端點。
d. 在 VPC 中選擇您環境的 Amazon VPC。
e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。
f. 選擇您環境的 Amazon VPC 安全群組。
g. 在政策中選擇完整存取。
h. 選擇建立端點。

Apache Airflow 所需的 VPC 端點

下一節顯示將 Apache Airflow 的 VPC 端點連接到現有 Amazon VPC 的步驟。

將 VPC 端點連接至私有子網路

1. 在 Amazon VPC 主控台上開啟端點頁面。
2. 選取您的 AWS 區域。
3. 建立 Apache Airflow API 的端點：
    a. 選擇建立端點。
    b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.airflow.api**，然後按下鍵盤上的 Enter。
    c. 選取服務端點。
    d. 在 VPC 中選擇您環境的 Amazon VPC。
    e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。
    f. 選擇您環境的 Amazon VPC 安全群組。
    g. 在政策中選擇完整存取。
    h. 選擇建立端點。
4. 建立 Apache Airflow 環境的第一個端點：
    a. 選擇建立端點。
    b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**.airflow.env**，然後按下鍵盤上的 Enter。
    c. 選取服務端點。
    d. 在 VPC 中選擇您環境的 Amazon VPC。

e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。

f. 選擇您環境的 Amazon VPC 安全群組。

g. 在政策中選擇完整存取。

h. 選擇建立端點。

5. 建立 Apache Airflow 操作的第二個端點：

a. 選擇建立端點。

b. 在依屬性篩選或依關鍵字文字搜尋欄位中，輸入：**`.airflow.ops`**，然後按下鍵盤上的 Enter。

c. 選取服務端點。

d. 在 VPC 中選擇您環境的 Amazon VPC。

e. 確定已選取不同可用區域中的兩個私有子網路，且已啟用啟用 DNS 名稱。

f. 選擇您環境的 Amazon VPC 安全群組。

g. 在政策中選擇完整存取。

h. 選擇建立端點。

（選用）為您的 Amazon S3 VPC 介面端點啟用私有 IP 地址

Amazon S3 Interface 端點不支援私有 DNS。S3 端點請求仍會解析為公有 IP 地址。若要將 S3 地址解析為私有 IP 地址，您需要在 Route 53 中為 S3 區域端點新增私有託管區域。 S3

使用 Route 53

本節說明使用 Route 533 為 S3 介面端點啟用私有 IP 地址的步驟。

1. 為您的 Amazon S3 VPC 介面端點（例如 s3.eu-west-1.amazonaws.com) 建立私有託管區域，並將其與您的 Amazon VPC 建立關聯。
2. 為您的 Amazon S3 VPC 介面端點（例如 s3.eu-west-1.amazonaws.com) 建立 ALIAS 記錄，以解析為您的 VPC 介面端點 DNS 名稱。
3. 建立 ALIAS Amazon S3 介面端點（例如 *.s3.eu-west-1.amazonaws.com) 的萬用字元記錄，可解析為 VPC 介面端點 DNS 名稱。

具有自訂 DNS VPCs

如果您的 Amazon VPC 使用自訂 DNS 路由，您需要透過建立 CNAME 記錄，在 DNS 解析程式（而非 Route 53，通常是執行 DNS 伺服器的 EC2 執行個體）中進行變更。例如：

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value:  *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

在 Amazon MWAA 上管理您自己的 Amazon VPC 端點

Amazon MWAA 使用 Amazon VPC 端點與設定 Apache Airflow 環境所需的各種 AWS 服務整合。有兩個主要使用案例適用於管理您的自有端點：

1. 這表示當您使用 AWS Organizations管理多個 AWS 帳戶 和共用資源時，可以在共用的 Amazon VPC 中建立 Apache Airflow 環境。
2. 它可讓您將許可縮小到使用端點的特定資源，以使用更嚴格的存取政策。

如果您選擇管理自己的 VPC 端點，則需負責為環境 RDS for PostgreSQL 資料庫和環境 Web 伺服器建立自己的端點。

如需 Amazon MWAA 如何在雲端中部署 Apache Airflow 的詳細資訊，請參閱 Amazon MWAA 架構圖。

> **Important**
>
> Amazon MWAA 不會驗證客戶受管端點的 IP 地址類型 (`AddressType`) 選項，因此請務必正確指定 `AddressType`（有效選項為 IPv4 或 IPv6）。

在共用的 Amazon VPC 中建立環境

如果您使用 AWS Organizations 管理多個 AWS 帳戶 共用資源，您可以使用客戶管理的 VPC 端點搭配 Amazon MWAA，與您組織中的另一個帳戶共用環境資源。

當您設定共用 VPC 存取時，擁有主要 Amazon VPC (擁有者) 的帳戶會與屬於相同組織的其他帳戶 (參與者) 共用 Amazon MWAA 所需的兩個私有子網路。共用這些子網路的參與者帳戶可以檢視、建立、修改和刪除共用 Amazon VPC 中的環境。

假設您有一個帳戶 ，Owner做為組織中Root的帳戶，並擁有 Amazon VPC 資源，以及參與者帳戶 Participant，即同一組織的成員。當 在與 共用的 Amazon VPC 中Participant建立新的 Amazon MWAA 時Owner，Amazon MWAA 會先建立服務 VPC 資源，然後進入PENDING狀態長達 72 小時。

環境狀態從 變更為 CREATING 後PENDING，代表 的委託人會Owner建立所需的端點。若要這樣做，Amazon MWAA 會在 Amazon MWAA 主控台中列出資料庫和 Web 伺服器端點。您也可以呼叫 GetEnvironment API 動作來取得服務端點。

> **Note**
>
> 如果您用來共用資源的 Amazon VPC 是私有 Amazon VPC，您仍然必須完成中所述的步驟the section called “管理對 VPC 端點的存取”。本主題涵蓋設定與 AWS 整合 AWS 之其他服務相關的不同 Amazon VPC 端點集，例如 Amazon ECR、Amazon ECS 和 Amazon SQS。這些服務對於在雲端中操作和管理 Apache Airflow 環境至關重要。

先決條件

在共用 VPC 中建立 Amazon MWAA 環境之前，您需要下列資源：

- AWS 帳戶，Owner用作擁有 Amazon VPC 的帳戶。
- AWS Organizations 組織單位，MyOrganization建立為根目錄。
- 下的第二個 AWS 帳戶ParticipantMyOrganization，為建立新環境的參與者帳戶提供服務。

此外，我們建議您熟悉在 Amazon VPC 中共用資源時，擁有者和參與者的責任和許可。

建立 Amazon VPC

首先，建立擁有者和參與者帳戶將共用的新 Amazon VPC：

1. 使用 登入 主控台，Owner然後開啟 CloudFormation 主控台。使用下列範本來建立堆疊。此堆疊會佈建多個聯網資源，包括 Amazon VPC，以及這兩個帳戶在此案例中將共用的子網路。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: >-
This template deploys a VPC, with a pair of public and private subnets spread
 across two Availability Zones. It deploys an internet gateway, with a default
```

```
route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
and default routes for them in the private subnets.
Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: mwaa-
  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16
  PublicSubnet1CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the public subnet in the first
Availability Zone
    Type: String
    Default: 10.192.10.0/24
  PublicSubnet2CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the public subnet in the second
Availability Zone
    Type: String
    Default: 10.192.11.0/24
  PrivateSubnet1CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the private subnet in the first
Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the private subnet in the second
Availability Zone
    Type: String
    Default: 10.192.21.0/24
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
    CidrBlock: !Ref VpcCIDR
    EnableDnsSupport: true
    EnableDnsHostnames: true
    Tags:
      - Key: Name
```

```
      Value: !Ref EnvironmentName
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
    Tags:
      - Key: Name
      Value: !Ref EnvironmentName
  InternetGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ1)'
  PublicSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
     VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ2)'
  PrivateSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet1CIDR
```

```
      MapPublicIpOnLaunch: false
        Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Subnet (AZ1)'
  PrivateSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Subnet (AZ2)'
  NatGateway1EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway2EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway1:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1
  NatGateway2:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2
  PublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Routes'
  DefaultPublicRoute:
```

```
    Type: 'AWS::EC2::Route'
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
  PrivateRouteTable1:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Routes (AZ1)'
  DefaultPrivateRoute1:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1
  PrivateSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1
  PrivateRouteTable2:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Routes (AZ2)'
  DefaultPrivateRoute2:
    Type: 'AWS::EC2::Route'
    Properties:
```

```
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2
  PrivateSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2
  SecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupName: mwaa-security-group
      GroupDescription: Security group with a self-referencing inbound rule.
      VpcId: !Ref VPC
  SecurityGroupIngress:
    Type: 'AWS::EC2::SecurityGroupIngress'
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: '-1'
      SourceSecurityGroupId: !Ref SecurityGroup
      Outputs:
        VPC:
          Description: A reference to the created VPC
          Value: !Ref VPC
          PublicSubnets:
          Description: A list of the public subnets
          Value: !Join
            - ','
            - - !Ref PublicSubnet1
            - !Ref PublicSubnet2
        PrivateSubnets:
          Description: A list of the private subnets
          Value: !Join
            - ','
            - - !Ref PrivateSubnet1
            - !Ref PrivateSubnet2
        PublicSubnet1:
          Description: A reference to the public subnet in the 1st Availability
Zone
          Value: !Ref PublicSubnet1
        PublicSubnet2:
          Description: A reference to the public subnet in the 2nd Availability
Zone
          Value: !Ref PublicSubnet2
```

```
        PrivateSubnet1:
          Description: A reference to the private subnet in the 1st Availability
Zone
          Value: !Ref PrivateSubnet1
        PrivateSubnet2:
          Description: A reference to the private subnet in the 2nd Availability
Zone
          Value: !Ref PrivateSubnet2
        SecurityGroupIngress:
          Description: Security group with self-referencing inbound rule
          Value: !Ref SecurityGroupIngress
```

2. 佈建新的 Amazon VPC 資源後，導覽至 AWS Resource Access Manager 主控台，然後選擇建立資源共享。
3. 從可與 共用的可用子網路清單中，選擇您在第一個步驟中建立的子網路Participant。

建立 環境

完成下列步驟，以使用客戶管理的 Amazon VPC 端點建立 Amazon MWAA 環境。

1. 使用 登入Participant，然後開啟 Amazon MWAA 主控台。完成步驟一：指定詳細資訊，以指定新環境的 Amazon S3 儲存貯體、DAG 資料夾和相依性。如需詳細資訊，請參閱入門。
2. 在設定進階設定頁面的聯網下，從共用的 Amazon VPC 選擇子網路。
3. 在端點管理下，從下拉式清單中選擇 CUSTOMER。
4. 保留頁面上其餘選項的預設值，然後在檢閱和建立頁面上選擇建立環境。

環境以 CREATING 狀態開始，然後變更為 PENDING。當環境為 時PENDING，請使用主控台記下資料庫端點服務名稱和 Web 伺服器端點服務名稱 （如果您設定私有 Web 伺服器）。

當您使用 Amazon MWAA 主控台建立新環境時。Amazon MWAA 會使用必要的傳入和傳出規則建立新的安全群組。記下該安全群組 ID。

在下一節中， Owner將使用服務端點和安全群組 ID 在共用的 Amazon VPC 中建立新的 Amazon VPC 端點。

建立 Amazon VPC 端點

完成下列步驟，為您的環境建立所需的 Amazon VPC 端點。

1. AWS 管理主控台 使用 登入 Owner，開啟 https：//https://console.aws.amazon.com/vpc/。
2. 從左側導覽面板中選擇安全群組，然後使用下列傳入和傳出規則，在共用的 Amazon VPC 中建立新的安全群組：

	Type	通訊協定	Source type (來源類型)	來源
傳入	所有流量	全部	全部	您的環境安全群組
傳出	所有流量	全部	全部	0.0.0.0/0

> **Warning**
>
> Owner 帳戶必須在Owner帳戶中設定安全群組，以允許從新環境到共用 Amazon VPC 的流量。您可以在 中建立新的安全群組Owner，或編輯現有的安全群組。

3. 選擇端點，然後使用先前步驟中的端點服務名稱，為環境資料庫和 Web 伺服器 （ 如果處於私有模式 ） 建立新的端點。選擇共用的 Amazon VPC、您用於環境的子網路，以及環境的安全群組。

如果成功，環境將從PENDING返回 變更為 CREATING，最後變更為 AVAILABLE。當它是時AVAILABLE，您可以登入 Apache Airflow 主控台。

共用 Amazon VPC 故障診斷

使用以下參考來解決在共用 Amazon VPC 中建立環境時遇到的問題。

PENDING 狀態CREATE_FAILED後 中的環境

- 驗證 Owner 是否Participant使用 與 共用子網路AWS Resource Access Manager。
- 確認資料庫和 Web 伺服器的 Amazon VPC 端點是在與環境相關聯的相同子網路中建立。
- 確認與端點搭配使用的安全群組允許來自環境所用安全群組的流量。Owner 帳戶會建立將 中安全群組參考Participant為 的規則*123456789012*/*security-group-id*：。

Type	通訊協定	Source type (來源類型)	來源
所有流量	全部	全部	*123456789012* /*sg-0909e8e81919*

如需詳細資訊，請參閱擁有者和參與者的責任和許可

環境卡在 **PENDING** 狀態

驗證每個 VPC 端點狀態，以確保其為 Available。如果您使用私有 Web 伺服器設定環境，您還必須為 Web 伺服器建立端點。如果環境卡在 中PENDING，這可能表示私有 Web 伺服器端點遺失。

收到**The Vpc Endpoint Service '*vpce-service-name*' does not exist**錯誤

如果您參考下列錯誤，請確認在擁有共用 VPC 的帳戶中建立端點Owner的帳戶：

```
ClientError: An error occurred (InvalidServiceName) when calling the
 CreateVpcEndpoint operation:

The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Amazon Managed Workflows for Apache Airflow 教學課程

本指南包含使用和設定 Amazon Managed Workflows for Apache Airflow 環境的step-by-step教學課程。

主題

- 教學課程：使用 設定私有網路存取 AWS Client VPN
- 教學課程：使用 Linux 堡壘主機設定私有網路存取
- 教學課程：限制 Amazon MWAA 使用者存取 DAGs的子集
- 教學課程：在 Amazon MWAA 上自動管理您自己的環境端點

教學課程：使用 設定私有網路存取 AWS Client VPN

本教學課程會逐步解說從電腦建立 VPN 通道到 Amazon Managed Workflows for Apache Airflow 環境的 Apache Airflow Web 伺服器的步驟。若要透過 VPN 通道連線至網際網路，您必須先建立 AWS Client VPN 端點。設定完成後，Client VPN 端點會充當 VPN 伺服器，允許從您的電腦安全連線至 VPC 中的資源。然後，您將使用 AWS Client VPN 桌面版從電腦連線至 Client VPN。

章節

- 私有網路
- 使用案例
- 開始之前
- 目標
- （選用）步驟一：識別您的 VPC、CIDR 規則和 VPC 安全性
- 步驟二：建立伺服器和用戶端憑證
- 步驟三：在本機儲存 CloudFormation 範本
- 步驟四：建立 Client VPN CloudFormation 堆疊
- 步驟五：將子網路與 Client VPN 建立關聯
- 步驟六：將授權輸入規則新增至 Client VPN
- 步驟七：下載 Client VPN 端點組態檔案
- 步驟八：連線至 AWS Client VPN
- 後續步驟？

私有網路

本教學假設您已為 Apache Airflow Web 伺服器選擇私有網路存取模式。

私有網路存取模式會將對 Apache Airflow UI 的存取限制在 Amazon VPC 中已獲授予環境 IAM 政策存取權的使用者。

當您建立具有私有 Web 伺服器存取權的環境時，您必須在 Python wheel 封存檔 (`.whl`) 中封裝所有相依性，然後在 `.whl`中參考 `requirements.txt`。如需使用 wheel 封裝和安裝相依性的說明，請參閱使用 Python wheel 管理相依性。

下圖說明在 Amazon MWAA 主控台上尋找私有網路選項的位置。

使用案例

您可以在建立 Amazon MWAA 環境之前或之後使用此教學課程。您必須使用與環境相同的 Amazon VPC、VPC 安全群組和私有子網路。如果您在建立 Amazon MWAA 環境後使用此教學課程，一旦完成這些步驟，您就可以返回 Amazon MWAA 主控台，並將 Apache Airflow Web 伺服器存取模式變更為私有網路。

開始之前

1. 檢查使用者許可。請確定您在 AWS Identity and Access Management (IAM) 中的帳戶有足夠的許可來建立和管理 VPC 資源。
2. 使用您的 Amazon MWAA VPC。本教學假設您正在將 Client VPN 與現有 VPC 建立關聯。Amazon VPC 必須位於與 Amazon MWAA 環境 AWS 區域 相同的 中，並有兩個私有子網路。如果您尚未建立 Amazon VPC，請在 中使用 CloudFormation 範本選項三：在沒有網際網路存取的情況下建立 Amazon VPC 網路。

目標

在本教學中，您將執行下列作業：

1. 使用現有 Amazon VPC 的 CloudFormation 範本建立 AWS Client VPN 端點。
2. 產生伺服器和用戶端憑證和金鑰，然後在 AWS 區域 與 Amazon MWAA 環境相同的 AWS Certificate Manager 中將伺服器憑證和金鑰上傳至 。

3. 下載和修改 Client VPN 的 Client VPN 端點組態檔案，並使用 檔案建立 VPN 設定檔，以使用 Client VPN for Desktop 進行連線。

（選用）步驟一：識別您的 VPC、CIDR 規則和 VPC 安全性

下一節說明如何尋找 Amazon VPC、VPC 安全群組IDs，以及如何識別在後續步驟中建立 Client VPN 所需的 CIDR 規則。

識別您的 CIDR 規則

下一節說明如何識別您需要建立 Client VPN 的 CIDR 規則。

識別 Client VPN 的 CIDR

1. 在 Amazon VPCs 主控台上開啟您的 Amazon VPC 頁面。
2. 使用導覽列中的區域選擇器，選擇 AWS 區域 與 Amazon MWAA 環境相同的 。
3. 選擇您的 Amazon VPC。
4. 假設私有子網路CIDRs 為：

   - 私有子網路 1：10.192.10.0/24
   - 私有子網路 2：10.192.11.0/24

   如果 Amazon VPC 的 CIDR 為 10.192.0.0/16，則您為 Client VPN 指定的用戶端 IPv4 CIDR 將為 10.192.0/22.0。
5. 儲存此 CIDR 值，以及 VPC ID 的值以供後續步驟使用。

識別您的 VPC 和安全群組

下一節說明如何尋找 Amazon VPC 和安全群組的 ID，您需要這些 ID 才能建立 Client VPN。

> **Note**
> 您可能正在使用多個安全群組。您需要在後續步驟中指定 VPC 的所有安全群組。

識別安全群組

1. 在 Amazon VPC 主控台上開啟安全群組頁面。

2. 使用導覽列中的區域選擇器來選擇 AWS 區域。
3. 在 VPC ID 中搜尋 Amazon VPC，並識別與 VPC 相關聯的安全群組。
4. 儲存安全群組和 VPC 的 ID 以進行後續步驟。

步驟二：建立伺服器和用戶端憑證

Client VPN 端點僅支援 1024 位元和 2048 位元 RSA 金鑰大小。下一節說明如何使用 OpenVPN easy-rsa 產生伺服器和用戶端憑證和金鑰，然後使用 AWS Command Line Interface () 將憑證上傳至 ACM AWS CLI。

建立用戶端憑證

1. 請依照這些快速步驟，透過用戶端身分驗證和授權：相互身分驗證 AWS CLI 中的 ，建立憑證並將其上傳至 ACM。
2. 在這些步驟中，上傳伺服器和用戶端憑證時，您必須在 AWS CLI 命令中指定 AWS 區域 與 Amazon MWAA 環境相同的 。以下是如何在這些命令中指定區域的一些範例：

   a. Example伺服器憑證的區域

   ```
   aws acm import-certificate --certificate fileb://server.crt --private-key
    fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
   ```

   b. Example用戶端憑證的區域

   ```
   aws acm import-certificate --certificate fileb://client1.domain.tld.crt
    --private-key fileb://client1.domain.tld.key --certificate-chain fileb://
   ca.crt --region us-west-2
   ```

   c. 完成這些步驟後，請儲存伺服器憑證和用戶端憑證 ARNs AWS CLI 回應中傳回的值。您將在 CloudFormation 範本中指定這些 ARNs以建立 Client VPN。
3. 在這些步驟中，用戶端憑證和私有金鑰會儲存至您的電腦。以下是在何處尋找這些登入資料的範例：

   a. Example在 macOS 上

   在 macOS 上，內容會儲存在 。 `/Users/your-user/custom_folder`如果您列出此目錄的所有 (`ls -a`) 內容，您會得到類似以下內容的內容：

   ```
   .
   ```

```
..
ca.crt
client1.domain.tld.crt
client1.domain.tld.key
server.crt
server.key
```

b. 完成這些步驟後，請儲存內容或記下 中用戶端憑證的位置`client1.domain.tld.crt`，以及 中的私有金鑰`client1.domain.tld.key`。您將將這些值新增至 Client VPN 的組態檔案。

步驟三：在本機儲存 CloudFormation 範本

下一節包含建立 Client VPN 的 CloudFormation 範本。您必須指定與 Amazon MWAA 環境相同的 Amazon VPC、VPC 安全群組和私有子網路。

- 複製下列範本的內容，並在本機儲存為 `mwaa_vpn_client.yaml`。您也可以下載 範本。

  取代下列值：

  - **YOUR_CLIENT_ROOT_CERTIFICATE_ARN** – 中 client1.domain.tld 憑證的 ARN`ClientRootCertificateChainArn`。
  - **YOUR_SERVER_CERTIFICATE_ARN** – 中伺服器憑證的 ARN`ServerCertificateArn`。
  - 中的用戶端 IPv4 CIDR 規則`ClientCidrBlock`。`10.192.0.0/22` 提供 的 CIDR 規則。
  - 您的 Amazon VPC ID`VpcId`。`vpc-010101010101` 提供 的 VPC。
  - 您在 中的 VPC 安全群組 ID`SecurityGroupIds`。`sg-0101010101` 提供 的安全群組。

```
AWSTemplateFormatVersion: 2010-09-09
Description: This template deploys a VPN Client Endpoint.
Resources:
  ClientVpnEndpoint:
    Type: 'AWS::EC2::ClientVpnEndpoint'
    Properties:
      AuthenticationOptions:
        - Type: "certificate-authentication"
          MutualAuthentication:
            ClientRootCertificateChainArn: "YOUR_CLIENT_ROOT_CERTIFICATE_ARN"
      ClientCidrBlock: 10.192.0.0/22
      ClientConnectOptions:
```

```
    Enabled: false
  ConnectionLogOptions:
    Enabled: false
  Description: "MWAA Client VPN"
  DnsServers: []
  SecurityGroupIds:
    - sg-0101010101
  SelfServicePortal: ''
  ServerCertificateArn: "YOUR_SERVER_CERTIFICATE_ARN"
  SplitTunnel: true
  TagSpecifications:
    - ResourceType: "client-vpn-endpoint"
      Tags:
      - Key: Name
        Value: MWAA-Client-VPN
  TransportProtocol: udp
  VpcId: vpc-010101010101
  VpnPort: 443
```

Note

如果您為環境使用多個安全群組，您可以使用下列格式指定多個安全群組：

```
SecurityGroupIds:
 - sg-0112233445566778b
 - sg-0223344556677889f
```

步驟四：建立 Client VPN CloudFormation 堆疊

建立 AWS Client VPN

1. 開啟 AWS CloudFormation 主控台。
2. 選擇範本已就緒，上傳範本檔案。
3. 選擇選擇檔案，然後選取您的`mwaa_vpn_client.yaml`檔案。
4. 選擇下一步、下一步。
5. 選取確認，然後選擇建立堆疊。

步驟五：將子網路與 Client VPN 建立關聯

將私有子網路與 建立關聯 AWS Client VPN

1. 開啟 Amazon VPC 主控台。
2. 選擇 Client VPN 端點頁面。
3. 選取您的 Client VPN，然後選擇關聯索引標籤：關聯。
4. 在下拉式清單中選擇下列項目：
   - VPC 中的 Amazon VPC。
   - 中其中一個私有子網路 選擇要關聯的子網路。
5. 選擇關聯。

> **Note**
>
> VPC 和子網路需要幾分鐘的時間才能與 Client VPN 建立關聯。

步驟六：將授權輸入規則新增至 Client VPN

您需要使用 VPC 的 CIDR 規則將授權輸入規則新增至 Client VPN。如果您想要從 Active Directory 群組或 SAML 型身分提供者 (IdP) 授權特定使用者或群組，請參閱 Client VPN 指南中的授權規則。

將 CIDR 新增至 AWS Client VPN

1. 開啟 Amazon VPC 主控台。
2. 選擇 Client VPN 端點頁面。
3. 選取您的 Client VPN，然後選擇授權輸入索引標籤。
4. 指定下列內容：
   - 要啟用之目的地網路中的 Amazon VPC CIDR 規則。例如：

     ```
     10.192.0.0/16
     ```
   - 選擇允許授予存取權的所有使用者存取。
   - 在描述中輸入描述名稱。

5. 選擇新增授權規則。

> **Note**
>
> 視 Amazon VPC 的網路元件而定，您可能還需要此授權傳入規則到您的網路存取控制清單 (NACL)。

步驟七：下載 Client VPN 端點組態檔案

下載組態檔

1. 請依照這些快速步驟，在下載 Client VPN 端點組態檔案下載 Client VPN 組態檔案。
2. 在這些步驟中，系統會要求您在 Client VPN 端點 DNS 名稱前面加上字串。範例如下：

   - Example端點 DNS 名稱

     如果您的 Client VPN 端點 DNS 名稱為：

     ```
     remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
     ```

     您可以新增字串來識別 Client VPN 端點，如下所示：

     ```
     remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-
     west-1.amazonaws.com 443
     ```

3. 在這些步驟中，系統會要求您在一組新的<cert></cert>標籤之間新增用戶端憑證的內容，以及在一組新的<key></key>標籤之間新增私有金鑰的內容。範例如下：

   a. 開啟命令提示字元，並將目錄變更為用戶端憑證和私有金鑰的位置。

   b. Example macOS client1.domain.tld.crt

      若要在 macOS 上顯示client1.domain.tld.crt檔案的內容，您可以使用 cat client1.domain.tld.crt。

      從終端機複製值並貼上，downloaded-client-config.ovpn如下所示：

      ```
      ZZZ1111dddaBBB
      -----END CERTIFICATE-----
      ```

```
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

c. Example macOS client1.domain.tld.key

若要顯示 的內容`client1.domain.tld.key`，您可以使用 `cat client1.domain.tld.key`。

從終端機複製值並貼上，`downloaded-client-config.ovpn`如下所示：

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.key
-----END CERTIFICATE-----
</key>
```

步驟八：連線至 AWS Client VPN

的用戶端 AWS Client VPN 是免費的。您可以直接將電腦連接到 ， AWS Client VPN end-to-end體驗。

連線至 Client VPN

1. 下載並安裝 AWS Client VPN 桌面版 。
2. 開啟 AWS Client VPN。
3. 在 VPN 用戶端功能表中選擇檔案、受管設定檔。
4. 選擇新增設定檔，然後選擇 `downloaded-client-config.ovpn`。
5. 在顯示名稱中輸入描述性名稱。

6. 選擇新增設定檔、完成。
7. 選擇連線。

連線至 Client VPN 後，您需要中斷與其他 VPNs連線，才能存取 Amazon VPC 中的任何資源。

> **Note**
>
> 您可能需要結束用戶端，並在能夠連線之前重新開始。

後續步驟？

- 了解如何在 中建立 Amazon MWAA 環境開始使用 Amazon Managed Workflows for Apache Airflow。您必須在與 Client VPN AWS 區域 相同的 中建立環境，並使用與 Client VPN 相同的 VPC、私有子網路和安全群組。

教學課程：使用 Linux 堡壘主機設定私有網路存取

本教學課程將逐步引導您建立 SSH 通道，從您的電腦到 Amazon Managed Workflows for Apache Airflow 環境的 Apache Airflow Web 伺服器。其假設您已建立 Amazon MWAA 環境。設定完成後，Linux 堡壘主機會充當跳轉伺服器，允許從您的電腦安全連線至 VPC 中的資源。然後，您將使用 SOCKS 代理管理附加元件來控制瀏覽器中的代理設定，以存取您的 Apache Airflow UI。

章節

- 私有網路
- 使用案例
- 開始之前
- 目標
- 步驟一：建立堡壘執行個體
- 步驟二：建立 SSH 通道
- 步驟三：將堡壘安全群組設定為傳入規則
- 步驟四：複製 Apache Airflow URL
- 步驟五：設定代理設定
- 步驟六：開啟 Apache Airflow UI

- 後續步驟？

私有網路

本教學假設您已為 Apache Airflow Web 伺服器選擇私有網路存取模式。

私有網路存取模式會將對 Apache Airflow UI 的存取限制在 Amazon VPC 中已獲授予環境 IAM 政策存取權的使用者。

當您建立具有私有 Web 伺服器存取權的環境時，您必須在 Python wheel 封存檔 (`.whl`) 中封裝所有相依性，然後在 `.whl`中參考 `requirements.txt`。如需使用 wheel 封裝和安裝相依性的說明，請參閱使用 Python wheel 管理相依性。

下圖說明在 Amazon MWAA 主控台上尋找私有網路選項的位置。

使用案例

您可以在建立 Amazon MWAA 環境後使用此教學課程。您必須使用與環境相同的 Amazon VPC、VPC 安全群組和公有子網路。

開始之前

1. 檢查使用者許可。請確定您在 AWS Identity and Access Management (IAM) 中的帳戶有足夠的許可來建立和管理 VPC 資源。
2. 使用您的 Amazon MWAA VPC。本教學假設您正在將堡壘主機與現有 VPC 建立關聯。Amazon VPC 必須與 Amazon MWAA 環境位於相同的區域，並具有兩個私有子網路，如 中所定義建立 VPC 網路。
3. 建立 SSH 金鑰。您需要在與 Amazon MWAA 環境相同的區域中建立 Amazon EC2 SSH 金鑰 (.pem)，才能連線至虛擬伺服器。如果您沒有 SSH 金鑰，請參閱《Amazon EC2 使用者指南》中的建立或匯入金鑰對。

目標

在本教學中，您將執行下列作業：

1. 使用CloudFormation 現有 VPC 的範本建立 Linux 堡壘主機執行個體。

2. 使用連接埠 上的輸入規則，將傳入流量授權至堡壘執行個體的安全群組22。
3. 授權從 Amazon MWAA 環境安全群組到堡壘執行個體安全群組的傳入流量。
4. 建立堡壘執行個體的 SSH 通道。
5. 安裝並設定 Firefox 瀏覽器的 FoxyProxy 附加元件，以存取 Apache Airflow UI。

步驟一：建立堡壘執行個體

下一節說明使用 CloudFormation 主控台上CloudFormation 現有 VPC 的範本建立 linux 堡壘執行個體的步驟。

建立 Linux 堡壘主機

1. 在 CloudFormation 主控台上開啟部署 Quick Start 頁面。
2. 使用導覽列中的區域選擇器，選擇與您的 Amazon MWAA 環境 AWS 區域 相同的 。
3. 選擇下一步。
4. 在堆疊名稱文字欄位中輸入名稱，例如 `mwaa-linux-bastion`。
5. 在參數、網路組態窗格中，選擇下列選項：
    a. 選擇 Amazon MWAA 環境的 VPC ID。
    b. 選擇 Amazon MWAA 環境的公有子網路 1 ID。
    c. 選擇 Amazon MWAA 環境的公有子網路 2 ID。
    d. 在允許堡壘外部存取 CIDR 中輸入最窄的可能地址範圍 （例如，內部 CIDR 範圍）。

    > **Note**
    > 識別範圍的最簡單方法是使用與公有子網路相同的 CIDR 範圍。例如， 建立 VPC 網路頁面上 CloudFormation 範本中的公有子網路為 `10.192.10.0/24`和 `10.192.11.0/24`。

6. 在 Amazon EC2 組態窗格中，選擇下列項目：
    a. 在金鑰對名稱的下拉式清單中選擇您的 SSH 金鑰。
    b. 在堡壘主機名稱中輸入名稱。
    c. 針對 TCP 轉送選擇 true。

> **Warning**
>
> 在此步驟中，TCP 轉送必須設為 true。否則，您將無法在下一個步驟中建立 SSH 通道。

7. 選擇下一步、下一步。
8. 選取確認，然後選擇建立堆疊。

若要進一步了解 Linux 堡壘主機的架構，請參閱 AWS 雲端上的 Linux 堡壘主機： 架構。

步驟二：建立 SSH 通道

下列步驟說明如何為 linux 堡壘建立 ssh 通道。SSH 通道會將請求從本機 IP 地址接收到 linux 堡壘，這就是為什麼在先前的步驟`true`中將 linux 堡壘的 TCP 轉送設定為 的原因。

macOS/Linux

使用命令列建立通道

1. 在 Amazon EC2 主控台上開啟執行個體頁面。
2. 選擇執行個體。
3. 在公有 IPv4 DNS 中複製地址。例如 `ec2-4-82-142-1.compute-1.amazonaws.com`。
4. 在命令提示中，導覽至存放 SSH 金鑰的目錄。
5. 執行下列命令，使用 ssh 連線至堡壘執行個體。在 中使用 SSH 金鑰名稱取代範例值`mykeypair.pem`。

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

使用 PuTTY 建立通道

1. 在 Amazon EC2 主控台上開啟執行個體頁面。
2. 選擇執行個體。
3. 在公有 IPv4 DNS 中複製地址。例如 `ec2-4-82-142-1.compute-1.amazonaws.com`。

4. 開啟 PuTTY，選取工作階段。
5. 在主機名稱中以 ec2-user@*YOUR_PUBLIC_IPV4_DNS* 輸入主機名稱，並以 輸入連接埠22。
6. 展開 SSH 索引標籤，選取驗證。在用於身分驗證的私有金鑰檔案中，選擇您的本機「ppk」檔案。
7. 在 SSH 下，選擇通道索引標籤，然後選取動態和自動選項。
8. 在來源連接埠中，新增8157連接埠（或任何其他未使用的連接埠），然後將目的地連接埠保留空白。選擇新增。
9. 選擇工作階段索引標籤，然後輸入工作階段名稱。例如 `SSH Tunnel`。
10. 選擇儲存、開啟。

> **Note**
>
> 您可能需要為公有金鑰輸入密碼短語。

> **Note**
>
> 如果您收到`Permission denied (publickey)`錯誤，建議您使用 AWSSupport-TroubleshootSSH 工具，然後選擇執行此自動化（主控台）來疑難排解 SSH 設定。

步驟三：將堡壘安全群組設定為傳入規則

允許從伺服器存取伺服器和定期網際網路，並將特殊維護安全群組連接到這些伺服器。下列步驟說明如何將堡壘安全群組設定為環境 VPC 安全群組的傳入流量來源。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在聯網窗格中，選擇 VPC 安全群組。
4. 選擇 Edit inbound Rules (編輯傳入規則)。
5. 選擇新增規則。
6. 在來源下拉式清單中選擇您的 VPC 安全群組 ID。
7. 將其餘選項保留空白，或設定為其預設值。
8. 選擇儲存規則。

步驟四：複製 Apache Airflow URL

下列步驟說明如何開啟 Amazon MWAA 主控台，並將 URL 複製到 Apache Airflow UI。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在 Airflow UI 中複製 URL 以進行後續步驟。

步驟五：設定代理設定

如果您使用 SSH 通道搭配動態連接埠轉送，您必須使用 SOCKS 代理管理附加元件，以控制在瀏覽器中的代理設定。例如，您可以使用 Chromium `--proxy-server`的功能來啟動瀏覽器工作階段，或在 Mozilla FireFox 瀏覽器中使用 FoxyProxy 延伸模組。

選項一：使用本機連接埠轉送設定 SSH 通道

如果您不想使用 SOCKS 代理，您可以使用本機連接埠轉送來設定 SSH 通道。下列範例命令透過轉送本機連接埠 8157 上的流量來存取 Amazon EC2 ResourceManager Web 介面。

1. 開啟新的命令提示視窗。
2. 輸入下列命令以開啟 SSH 通道。

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-vpce.us-
east-1.airflow.amazonaws.com:443 ubuntu@YOUR_PUBLIC_IPV4_DNS.us-
east-1.compute.amazonaws.com
```

   -L 表示使用本機連接埠轉送，您可以使用它來指定本機連接埠，用來將資料轉送到節點本機 Web 伺服器上已識別的遠端連接埠。

3. 在瀏覽器`http://localhost:8157/`中輸入 。

> **Note**
>
> 您可能需要使用 `https://localhost:8157/`。

選項二：使用命令列的代理

您可以使用大多數 Web 瀏覽器，使用命令列或組態參數來設定代理。例如，使用 Chromium，您可以使用下列命令啟動瀏覽器：

```
chromium --proxy-server="socks5://localhost:8157"
```

這會啟動瀏覽器工作階段，使用您在先前步驟中建立的 ssh 通道來代理其請求。您可以開啟私有 Amazon MWAA 環境 URL（使用 https：//），如下所示：

```
https://YOUR_VPC_ENDPOINT_ID-vpce.us-east-1.airflow.amazonaws.com/home.
```

選項三：使用 FoxyProxy for Mozilla Firefox 的代理

下面的範例演示了 Mozilla Firefox 的 FoxyProxy Standard (7.5.1 版) 組態。FoxyProxy 提供一組代理管理工具。它可讓您將代理伺服器用於符合對應於 Apache Airflow UI 所用網域之模式的 URLs。

1. 在 Firefox 中，開啟 FoxyProxy 標準擴充功能頁面。
2. 選擇新增至 Firefox。
3. 選擇新增。
4. 選擇瀏覽器工具列中的 FoxyProxy 圖示，然後選擇選項。
5. 複製下列程式碼，並在本機儲存為 `mwaa-proxy.json`。使用 Apache Airflow URL 取代 *YOUR_HOST_NAME* 中的範例值。

```
{
  "e0b7kh1606694837384": {
    "type": 3,
    "color": "#66cc66",
    "title": "airflow",
    "active": true,
    "address": "localhost",
    "port": 8157,
    "proxyDNS": false,
    "username": "",
    "password": "",
    "whitePatterns": [
      {
        "title": "airflow-ui",
        "pattern": "YOUR_HOST_NAME",
        "type": 1,
```

```
        "protocols": 1,
        "active": true
      }
    ],
    "blackPatterns": [],
    "pacURL": "",
    "index": -1
  },
  "k20d21508277536715": {
    "active": true,
    "title": "Default",
    "notes": "These are the settings that are used when no patterns match a URL.",
    "color": "#0055E5",
    "type": 5,
    "whitePatterns": [
      {
        "title": "all URLs",
        "active": true,
        "pattern": "*",
        "type": 1,
        "protocols": 1
      }
    ],
    "blackPatterns": [],
      "index": 9007199254740991
  },
  "logging": {
    "active": true,
    "maxSize": 500
  },
  "mode": "patterns",
  "browserVersion": "82.0.3",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

6. 在從 FoxyProxy 6.0+ 匯入設定窗格中，選擇匯入設定並選取`mwaa-proxy.json`檔案。
7. 選擇確定。

步驟六：開啟 Apache Airflow UI

下列步驟說明如何開啟 Apache Airflow UI。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇開啟氣流使用者介面。

後續步驟？

- 了解如何在 中對堡壘主機的 SSH 通道上執行 Airflow CLI 命令Apache Airflow CLI 命令參考。
- 了解如何將 DAG 程式碼上傳至 中的 Amazon S3 儲存貯體新增或更新 DAGs。

教學課程：限制 Amazon MWAA 使用者存取 DAGs的子集

Amazon MWAA 會將您的 IAM 主體映射至一或多個 Apache Airflow 的預設角色，以管理對您環境的存取。使用下列教學課程，限制個別 Amazon MWAA 使用者只能存取特定 DAG 或一組 DAGs 並與之互動。

> **Note**
>
> 只要可以擔任 IAM 角色，即可使用聯合存取完成本教學中的步驟。

主題

- 先決條件
- 步驟一：使用預設 Public Apache Airflow 角色將 Amazon MWAA Webserver 存取權提供給 IAM 主體。
- 步驟二：建立新的 Apache Airflow 自訂角色
- 步驟三：將您建立的角色指派給 Amazon MWAA 使用者
- 後續步驟
- 相關資源

先決條件

若要完成本教學課程中的步驟，您需要下列項目：

- 具有多個 DAGs Amazon MWAA 環境
- Admin 具有 AdministratorAccess 許可的 IAM 主體，以及 IAM 使用者 MWAAUser，作為您可以限制 DAG 存取的主體。如需管理員角色的詳細資訊，請參閱《IAM 使用者指南》中的管理員任務函數

> **Note**
>
> 請勿將許可政策直接連接至您的 IAM 使用者。我們建議您設定使用者可以擔任的 IAM 角色，以取得 Amazon MWAA 資源的暫時存取權。

- 第 AWS Command Line Interface 2 版已安裝。

步驟一：使用預設 **Public** Apache Airflow 角色將 Amazon MWAA Webserver 存取權提供給 IAM 主體。

使用 授予許可 AWS 管理主控台

1. AWS 帳戶 使用 Admin角色登入您的 ，並開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇使用者，然後從使用者資料表中選擇您的 Amazon MWAA IAM 使用者。
3. 在使用者詳細資訊頁面的摘要下，選擇許可索引標籤，然後選擇許可政策以展開卡片，然後選擇新增許可。
4. 在授予許可區段中，選擇直接連接現有政策，然後選擇建立政策以建立和連接您自己的自訂許可政策。
5. 在建立政策頁面上，選擇 JSON，然後在政策編輯器中複製並貼上下列 JSON 許可政策。Tha 政策會使用預設 Public Apache Airflow 角色將 Web 伺服器存取權授予使用者。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
            "arn:aws:airflow:us-
east-1:111122223333:role/YOUR_ENVIRONMENT_NAME/Public"
            ]
        }
    ]
}
```

步驟二：建立新的 Apache Airflow 自訂角色

使用 Apache Airflow UI 建立新角色

1. 使用您的管理員 IAM 角色，開啟 Amazon MWAA 主控台並啟動您環境的 Apache Airflow UI。
2. 從頂端的導覽窗格中，將滑鼠游標暫留在安全性上以開啟下拉式清單，然後選擇列出角色以存取預設 Apache Airflow 角色。
3. 從角色清單中，選取使用者，然後在頁面開頭選擇動作以開啟下拉式清單。選擇複製角色，並確認確定

   > **Note**
   >
   > 複製 Ops 或 Viewer 角色，分別授予更多或更少的存取權。

4. 找到您在資料表中建立的新角色，然後選擇編輯記錄。
5. 在編輯角色頁面上，執行下列動作：

   - 針對名稱，在文字欄位中輸入角色的新名稱。例如 **Restricted**。
   - 如需許可清單，請移除 `can read on DAGs`和 `can edit on DAGs`，然後為您要提供存取權的一組 DAGs 新增讀取和寫入許可。例如，對於 DAG、`example_dag.py`、新增 **can read on DAG:*example_dag*** 和 **can edit on DAG:*example_dag***。

   選擇儲存。現在，您有一個新角色，限制對 Amazon MWAA 環境中可用 DAGs 子集的存取。您可以將此角色指派給任何現有的 Apache Airflow 使用者。

步驟三：將您建立的角色指派給 Amazon MWAA 使用者

指派新角色

1. 使用 的存取憑證`MWAAUser`，執行下列 CLI 命令來擷取您環境的 Web 伺服器 URL。

   ```
   aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq
    '.Environment.WebserverUrl'
   ```

   如果成功，您將參考下列輸出：

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. MWAAUser 登入 後 AWS 管理主控台，開啟新的瀏覽器視窗並存取下列 URI。將 取代Webserver-URL為您的資訊。

```
https://<Webserver-URL>/home
```

如果成功，您會收到Forbidden錯誤頁面，因為 MWAAUser 尚未獲得存取 Apache Airflow UI 的許可。

3. Admin 登入 後 AWS 管理主控台，再次開啟 Amazon MWAA 主控台，並啟動您環境的 Apache Airflow UI。
4. 從 UI 儀表板中，展開安全性下拉式清單，這次選擇列出使用者。
5. 在使用者表格中，尋找新的 Apache Airflow 使用者，然後選擇編輯記錄。使用者的名字將符合以下模式的 IAM 使用者名稱：user/*mwaa-user*。
6. 在編輯使用者頁面上的角色區段中，新增您建立的新自訂角色，然後選擇儲存。

> **Note**
>
> 姓氏欄位為必要欄位，但空格符合需求。

IAM Public主體授予存取 Apache Airflow UI 的MWAAUser許可，而新角色則提供取得其 DAGs所需的額外許可。

> **Important**
>
> IAM 未授權且使用 Apache Airflow UI 新增的 5 個預設角色 （ 例如 Admin)，都會在下次使用者登入時移除。

後續步驟

- 若要進一步了解如何管理 Amazon MWAA 環境的存取權，以及取得可供環境使用者使用的 JSON IAM 政策範例，請參閱 the section called “存取 Amazon MWAA 環境”

相關資源

- 存取控制 (Apache Airflow 文件） – 進一步了解 Apache Airflow 文件網站上的預設 Apache Airflow 角色。

教學課程：在 Amazon MWAA 上自動管理您自己的環境端點

如果您使用 AWS Organizations管理多個 AWS 帳戶 共用資源，Amazon MWAA 可讓您建立和管理自己的 Amazon VPC 端點。這表示您可以使用更嚴格的安全政策，僅允許存取您環境所需的資源。

當您在共用的 Amazon VPC 中建立環境時，擁有主要 Amazon VPC (擁有者) 的帳戶會與屬於相同組織的其他帳戶 (參與者) 共用 Amazon MWAA 所需的兩個私有子網路。然後，共用這些子網路的參與者帳戶可以檢視、建立、修改和刪除共用 VPC 中的環境。

當您在共用或其他政策限制的 Amazon VPC 中建立環境時，Amazon MWAA 會先建立服務 VPC 資源，然後輸入`PENDING`狀態長達 72 小時。

當環境狀態從 變更為 `CREATING` 時`PENDING`，Amazon MWAA 會傳送狀態變更的 Amazon EventBridge 通知。這可讓擁有者帳戶根據來自 Amazon MWAA 主控台或 API 的端點服務資訊，或以程式設計方式，代表參與者建立必要的端點。在下文中，我們使用 Lambda 函數和 EventBridge 規則來建立新的 Amazon VPC 端點，以接聽 Amazon MWAA 狀態變更通知。

在這裡，我們會在與環境相同的 Amazon VPC 中建立新的端點。若要設定共用的 Amazon VPC，請在擁有者帳戶中建立 EventBridge 規則和 Lambda 函數，並在參與者帳戶中建立 Amazon MWAA 環境。

主題

- 先決條件
- 建立 Amazon VPC
- 建立 Lambda 函式
- 建立 EventBridge 規則
- 建立 Amazon MWAA 環境

先決條件

若要完成本教學課程中的步驟，您將需要下列項目：

- ...

建立 Amazon VPC

使用下列 CloudFormation 範本和 AWS CLI 命令建立新的 Amazon VPC。範本會設定 Amazon VPC 資源，並修改端點政策以限制對特定佇列的存取。

1. 下載 CloudFormation 範本，然後解壓縮.yml檔案。
2. 在新的命令提示字元視窗中，導覽至您儲存範本的資料夾，然後使用 create-stack建立堆疊。--template-body 旗標會指定範本的路徑。

```
aws cloudformation create-stack --stack-name stack-name --template-body file://cfn-vpc-private-network.yml
```

在下一節中，您將建立 Lambda 函數。

建立 Lambda 函式

使用下列 Python 程式碼和 IAM JSON 政策建立新的 Lambda 函數和執行角色。此函數會為私有 Apache Airflow Web 伺服器和 Amazon SQS 佇列建立 Amazon VPC 端點。在擴展您的環境時，Amazon MWAA 使用 Amazon SQS 在多個工作者之間使用 Celery 將任務排入佇列。

1. 下載 Python 函數程式碼。
2. 下載 IAM 許可政策，然後解壓縮 檔案。
3. 開啟命令提示，然後導覽至您儲存 JSON 許可政策的資料夾。使用 IAM create-role命令來建立新的角色。

```
aws iam create-role --role-name function-role \
      --assume-role-policy-document file://lambda-mwaa-vpce-policy.json
```

請注意 AWS CLI 回應中的角色 ARN。在下一個步驟中，我們使用函數的 ARN 將此新角色指定為函數的執行角色。

4. 導覽至您儲存函數程式碼的資料夾，然後使用 create-function命令建立新的函數。

```
aws lambda create-function --function-name mwaa-vpce-lambda \
--zip-file file://mwaa-lambda-shared-vpc.zip --runtime python3.8 --role
 arn:aws:iam::123456789012:role/function-role --handler lambda_handler
```

請注意 AWS CLI 回應中的函數 ARN。在下一個步驟中，我們會指定 ARN，將函數設定為新 EventBridge 規則的目標。

在下一節中，您會建立 EventBridge 規則，在環境進入 PENDING 狀態時叫用此函數。

建立 EventBridge 規則

執行下列動作來建立新的規則，以接聽 Amazon MWAA 通知並鎖定新的 Lambda 函數。

1. 使用 EventBridge `put-rule`命令建立新的 EventBridge 規則。

```
aws events put-rule --name "mwaa-lambda-rule" \
--event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment
 Status Change\"]}"
```

事件模式會接聽 Amazon MWAA 每當環境狀態變更時傳送的通知。

```
{
     "source": ["aws.airflow"],
     "detail-type": ["MWAA Environment Status Change"]
     }
```

2. 使用 `put-targets`命令將 Lambda 函數新增為新規則的目標。

```
aws events put-targets --rule "mwaa-lambda-rule" \
--targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:mwaa-vpce-
lambda"
```

您已準備好使用客戶管理的 Amazon VPC 端點建立新的 Amazon MWAA 環境。

建立 Amazon MWAA 環境

使用 Amazon MWAA 主控台建立具有客戶受管 Amazon VPC 端點的新環境。

1. 開啟 Amazon MWAA 主控台，然後選擇建立環境。
2. 針對名稱輸入唯一的名稱。
3. 針對 Airflow 版本，選擇最新版本。
4. 選擇 Amazon S3 儲存貯體和 DAGs，例如`dags/`用於環境，然後選擇下一步。

5. 在設定進階設定頁面上，執行下列動作：
   a. 針對虛擬私有雲端，選擇您在上一個步驟中建立的 Amazon VPC。
   b. 針對 Web 伺服器存取，選擇公有網路 （網際網路可存取）。
   c. 針對安全群組，選擇您建立的安全群組 CloudFormation。由於先前步驟中 AWS PrivateLink 端點的安全群組是自我參考的，因此您必須為您的環境選擇相同的安全群組。
   d. 針對端點管理，選擇客戶受管端點。
6. 保留剩餘的預設設定，然後選擇下一步。
7. 檢閱您的選擇，然後選擇建立環境。

> **Tip**
>
> 如需設定新環境的詳細資訊，請參閱 Amazon MWAA 入門。

當環境為 時PENDING，Amazon MWAA 會傳送符合您為規則設定之事件模式的通知。此規則會叫用您的 Lambda 函數。函數會剖析通知事件，並取得 Web 伺服器和 Amazon SQS 佇列所需的端點資訊。然後，它會在您的 Amazon VPC 中建立端點。

當端點可用時，Amazon MWAA 會繼續建立您的環境。準備就緒時，環境狀態會變更為 ，AVAILABLE您可以使用 Amazon MWAA 主控台存取 Apache Airflow Web 伺服器。

Amazon Managed Workflows for Apache Airflow 的程式碼範例

本指南包含程式碼範例，包括 DAGs和自訂外掛程式，您可以在 Amazon Managed Workflows for Apache Airflow 環境中使用。如需搭配 AWS 服務使用 Apache Airflow 的更多範例，請參閱 Apache Airflow GitHub 儲存庫中的 `dags`目錄。

範例

- 使用 DAG 在 CLI 中匯入變數
- 使用 建立 SSH 連線 SSHOperator
- 在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow Snowflake 連線
- 使用 DAG 在 CloudWatch 中寫入自訂指標
- Amazon MWAA 環境上的 Aurora PostgreSQL 資料庫清除
- 將環境中繼資料匯出至 Amazon S3 上的 CSV 檔案
- 針對 AWS Secrets Manager Apache Airflow 變數在 中使用私密金鑰
- 在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線
- 使用 Oracle 建立自訂外掛程式
- 在 Amazon MWAA 上變更 DAG 的時區
- 重新整理 CodeArtifact 權杖
- 使用 Apache Hive 和 Hadoop 建立自訂外掛程式
- 為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式
- 使用 Lambda 函數叫用 DAGs
- 在不同的 Amazon MWAA 環境中叫用 DAGs
- 搭配使用 Amazon MWAA 與 Amazon RDS for Microsoft SQL Server
- 搭配使用 Amazon MWAA 與 Amazon EKS
- 使用 連線至 Amazon ECS ECSOperator
- 搭配 Amazon MWAA 使用 dbt
- AWS 部落格和教學課程

使用 DAG 在 CLI 中匯入變數

下列範例程式碼會使用 Amazon Managed Workflows for Apache Airflow 上的 CLI 匯入變數。

主題

- 版本
- 先決條件
- 權限
- 相依性
- 範例程式碼
- 後續步驟？

版本

您可以在 Python 3.10 中使用 Apache Airflow v2 和 Python 3.11 中使用此頁面的程式碼範例。 https://peps.python.org/pep-0619/

先決條件

使用此頁面上的程式碼範例不需要額外的許可。

權限

您的 AWS 帳戶 需要存取 `AmazonMWAAAirflowCliAccess`政策。若要進一步了解，請參閱 Apache Airflow CLI 政策：AmazonMWAAAirflowCliAccess。

相依性

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

範例程式碼

下列範例程式碼需要三個輸入：您的 Amazon MWAA 環境名稱 （在 中`mwaa_env`)、您環境 AWS 區域 的 （在 中`aws_region`)，以及包含您要匯入之變數的本機檔案 （在 中`var_file`)。

```
import boto3
import json
```

```
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwaa_env=''
aws_region=''
var_file=''

try:
    opts, args = getopt.getopt(argv, 'e:v:r:', ['environment', 'variable-
file','region'])
    #if len(opts) == 0 and len(opts) > 3:
    if len(opts) != 3:
        print ('Usage: -e MWAA environment -v variable file location and filename -r
 aws region')
    else:
        for opt, arg in opts:
            if opt in ("-e"):
                mwaa_env=arg
            elif opt in ("-r"):
                aws_region=arg
            elif opt in ("-v"):
                var_file=arg

        boto3.setup_default_session(region_name="{}".format(aws_region))
        mwaa_env_name = "{}".format(mwaa_env)

        client = boto3.client('mwaa')
        mwaa_cli_token = client.create_cli_token(
            Name=mwaa_env_name
        )

        with open ("{}".format(var_file), "r") as myfile:
            fileconf = myfile.read().replace('\n', '')

        json_dictionary = json.loads(fileconf)
        for key in json_dictionary:
            print(key, " ", json_dictionary[key])
            val = (key + " " + json_dictionary[key])
            mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
            mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
```

```
            raw_data = "variables set {0}".format(val)
            mwaa_response = requests.post(
                mwaa_webserver_hostname,
                headers={
                    'Authorization': mwaa_auth_token,
                    'Content-Type': 'text/plain'
                    },
                data=raw_data
                )
            mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
            mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')
            print(mwaa_response.status_code)
            print(mwaa_std_err_message)
            print(mwaa_std_out_message)

except:
    print('Use this script with the following options: -e MWAA environment -v variable
 file location and filename -r aws region')
    print("Unexpected error:", sys.exc_info()[0])
    sys.exit(2)
```

後續步驟？

- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 dags 資料夾新增或更新 DAGs。

使用 建立 SSH 連線 **SSHOperator**

下列範例說明如何使用定向無環圖 (DAG) SSHOperator中的 ，從 Amazon Managed Workflows for Apache Airflow 環境連線至遠端 Amazon EC2 執行個體。您可以使用類似的方法來連線至具有 SSH 存取的任何遠端執行個體。

在下列範例中，您將 SSH 私密金鑰 (.pem) 上傳至 Amazon S3 上的環境dags目錄。然後，您可以使用 安裝必要的相依性，requirements.txt並在 UI 中建立新的 Apache Airflow 連線。最後，您撰寫的 DAG 會建立與遠端執行個體的 SSH 連線。

主題

- 版本

- 先決條件
- 權限
- 要求
- 將您的私密金鑰複製到 Amazon S3
- 建立新的 Apache Airflow 連線
- 範例程式碼

版本

您可以在 Python 3.10 中使用 Apache Airflow v2 和 Python 3.11 中使用此頁面的程式碼範例。 https://peps.python.org/pep-0619/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。
- SSH 私密金鑰。程式碼範例假設您在與 Amazon MWAA 環境相同的區域中有 Amazon EC2 執行個體和 。 `.pem`如果您沒有金鑰，請參閱《Amazon EC2 使用者指南》中的建立或匯入金鑰對。

權限

使用此頁面上的程式碼範例不需要額外的許可。

要求

將下列參數新增至 `requirements.txt`，以在 Web 伺服器上安裝`apache-airflow-providers-ssh`套件。一旦您的環境更新且 Amazon MWAA 成功安裝相依性，您將在 UI 中取得新的 SSH 連線類型。

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
apache-airflow-providers-ssh
```

> **Note**
>
> `-c` 定義 中的限制條件 URL`requirements.txt`。這可確保 Amazon MWAA 為您的環境安裝正確的套件版本。

將您的私密金鑰複製到 Amazon S3

使用下列 AWS Command Line Interface 命令將您的`.pem`金鑰複製到 Amazon S3 中的環境`dags`目錄。

```
aws s3 cp your-secret-key.pem s3://amzn-s3-demo-bucket/dags/
```

Amazon MWAA 會將 中的內容`dags`，包括`.pem`金鑰，複製到本機`/usr/local/airflow/dags/`目錄。透過這樣做，Apache Airflow 可以存取金鑰。

建立新的 Apache Airflow 連線

使用 Apache Airflow UI 建立新的 SSH 連線

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 從環境清單中，為您的環境選擇 Open Airflow UI。
3. 在 Apache Airflow UI 頁面上，從主導覽列選擇管理員以展開下拉式清單，然後選擇連線。
4. 在列出連線頁面上，選擇 + ，或新增記錄按鈕以新增連線。
5. 在新增連線頁面上，新增下列資訊：
   a. 針對連線 ID，輸入 **`ssh_new`**。
   b. 針對連線類型，從下拉式清單中選擇 SSH。

      > **Note**
      >
      > 如果清單中沒有 SSH 連線類型，Amazon MWAA 尚未安裝必要的`apache-airflow-providers-ssh`套件。更新您的`requirements.txt`檔案以包含此套件，然後再試一次。

   c. 針對主機，輸入您要連線之 Amazon EC2 執行個體的 IP 地址。例如 **`12.345.67.89`**。

d. 針對使用者名稱，**ec2-user**如果您要連線至 Amazon EC2 執行個體，請輸入 。您的使用者名稱可能不同，取決於您希望 Apache Airflow 連線的遠端執行個體類型。

e. 對於 Extra，以 JSON 格式輸入下列鍵/值對：

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

此鍵/值對會指示 Apache Airflow 在本機/dags目錄中搜尋私密金鑰。

範例程式碼

下列 DAG 使用 SSHOperator 連線到您的目標 Amazon EC2 執行個體，然後執行 hostname Linux 命令來列印執行個體的名稱。您可以修改 DAG 以在遠端執行個體上執行任何命令或指令碼。

1. 開啟終端機，然後導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 ssh.py。

```
from airflow.decorators import dag
from datetime import datetime
from airflow.providers.ssh.operators.ssh import SSHOperator

@dag(
    dag_id="ssh_operator_example",
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    catchup=False,
    )
def ssh_dag():
    task_1=SSHOperator(
        task_id="ssh_task",
        ssh_conn_id='ssh_new',
        command='hostname',
    )

my_ssh_dag = ssh_dag()
```

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，您會收到類似 ssh_operator_example DAG 中 任務日誌中下列項目ssh_task的輸出：

```
[2022-01-01, 12:00:00 UTC] {{base.py:79}} INFO - Using connection to: id: ssh_new.
 Host: 12.345.67.89, Port: None,
Schema: , Login: ec2-user, Password: None, extra: {'key_file': '/usr/local/airflow/
dags/your-secret-key.pem'}
[2022-01-01, 12:00:00 UTC] {{ssh.py:264}} WARNING - Remote Identification Change is
 not verified. This won't protect against Man-In-The-Middle attacks [2022-01-01,
 12:00:00 UTC] {{ssh.py:270}} WARNING - No Host Key Verification. This won't
 protect against Man-In-The-Middle attacks
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Connected (version 2.0,
 client OpenSSH_7.4)
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Authentication (publickey)
 successful!
[2022-01-01, 12:00:00 UTC] {{ssh.py:139}} INFO - Running command: hostname
[2022-01-01, 12:00:00 UTC]{{ssh.py:171}} INFO - ip-123-45-67-89.us-
west-2.compute.internal
[2022-01-01, 12:00:00 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=ssh_operator_example, task_id=ssh_task, execution_date=20220712T200914,
 start_date=20220712T200915, end_date=20220712T200916
```

在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow Snowflake 連線

下列範例呼叫 AWS Secrets Manager 會在 Amazon Managed Workflows for Apache Airflow 上取得 Apache Airflow Snowflake 連線的私密金鑰。它假設您已完成 中的步驟使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

主題

- 版本
- 先決條件
- 權限
- 要求
- 範例程式碼

- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Secrets Manager 後端做為 Apache Airflow 組態選項，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。
- Secrets Manager 中的 Apache Airflow 連線字串，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

權限

- Secrets Manager 許可，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

要求

若要使用此頁面上的範例程式碼，請將下列相依性新增至您的 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
apache-airflow-providers-snowflake==1.3.0
```

範例程式碼

下列步驟說明如何建立 DAG 程式碼，呼叫 Secrets Manager 來取得秘密。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

   ```
   cd dags
   ```

2. 複製下列程式碼範例的內容，並在本機儲存為 `snowflake_connection.py`。

   ```
   """
   ```

```
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
from airflow.utils.dates import days_ago

snowflake_query = [
    """use warehouse "MY_WAREHOUSE";""",
    """select * from "SNOWFLAKE_SAMPLE_DATA"."WEATHER"."WEATHER_14_TOTAL" limit
 100;""",
]

with DAG(dag_id='snowflake_test', schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    snowflake_select = SnowflakeOperator(
        task_id="snowflake_select",
        sql=snowflake_query,
        snowflake_conn_id="snowflake_conn",
    )
```

後續步驟？

- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 dags 資料夾新增或更新 DAGs。

使用 DAG 在 CloudWatch 中寫入自訂指標

您可以使用下列程式碼範例來撰寫執行 的導向無環圖 (DAG)，Python0perator以擷取 Amazon MWAA 環境的作業系統層級指標。然後，DAG 會將資料作為自訂指標發佈至 Amazon CloudWatch。

自訂作業系統層級指標可讓您進一步了解環境工作者如何利用虛擬記憶體和 CPU 等資源。您可以使用此資訊來選取最適合工作負載的環境類別。

主題

- 版本
- 先決條件
- 權限
- 相依性
- 程式碼範例

版本

您可以在 Python 3.10 中使用 Apache Airflow v2 和 Python 3.11 中使用此頁面的程式碼範例。 https://peps.python.org/pep-0619/

先決條件

若要使用此頁面上的程式碼範例，您需要下列項目：

- Amazon MWAA 環境。

權限

使用此頁面上的程式碼範例不需要額外的許可。

相依性

- 使用此頁面上的程式碼範例不需要額外的相依性。

程式碼範例

1. 在命令提示中，導覽至存放 DAG 程式碼的資料夾。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並將其儲存為 dag-custom-metrics.py。將 取代MWAA-ENV-NAME為您的環境名稱。

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from datetime import datetime
import os,json,boto3,psutil,socket

def publish_metric(client,name,value,cat,unit='None'):
    environment_name = os.getenv("MWAA_ENV_NAME")
    value_number=float(value)
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    print('writing value',value_number,'to metric',name)
    response = client.put_metric_data(
        Namespace='MWAA-Custom',
        MetricData=[
            {
                'MetricName': name,
                'Dimensions': [
                    {
                        'Name': 'Environment',
                        'Value': environment_name
                    },
                    {
                        'Name': 'Category',
                        'Value': cat
                    },
                    {
                        'Name': 'Host',
                        'Value': ip_address
                    },
                ],
                'Timestamp': datetime.now(),
                'Value': value_number,
                'Unit': unit
            },
        ]
    )
```

```
    print(response)
    return response

def python_fn(**kwargs):
    client = boto3.client('cloudwatch')

    cpu_stats = psutil.cpu_stats()
    print('cpu_stats', cpu_stats)

    virtual = psutil.virtual_memory()
    cpu_times_percent = psutil.cpu_times_percent(interval=0)

    publish_metric(client=client, name='virtual_memory_total',
 cat='virtual_memory', value=virtual.total, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_available',
 cat='virtual_memory', value=virtual.available, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_used', cat='virtual_memory',
 value=virtual.used, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_free', cat='virtual_memory',
 value=virtual.free, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_active',
 cat='virtual_memory', value=virtual.active, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_inactive',
 cat='virtual_memory', value=virtual.inactive, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_percent',
 cat='virtual_memory', value=virtual.percent, unit='Percent')

    publish_metric(client=client, name='cpu_times_percent_user',
 cat='cpu_times_percent', value=cpu_times_percent.user, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_system',
 cat='cpu_times_percent', value=cpu_times_percent.system, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_idle',
 cat='cpu_times_percent', value=cpu_times_percent.idle, unit='Percent')

    return "OK"


with DAG(dag_id=os.path.basename(__file__).replace(".py", ""),
 schedule_interval='*/5 * * * *', catchup=False, start_date=days_ago(1)) as dag:
    t = PythonOperator(task_id="memory_test", python_callable=python_fn,
 provide_context=True)
```

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果 DAG 執行成功，您會在 Apache Airflow 日誌中取得類似以下內容的內容：

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO -
 cpu_stats scpustats(ctx_switches=3253992384, interrupts=1964237163,
 soft_interrupts=492328209, syscalls=0)
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 16024199168.0 to metric virtual_memory_total
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 14356287488.0 to metric virtual_memory_available
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': '6ef60085-07ab-4865-8abf-dc94f90cab46', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': '6ef60085-07ab-4865-8abf-dc94f90cab46',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 1342296064.0 to metric virtual_memory_used
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
...
[2022-08-16, 10:54:46 UTC] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-08-16, 10:54:46 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=dag-custom-metrics, task_id=memory_test, execution_date=20220816T175444,
 start_date=20220816T175445, end_date=20220816T175446
[2022-08-16, 10:54:46 UTC] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
```

Amazon MWAA 環境上的 Aurora PostgreSQL 資料庫清除

Amazon Managed Workflows for Apache Airflow 使用 Aurora PostgreSQL 資料庫做為 Apache Airflow 中繼資料資料庫，其中存放 DAG 執行和任務執行個體。下列範例程式碼會定期從 Amazon MWAA 環境的專用 Aurora PostgreSQL 資料庫清除項目。

主題

- 版本
- 先決條件
- 相依性
- 範例程式碼

版本

此頁面上的程式碼範例專屬於 Amazon MWAA 上支援的 Apache Airflow v2。請參閱支援的 Apache Airflow 版本。

> **Tip**
>
> 對於 Apache Airflow v3 使用者：如果您想要清除資料庫 （從中繼存放區資料表清除舊記錄），請執行 `db clean` CLI 命令。

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。

相依性

若要搭配 Apache Airflow v2 使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

範例程式碼

下列 DAG 會清除 中指定資料表的中繼資料資料庫TABLES_TO_CLEAN。此範例會從超過 30 天的指定資料表中刪除資料。若要調整項目的刪除時間，請將 MAX_AGE_IN_DAYS設定為不同的值。

Apache Airflow v2.4 to 2.10.3

```
from airflow import DAG
from airflow.models.param import Param
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

from datetime import datetime, timedelta

# Note: Database commands might time out if running longer than 5 minutes. If this
 occurs, please increase the MAX_AGE_IN_DAYS (or change
# timestamp parameter to an earlier date) for initial runs, then reduce on
 subsequent runs until the desired retention is met.

MAX_AGE_IN_DAYS = 30

# To clean specific tables, please provide a comma-separated list per
# https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-
ref.html#clean
# A value of None will clean all tables

TABLES_TO_CLEAN = None

with DAG(
    dag_id="clean_db_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1),
    params={
        "timestamp": Param(
            default=(datetime.now()-timedelta(days=MAX_AGE_IN_DAYS)).strftime("%Y-
%m-%d %H:%M:%S"),
            type="string",
            minLength=1,
            maxLength=255,
        ),
    }
) as dag:
```

```
    if TABLES_TO_CLEAN:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --tables '"+TABLES_TO_CLEAN+"' --skip-archive --yes"
    else:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --skip-archive --yes"

    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=bash_command
    )
```

Apache Airflow v2.2 and earlier

```
from airflow import settings
from airflow.utils.dates import days_ago
from airflow.models import DagTag, DagModel, DagRun, ImportError, Log, SlaMiss,
 RenderedTaskInstanceFields, TaskInstance, TaskReschedule, XCom
from airflow.decorators import dag, task
from airflow.utils.dates import days_ago
from time import sleep

from airflow.version import version
major_version, minor_version = int(version.split('.')[0]), int(version.split('.')
[1])
if major_version >= 2 and minor_version >= 6:
    from airflow.jobs.job import Job
else:
    # The BaseJob class was renamed as of Apache Airflow v2.6
    from airflow.jobs.base_job import BaseJob as Job

# Delete entries for the past 30 days. Adjust MAX_AGE_IN_DAYS to set how far back
 this DAG cleans the database.
MAX_AGE_IN_DAYS = 30
MIN_AGE_IN_DAYS = 0
DECREMENT = -7

# This is a list of (table, time) tuples.
# table = the table to clean in the metadata database
# time  = the column in the table associated to the timestamp of an entry
#         or None if not applicable.
TABLES_TO_CLEAN = [[Job, Job.latest_heartbeat],
    [TaskInstance, TaskInstance.execution_date],
```

```
    [TaskReschedule, TaskReschedule.execution_date],
    [DagTag, None],
    [DagModel, DagModel.last_parsed_time],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [XCom, XCom.execution_date],
]

@task()
def cleanup_db_fn(x):
    session = settings.Session()

    if x[1]:
        for oldest_days_ago in range(MAX_AGE_IN_DAYS, MIN_AGE_IN_DAYS, DECREMENT):
            earliest_days_ago = max(oldest_days_ago + DECREMENT, MIN_AGE_IN_DAYS)
            print(f"deleting {str(x[0])} entries between {earliest_days_ago} and
 {oldest_days_ago} days old...")
            earliest_date = days_ago(earliest_days_ago)
            oldest_date = days_ago(oldest_days_ago)
            query = session.query(x[0]).filter(x[1] >= earliest_date).filter(x[1] <=
 oldest_date)
            query.delete(synchronize_session= False)
            session.commit()
            sleep(5)
    else:
        # No time column specified for the table. Delete all entries
        print("deleting", str(x[0]), "...")
        query = session.query(x[0])
        query.delete(synchronize_session= False)
        session.commit()

    session.close()


@dag(
    dag_id="cleanup_db",
    schedule_interval="@weekly",
    start_date=days_ago(7),
    catchup=False,
    is_paused_upon_creation=False
)
```

```
def clean_db_dag_fn():
    t_last=None
    for x in TABLES_TO_CLEAN:
        t=cleanup_db_fn(x)
        if t_last:
            t_last >> t
        t_last = t

clean_db_dag = clean_db_dag_fn()
```

將環境中繼資料匯出至 Amazon S3 上的 CSV 檔案

使用下列程式碼範例建立定向無環圖 (DAG)，查詢資料庫以取得一系列 DAG 執行資訊，並將資料寫入 Amazon S3 上存放`.csv`的檔案。

您可能想要從環境的 Aurora PostgreSQL 資料庫匯出資訊，以在本機檢查資料、將其封存在物件儲存中，或將其與 Amazon S3 等工具結合到 Amazon Redshift 運算子和資料庫清除，以將 Amazon MWAA 中繼資料移出環境，但保留以供未來分析。

您可以查詢資料庫，尋找 Apache Airflow 模型中列出的任何物件。此程式碼範例使用三種模型`DagRun`、 `TaskFail`和 `TaskInstance`，可提供與 DAG 執行相關的資訊。

主題

- 版本
- 先決條件
- 許可
- 要求
- 範例程式碼

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。
- 您要匯出中繼資料資訊的新 Amazon S3 儲存貯體。

許可

Amazon MWAA 需要 Amazon S3 動作的許可`s3:PutObject`，才能將查詢的中繼資料資訊寫入 Amazon S3。將下列政策陳述式新增至您環境的執行角色。

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
}
```

此政策僅限制對 *amzn-s3-demo-bucket* 的寫入存取。

要求

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

範例程式碼

下列步驟說明如何建立查詢 Aurora PostgreSQL 的 DAG，並將結果寫入新的 Amazon S3 儲存貯體。

1. 在終端機中，導覽至存放 DAG 程式碼的目錄。例如：

   ```
   cd dags
   ```

2. 複製下列程式碼範例的內容，並將其儲存為 `metadata_to_csv.py`。您可以變更指派給 的值`MAX_AGE_IN_DAYS`，以控制 DAG 從中繼資料資料庫查詢的最舊記錄的存留期。

   ```
   from airflow.decorators import dag, task
   from airflow import settings
   import os
   import boto3
   from airflow.utils.dates import days_ago
   from airflow.models import DagRun, TaskFail, TaskInstance
   import csv, re
   from io import StringIO
   ```

```
DAG_ID = os.path.basename(__file__).replace(".py", "")

MAX_AGE_IN_DAYS = 30
S3_BUCKET = '<your-export-bucket>'
S3_KEY = 'files/export/{0}.csv'

# You can add other objects to export from the metadatabase,
OBJECTS_TO_EXPORT = [
    [DagRun,DagRun.execution_date],
    [TaskFail,TaskFail.end_date],
    [TaskInstance, TaskInstance.execution_date],
]

@task()
def export_db_task(**kwargs):
    session = settings.Session()
    print("session: ",str(session))

    oldest_date = days_ago(MAX_AGE_IN_DAYS)
    print("oldest_date: ",oldest_date)

    s3 = boto3.client('s3')

    for x in OBJECTS_TO_EXPORT:
        query = session.query(x[0]).filter(x[1] >= days_ago(MAX_AGE_IN_DAYS))
        print("type",type(query))
        allrows=query.all()
        name=re.sub("[<>']", "", str(x[0]))
        print(name,": ",str(allrows))

        if len(allrows) > 0:
            outfileStr=""
            f = StringIO(outfileStr)
            w = csv.DictWriter(f, vars(allrows[0]).keys())
            w.writeheader()
            for y in allrows:
                w.writerow(vars(y))
            outkey = S3_KEY.format(name[6:])
            s3.put_object(Bucket=S3_BUCKET, Key=outkey, Body=f.getvalue())

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
```

```
    start_date=days_ago(1),
    )
def export_db():
    t = export_db_task()

metadb_to_s3_test = export_db()
```

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，您將在任務的任務日誌中輸出類似以下內容export_db：

```
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.dagrun.DagRun : [your-tasks]
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskfail.TaskFail :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskinstance.TaskInstance :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=metadb_to_s3, task_id=export_db, execution_date=20220101T000000,
 start_date=20220101T000000, end_date=20220101T000000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

您現在可以在 的新 Amazon S3 儲存貯體中存取和下載匯出.csv的檔案/files/export/。

針對 AWS Secrets Manager Apache Airflow 變數在 中使用私密金鑰

下列範例呼叫 AWS Secrets Manager 會在 Amazon Managed Workflows for Apache Airflow 上取得 Apache Airflow 變數的私密金鑰。它假設您已完成 中的步驟使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

主題

- 版本
- 先決條件
- 權限
- 要求
- 範例程式碼
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Secrets Manager 後端做為 Apache Airflow 組態選項，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。
- Secrets Manager 中的 Apache Airflow 變數字串，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

權限

- Secrets Manager 許可，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

要求

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

範例程式碼

下列步驟說明如何建立 DAG 程式碼，呼叫 Secrets Manager 來取得秘密。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 `secrets-manager-var.py`。

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.models import Variable
from airflow.utils.dates import days_ago
from datetime import timedelta
import os
DAG_ID = os.path.basename(__file__).replace(".py", "")
DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}
def get_variable_fn(**kwargs):
    my_variable_name = Variable.get("test-variable", default_var="undefined")
    print("my_variable_name: ", my_variable_name)
    return my_variable_name
with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['variable']
) as dag:
    get_variable = PythonOperator(
        task_id="get_variable",
        python_callable=get_variable_fn,
        provide_context=True
    )
```

後續步驟？

- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 dags 資料夾新增或更新 DAGs。

在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線

下列範例呼叫 AWS Secrets Manager 會在 Amazon Managed Workflows for Apache Airflow 上取得 Apache Airflow 連線的私密金鑰。它假設您已完成 中的步驟使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

主題

- 版本
- 先決條件
- 權限
- 要求
- 範例程式碼
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Secrets Manager 後端做為 Apache Airflow 組態選項，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。
- Secrets Manager 中的 Apache Airflow 連線字串，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

權限

- Secrets Manager 許可，如 所列使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。

要求

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

範例程式碼

下列步驟說明如何建立 DAG 程式碼，呼叫 Secrets Manager 來取得秘密。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 `secrets-manager.py`。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os
```

```
### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type(region_name='us-east-1')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

後續步驟？

- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 dags 資料夾新增或更新 DAGs。

使用 Oracle 建立自訂外掛程式

下列範例會逐步解說使用 Oracle for Amazon MWAA 建立自訂外掛程式的步驟，並可與您的 plugins.zip 檔案中的其他自訂外掛程式和二進位檔結合使用。

內容

- 版本
- 先決條件
- 權限
- 要求
- 範例程式碼
- 建立自訂外掛程式
  - 下載相依性
  - 自訂外掛程式
  - Plugins.zip
- Airflow 組態選項
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。
- 在任何日誌層級CRITICAL或環境的上一節中啟用工作者記錄。如需 Amazon MWAA 日誌類型以及如何管理日誌群組的詳細資訊，請參閱 the section called “存取 Airflow 日誌”

權限

使用此頁面上的程式碼範例不需要額外的許可。

要求

若要使用此頁面上的範例程式碼，請將下列相依性新增至您的 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

範例程式碼

下列步驟說明如何建立 DAG 程式碼來測試自訂外掛程式。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

   ```
   cd dags
   ```

2. 複製下列程式碼範例的內容，並在本機儲存為 `oracle.py`。

   ```
   from airflow import DAG
   from airflow.operators.python_operator import PythonOperator
   from airflow.utils.dates import days_ago
   import os
   import cx_Oracle

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   def testHook(**kwargs):
       cx_Oracle.init_oracle_client()
       version = cx_Oracle.clientversion()
       print("cx_Oracle.clientversion",version)
       return version

   with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
    start_date=days_ago(1)) as dag:
       hook_test = PythonOperator(
           task_id="hook_test",
           python_callable=testHook,
           provide_context=True
       )
   ```

建立自訂外掛程式

本節說明如何下載相依性、建立自訂外掛程式和 plugins.zip。

下載相依性

Amazon MWAA 會在每個 Amazon MWAA 排程器和工作者容器/usr/local/airflow/plugins上，將 plugins.zip 的內容擷取至 。這是用來將二進位檔新增至您的環境。下列步驟說明如何組合自訂外掛程式所需的檔案。

提取 Amazon Linux 容器映像

1. 在您的命令提示字元中，提取 Amazon Linux 容器映像，並在本機執行容器。例如：

   ```
   docker pull amazonlinux
         docker run -it amazonlinux:latest /bin/bash
   ```

   您的命令提示可以叫用 bash 命令列。例如：

   ```
   bash-4.2#
   ```

2. 安裝 Linux 原生非同步 I/O 設施 (libaio)。

   ```
   yum -y install libaio
   ```

3. 保持此視窗開啟以進行後續步驟。我們將在本機複製下列檔案：lib64/libaio.so.1、lib64/libaio.so.1.0.0、lib64/libaio.so.1.0.1。

下載用戶端資料夾

1. 在本機安裝 unzip 套件。例如：

   ```
   sudo yum install unzip
   ```

2. 建立 oracle_plugin 目錄。例如：

   ```
   mkdir oracle_plugin
   cd oracle_plugin
   ```

3. 使用以下 curl 命令從 Oracle Instant Client Downloads for Linux x86-64 (64 位元） 下載 instantclient-basic-linux.x64-18.5.0.0.0dbru.zip。 https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. 解壓縮 `client.zip` 檔案。例如：

```
unzip *.zip
```

從 Docker 擷取檔案

1. 在新的命令提示字元中，顯示並寫下您的 Docker 容器 ID。例如：

```
docker container ls
```

您的命令提示可以傳回所有容器及其 IDs。例如：

```
debc16fd6970
```

2. 在您的`oracle_plugin`目錄中，將 `lib64/libaio.so.1`、`lib64/libaio.so.1.0.0`、`lib64/libaio.so.1.0.1` 檔案擷取至本機`instantclient_18_5`資料夾。例如：

```
docker cp debc16fd6970:/lib64/libaio.so.1 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.0 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.1 instantclient_18_5/
```

自訂外掛程式

Apache Airflow 會在啟動時執行外掛程式資料夾中 Python 檔案的內容。這用於設定和修改環境變數。下列步驟說明自訂外掛程式的範例程式碼。

- 複製下列程式碼範例的內容，並在本機儲存為 `env_var_plugin_oracle.py`。

```
from airflow.plugins_manager import AirflowPlugin
import os
```

```
os.environ["LD_LIBRARY_PATH"]='/usr/local/airflow/plugins/instantclient_18_5'
os.environ["DPI_DEBUG_LEVEL"]="64"

class EnvVarPlugin(AirflowPlugin):
    name = 'env_var_plugin'
```

Plugins.zip

下列步驟說明如何建立 plugins.zip。此範例的內容可以與其他外掛程式和二進位檔合併為單一plugins.zip檔案。

壓縮外掛程式目錄的內容

1. 在您的命令提示字元中，導覽至 oracle_plugin目錄。例如：

   ```
   cd oracle_plugin
   ```

2. 在 plugins.zip 中壓縮instantclient_18_5目錄。例如：

   ```
   zip -r ../plugins.zip ./
   ```

   您的命令提示字元會顯示：

   ```
   oracle_plugin$ ls
   client.zip  instantclient_18_5
   ```

3. 移除 client.zip 檔案。例如：

   ```
   rm client.zip
   ```

壓縮 env_var_plugin_oracle.py 檔案

1. 將 env_var_plugin_oracle.py 檔案新增至 plugins.zip 的根目錄。例如：

   ```
   zip plugins.zip env_var_plugin_oracle.py
   ```

2. 您的 plugins.zip 現在包含下列項目：

   ```
   env_var_plugin_oracle.py
   ```

```
instantclient_18_5/
```

Airflow 組態選項

如果您使用的是 Apache Airflow v2，請將 新增`core.lazy_load_plugins : False`為 Apache Airflow 組態選項。若要進一步了解，請參閱使用組態選項載入 2 中的外掛程式。

後續步驟？

- 了解如何在此範例中將 `requirements.txt` 檔案上傳至 中的 Amazon S3 儲存貯體安裝 Python 相依性。
- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 `dags` 資料夾新增或更新 DAGs。
- 進一步了解如何在此範例中將`plugins.zip`檔案上傳至 中的 Amazon S3 儲存貯體安裝自訂外掛程式。

在 Amazon MWAA 上變更 DAG 的時區

Apache Airflow 預設會以 UTC+0 排程您的導向無環圖 (DAG)。下列步驟說明如何變更 Amazon MWAA 使用 Pendulum 執行 DAGs時區。或者，本主題示範如何建立自訂外掛程式，以變更您環境 Apache Airflow 日誌的時區。

主題

- 版本
- 先決條件
- 許可
- 建立外掛程式以變更 Airflow 日誌中的時區
- 建立 plugins.zip
- 範例程式碼
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。

許可

使用此頁面上的程式碼範例不需要額外的許可。

建立外掛程式以變更 Airflow 日誌中的時區

Apache Airflow 會在啟動時在 `plugins`目錄中執行 Python 檔案。使用下列外掛程式，您可以覆寫執行器的時區，以修改 Apache Airflow 寫入日誌的時區。

1. `plugins` 為您的自訂外掛程式建立名為 的目錄，然後導覽至 目錄。例如：

```
$ mkdir plugins
$ cd plugins
```

2. 複製下列程式碼範例的內容，並在本機儲存為 `dag-timezone-plugin.py` 資料夾中的 `plugins`。

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. 在 `plugins`目錄中，建立名為 的空白 Python 檔案`__init__.py`。您的`plugins`目錄應該類似於以下內容：

```
plugins/
 |-- __init__.py
 |-- dag-timezone-plugin.py
```

建立 plugins.zip

下列步驟說明如何建立 plugins.zip。此範例的內容可以與其他外掛程式和二進位檔合併為單一plugins.zip檔案。

1. 在您的命令提示字元中，導覽至上一個步驟的 plugins 目錄。例如：

```
cd plugins
```

2. 壓縮plugins目錄中的內容。

```
zip -r ../plugins.zip ./
```

3. plugins.zip 上傳至您的 S3 儲存貯體

```
aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

範例程式碼

若要變更 DAG 執行的預設時區 (UTC+0)，我們將使用名為 Pendulum 的程式庫，Python 程式庫可用來使用時區感知日期時間。

1. 在命令提示中，導覽至存放 DAGs目錄。例如：

```
cd dags
```

2. 複製下列範例的內容並儲存為 tz-aware-dag.py。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
# Import the Pendulum library.
import pendulum

# Instantiate Pendulum and set your timezone.
local_tz = pendulum.timezone("America/Los_Angeles")

with DAG(
    dag_id = "tz_test",
    schedule_interval="0 12 * * *",
```

```
    catchup=False,
    start_date=datetime(2022, 1, 1, tzinfo=local_tz)
) as dag:
    bash_operator_task = BashOperator(
        task_id="tz_aware_task",
        dag=dag,
        bash_command="date"
    )
```

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，您將輸出類似 `tz_test` DAG 中 的任務日誌`tz_aware_task`中的以下內容：

```
[2022-08-01, 12:00:00 PDT] {{subprocess.py:74}} INFO - Running command: ['bash', '-
c', 'date']
[2022-08-01, 12:00:00 PDT] {{subprocess.py:85}} INFO - Output:
[2022-08-01, 12:00:00 PDT] {{subprocess.py:89}} INFO - Mon Aug  1 12:00:00 PDT 2022
[2022-08-01, 12:00:00 PDT] {{subprocess.py:93}} INFO - Command exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=tz_test, task_id=tz_aware_task, execution_date=20220801T190033,
 start_date=20220801T190035, end_date=20220801T190035
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

後續步驟？

- 進一步了解如何在此範例中將`plugins.zip`檔案上傳至 中的 Amazon S3 儲存貯體安裝自訂外掛程式。

重新整理 CodeArtifact 權杖

如果您使用 CodeArtifact 安裝 Python 相依性，Amazon MWAA 需要作用中的字符。若要允許 Amazon MWAA 在執行時間存取 CodeArtifact 儲存庫，您可以使用啟動指令碼並使用`PIP_EXTRA_INDEX_URL`字符設定 。

下列主題說明如何建立啟動指令碼，該指令碼使用 `get_authorization_token` CodeArtifact API 操作在每次環境啟動或更新時擷取新的字符。

主題

- 版本
- 先決條件
- 權限
- 範例程式碼
- 後續步驟？

版本

您可以在 Python 3.10 中使用 Apache Airflow v2 和 Python 3.11 中使用此頁面的程式碼範例。 https://peps.python.org/pep-0619/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。
- CodeArtifact 儲存庫，您可以在其中存放您環境的相依性。

權限

若要重新整理 CodeArtifact 字符並將結果寫入 Amazon S3 Amazon MWAA，必須在執行角色中具有下列許可。

- `codeartifact:GetAuthorizationToken` 動作允許 Amazon MWAA 從 CodeArtifact 擷取新權杖。下列政策會為您建立的每個 CodeArtifact 網域授予許可。您可以透過修改 陳述式中的資源值，並僅指定您希望環境存取的網域，進一步限制對網域的存取。

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- 呼叫 CodeArtifact GetAuthorizationToken API 操作需要 sts:GetServiceBearerToken動作。此操作會傳回權杖，在pip搭配 CodeArtifact 使用 等套件管理員時必須使用該權杖。若要搭配 CodeArtifact 儲存庫使用套件管理員，您環境的執行角色角色必須允許 sts:GetServiceBearerToken，如下列政策陳述式所列。

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

範例程式碼

下列步驟說明如何建立更新 CodeArtifact 字符的啟動指令碼。

1. 複製下列程式碼範例的內容，並在本機儲存為 code_artifact_startup_script.sh。

```
#!/bin/sh

# Startup script for MWAA, refer to https://docs.aws.amazon.com/mwaa/latest/
userguide/using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"
echo "CodeArtifact startup setup complete"
```

2. 導覽至您儲存指令碼的資料夾。在cp新的提示視窗中使用 ，將指令碼上傳至您的儲存貯體。將 *amzn-s3-demo-bucket* 取代為您的資訊。

```
aws s3 cp code_artifact_startup_script.sh s3://amzn-s3-demo-bucket/
code_artifact_startup_script.sh
```

如果成功，Amazon S3 會輸出物件的 URL 路徑：

```
upload: ./code_artifact_startup_script.sh to s3://amzn-s3-demo-bucket/
code_artifact_startup_script.sh
```

上傳指令碼後，您的環境會在啟動時更新並執行指令碼。

後續步驟？

- 了解如何使用啟動指令碼在 中自訂您的環境the section called “使用啟動指令碼”。
- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 dags 資料夾新增或更新 DAGs。
- 進一步了解如何在此範例中將plugins.zip檔案上傳至 中的 Amazon S3 儲存貯體安裝自訂外掛程式。

使用 Apache Hive 和 Hadoop 建立自訂外掛程式

Amazon MWAA 會將 的內容擷取plugins.zip至 /usr/local/airflow/plugins。這可用於將二進位檔新增至您的容器。此外，Apache Airflow 會在啟動時在 plugins 資料夾中執行 Python 檔案的內容，讓您能夠設定和修改環境變數。下列範例會逐步解說在 Amazon Managed Workflows for Apache Airflow 環境上使用 Apache Hive 和 Hadoop 建立自訂外掛程式的步驟，並可與其他自訂外掛程式和二進位檔結合使用。

主題

- 版本
- 先決條件
- 權限
- 要求
- 下載相依性

- 自訂外掛程式
- Plugins.zip
- 範例程式碼
- Airflow 組態選項
- 後續步驟？

版本

您可以在 Python 3.10 中使用 Apache Airflow v2 和 Python 3.11 中使用此頁面的程式碼範例。 https://peps.python.org/pep-0619/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。

權限

使用此頁面上的程式碼範例不需要額外的許可。

要求

若要使用此頁面上的範例程式碼，請將下列相依性新增至您的 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

下載相依性

Amazon MWAA 會在每個 Amazon MWAA 排程器和工作者容器`/usr/local/airflow/plugins`上，將 plugins.zip 的內容擷取至 。這是用來將二進位檔新增至您的環境。下列步驟說明如何組合自訂外掛程式所需的檔案。

1. 在命令提示字元中，導覽至您要建立外掛程式的目錄。例如：

```
cd plugins
```

2. 從鏡像下載 Hadoop，例如：

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

3. 從鏡像下載 Hive，例如：

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

4. 建立目錄。例如：

```
mkdir hive_plugin
```

5. 擷取 Hadoop。

```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. 擷取 Hive。

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

自訂外掛程式

Apache Airflow 會在啟動時執行外掛程式資料夾中 Python 檔案的內容。這用於設定和修改環境變數。下列步驟說明自訂外掛程式的範例程式碼。

1. 在您的命令提示字元中，導覽至 `hive_plugin`目錄。例如：

```
cd hive_plugin
```

2. 複製下列程式碼範例的內容，並在本機儲存為 `hive_plugin.py` `hive_plugin`目錄中的 。

```
from airflow.plugins_manager import AirflowPlugin
import os
os.environ["JAVA_HOME"]="/usr/lib/jvm/jre"
os.environ["HADOOP_HOME"]='/usr/local/airflow/plugins/hadoop-3.3.0'
os.environ["HADOOP_CONF_DIR"]='/usr/local/airflow/plugins/hadoop-3.3.0/etc/hadoop'
os.environ["HIVE_HOME"]='/usr/local/airflow/plugins/apache-hive-3.1.2-bin'
```

```
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/plugins/
hadoop-3.3.0:/usr/local/airflow/plugins/apache-hive-3.1.2-bin/bin:/usr/local/
airflow/plugins/apache-hive-3.1.2-bin/lib"
os.environ["CLASSPATH"] = os.getenv("CLASSPATH") + ":/usr/local/airflow/plugins/
apache-hive-3.1.2-bin/lib"
class EnvVarPlugin(AirflowPlugin):
    name = 'hive_plugin'
```

3. 處理下列文字的內容，並在本機儲存為 .airflowignore hive_plugin目錄中的 。

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

下列步驟說明如何建立 plugins.zip。此範例的內容可以與其他外掛程式和二進位檔合併為單一plugins.zip檔案。

1. 在您的命令提示字元中，導覽至上一個步驟的 hive_plugin 目錄。例如：

```
cd hive_plugin
```

2. 壓縮plugins資料夾中的內容。

```
zip -r ../hive_plugin.zip ./
```

範例程式碼

下列步驟說明如何建立 DAG 程式碼來測試自訂外掛程式。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 hive.py。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago
```

```
with DAG(dag_id="hive_test_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    hive_test = BashOperator(
        task_id="hive_test",
        bash_command='hive --help'
    )
```

Airflow 組態選項

如果您使用的是 Apache Airflow v2，請將 新增`core.lazy_load_plugins : False`為 Apache Airflow 組態選項。若要進一步了解，請參閱使用組態選項載入 2 中的外掛程式。

後續步驟？

- 了解如何在此範例中將`requirements.txt`檔案上傳至 中的 Amazon S3 儲存貯體安裝 Python 相依性。
- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 `dags` 資料夾新增或更新 DAGs。
- 進一步了解如何在此範例中將`plugins.zip`檔案上傳至 中的 Amazon S3 儲存貯體安裝自訂外掛程式。

為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式

下列範例說明如何在 Amazon Managed Workflows for Apache Airflow 上使用`PythonVirtualenvOperator`自訂外掛程式修補 Apache Airflow。

主題

- 版本
- 先決條件
- 權限
- 要求
- 自訂外掛程式範例程式碼
- Plugins.zip
- 範例程式碼

- Airflow 組態選項
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。

權限

使用此頁面上的程式碼範例不需要額外的許可。

要求

若要使用此頁面上的範例程式碼，請將下列相依性新增至您的 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
virtualenv
```

自訂外掛程式範例程式碼

Apache Airflow 會在啟動時執行外掛程式資料夾中 Python 檔案的內容。此外掛程式會在該啟動程序`PythonVirtualenvOperator`期間修補內建 ，使其與 Amazon MWAA 相容。下列步驟顯示自訂外掛程式的範例程式碼。

1. 在您的命令提示字元中，導覽至上一節中的 `plugins` 目錄。例如：

   ```
   cd plugins
   ```

2. 複製下列程式碼範例的內容，並在本機儲存為 `virtual_python_plugin.py`。

   ```
   """
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   ```

```
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Plugins.zip

下列步驟說明如何建立 plugins.zip。

1. 在您的命令提示字元中，導覽至上一節virtual_python_plugin.py中包含 的目錄。例如：

```
cd plugins
```

2. 壓縮plugins資料夾中的內容。

```
zip plugins.zip virtual_python_plugin.py
```

範例程式碼

下列步驟說明如何建立自訂外掛程式的 DAG 程式碼。

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 virtualenv_test.py。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from airflow.operators.python import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
```

```
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Airflow 組態選項

如果您使用的是 Apache Airflow v2，請將 新增`core.lazy_load_plugins : False`為 Apache Airflow 組態選項。若要進一步了解，請參閱使用組態選項載入 2 中的外掛程式。

後續步驟？

- 了解如何在此範例中將`requirements.txt`檔案上傳至 中的 Amazon S3 儲存貯體安裝 Python 相依性。
- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 `dags` 資料夾新增或更新 DAGs。
- 進一步了解如何在此範例中將`plugins.zip`檔案上傳至 中的 Amazon S3 儲存貯體安裝自訂外掛程式。

使用 Lambda 函數叫用 DAGs

下列程式碼範例使用 AWS Lambda函數來取得 Apache Airflow CLI 字符，並在 Amazon MWAA 環境中調用定向無環圖 (DAG)。

主題

- 版本
- 先決條件
- 許可
- 相依性
- 程式碼範例

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此程式碼範例，您必須：

- 為您的 Amazon MWAA 環境使用公有網路存取模式。
- 使用最新的 Python 執行時間來擁有 Lambda 函數。

> **Note**
>
> 如果 Lambda 函數和您的 Amazon MWAA 環境位於相同的 VPC 中，您可以在私有網路上使用此程式碼。對於此組態，Lambda 函數的執行角色需要呼叫 Amazon Elastic Compute Cloud (Amazon EC2) CreateNetworkInterface API 操作的許可。您可以使用 `AWSLambdaVPCAccessExecutionRole` AWS受管政策提供此許可。

許可

若要使用此頁面上的程式碼範例，Amazon MWAA 環境的執行角色需要存取 才能執行`airflow:CreateCliToken`動作。您可以使用 `AmazonMWAAAirflowCliAccess` AWS受管政策提供此許可：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

如需詳細資訊，請參閱 Apache Airflow CLI 政策：AmazonMWAAAirflowCliAccess。

相依性

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

程式碼範例

1. 在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 從函數清單中選擇您的 Lambda 函數。
3. 在函數頁面上，複製下列程式碼，並以您的資源名稱取代下列程式碼：

   - YOUR_ENVIRONMENT_NAME – Amazon MWAA 環境的名稱。
   - YOUR_DAG_NAME – 您要叫用的 DAG 名稱。

```
import boto3
import http.client
import base64
import ast
mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

def lambda_handler(event, context):
    # get web token
    mwaa_cli_token = client.create_cli_token(
        Name=mwaa_env_name
    )

    conn = http.client.HTTPSConnection(mwaa_cli_token['WebServerHostname'])
    payload = mwaa_cli_command + " " + dag_name
    headers = {
      'Authorization': 'Bearer ' + mwaa_cli_token['CliToken'],
      'Content-Type': 'text/plain'
    }
```

```
    conn.request("POST", "/aws_mwaa/cli", payload, headers)
    res = conn.getresponse()
    data = res.read()
    dict_str = data.decode("UTF-8")
    mydata = ast.literal_eval(dict_str)
    return base64.b64decode(mydata['stdout'])
```

4. 選擇部署。
5. 選擇測試以使用 Lambda 主控台叫用函數。
6. 若要驗證您的 Lambda 是否已成功調用 DAG，請使用 Amazon MWAA 主控台導覽至您環境的 Apache Airflow UI，然後執行下列動作：
   a. 在 DAGs頁面上，在 DAG 清單中找到您的新目標 DAGs。
   b. 在上次執行下，檢查最新 DAG 執行的時間戳記。此時間戳記應緊密符合您`invoke_dag`其他環境中 的最新時間戳記。
   c. 在最近任務下，檢查上次執行是否成功。

在不同的 Amazon MWAA 環境中叫用 DAGs

下列程式碼範例會建立 Apache Airflow CLI 字符。然後，程式碼會在一個 Amazon MWAA 環境中使用定向無環圖 (DAG)，在不同的 Amazon MWAA 環境中叫用 DAG。

主題

- 版本
- 先決條件
- 許可
- 相依性
- 程式碼範例

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的程式碼範例，您需要下列項目：

- 兩個具有公有網路 Web 伺服器存取權的 Amazon MWAA 環境，包括您目前的環境。
- 上傳至目標環境 Amazon Simple Storage Service (Amazon S3) 儲存貯體的範例 DAG。

許可

若要使用此頁面上的程式碼範例，您環境的執行角色必須具有建立 Apache Airflow CLI 權杖的許可。您可以連接 AWS受管政策`AmazonMWAAAirflowCliAccess`來授予此許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
}
```

如需詳細資訊，請參閱 Apache Airflow CLI 政策：AmazonMWAAAirflowCliAccess。

相依性

若要搭配 Apache Airflow v2 和更新版本使用此程式碼範例，不需要額外的相依性。使用 aws-mwaa-docker-images 來安裝 Apache Airflow。

程式碼範例

下列程式碼範例假設您在目前環境中使用 DAG，以在另一個環境中叫用 DAG。

1. 在終端機中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並將其儲存為 `invoke_dag.py`。將下列值取代為您的資訊。

- your-new-environment-name – 您要叫用 DAG 的其他環境名稱。
- your-target-dag-id – 您要叫用之其他環境中的 DAG ID。

```
from airflow.decorators import dag, task
import boto3
from datetime import datetime, timedelta
import os, requests

DAG_ID = os.path.basename(__file__).replace(".py", "")

@task()
def invoke_dag_task(**kwargs):
    client = boto3.client('mwaa')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwaa/cli"
    body = 'dags trigger your-target-dag-id'
    headers = {
        'Authorization' : 'Bearer ' + token['CliToken'],
        'Content-Type': 'text/plain'
        }
    requests.post(url, data=body, headers=headers)

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    dagrun_timeout=timedelta(minutes=60),
    catchup=False
    )
def invoke_dag():
    t = invoke_dag_task()

invoke_dag_test = invoke_dag()
```

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果 DAG 成功執行，您會在 的任務日誌中收到類似以下的輸出invoke_dag_task。

```
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: None
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=invoke_dag, task_id=invoke_dag_task, execution_date=20220101T120000,
 start_date=20220101T120000, end_date=20220101T120000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

若要確認您的 DAG 已成功調用，請導覽至新環境的 Apache Airflow UI，然後執行下列動作：

a. 在 DAGs頁面上，在 DAG 清單中找到您的新目標 DAGs。

b. 在上次執行下，檢查最新 DAG 執行的時間戳記。此時間戳記應緊密符合您`invoke_dag`其他環境中 的最新時間戳記。

c. 在最近任務下，檢查上次執行是否成功。

搭配使用 Amazon MWAA 與 Amazon RDS for Microsoft SQL Server

您可以使用 Amazon Managed Workflows for Apache Airflow 連線到 RDS for SQL Server。下列範例程式碼使用 Amazon Managed Workflows for Apache Airflow 環境上的 DAGs 來連線至 Amazon RDS for Microsoft SQL Server 並執行查詢。

主題

- 版本
- 先決條件
- 相依性
- Apache Airflow v2 連線
- 範例程式碼
- 後續步驟？

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。
- Amazon MWAA 和 RDS for SQL Server 在相同的 Amazon VPC/ 中執行
- Amazon MWAA 和伺服器的 VPC 安全群組已設定下列連線：
  - Amazon MWAA 安全群組中為 Amazon RDS 1433開啟之連接埠的傳入規則
  - 或從 Amazon MWAA 至 RDS 1433開啟之連接埠的傳出規則
- RDS for SQL Server 的 Apache Airflow Connection 會反映先前程序中建立之 Amazon RDS SQL Server 資料庫的主機名稱、連接埠、使用者名稱和密碼。

相依性

若要使用本節中的範例程式碼，請將下列相依性新增至您的 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
apache-airflow-providers-microsoft-mssql==1.0.1
   apache-airflow-providers-odbc==1.0.1
   pymssql==2.2.1
```

Apache Airflow v2 連線

如果您在 Apache Airflow v2 中使用連線，請確定 Airflow 連線物件包含下列鍵/值對：

1. Conn ID：mssql_default
2. Conn 類型：Amazon Web Services
3. 主機： YOUR_DB_HOST
4. 結構描述：
5. 登入：admin
6. 密碼：
7. 連接埠：1433
8. 額外：

範例程式碼

1. 在命令提示中，導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 sql-server.py。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import pymssql
import logging
import sys
from airflow import DAG
from datetime import datetime
from airflow.operators.mssql_operator import MsSqlOperator
from airflow.operators.python_operator import PythonOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'mssql_conn_example', default_args=default_args, schedule_interval=None)

drop_db = MsSqlOperator(
   task_id="drop_db",
   sql="DROP DATABASE IF EXISTS testdb;",
```

```
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_db = MsSqlOperator(
   task_id="create_db",
   sql="create database testdb;",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_table = MsSqlOperator(
   task_id="create_table",
   sql="CREATE TABLE testdb.dbo.pet (name VARCHAR(20), owner VARCHAR(20));",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

insert_into_table = MsSqlOperator(
   task_id="insert_into_table",
   sql="INSERT INTO testdb.dbo.pet VALUES ('Olaf', 'Disney');",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

def select_pet(**kwargs):
   try:
        conn = pymssql.connect(
            server='sampledb.<xxxxxx>.<region>.rds.amazonaws.com',
            user='admin',
            password='<yoursupersecretpassword>',
            database='testdb'
        )

        # Create a cursor from the connection
        cursor = conn.cursor()
        cursor.execute("SELECT * from testdb.dbo.pet")
        row = cursor.fetchone()

        if row:
```

```
            print(row)
   except:
      logging.error("Error when creating pymssql database connection: %s",
 sys.exc_info()[0])

select_query = PythonOperator(
    task_id='select_query',
    python_callable=select_pet,
    dag=dag,
)

drop_db >> create_db >> create_table >> insert_into_table >> select_query
```

後續步驟？

- 了解如何在此範例中將 `requirements.txt` 檔案上傳至 中的 Amazon S3 儲存貯體安裝 Python 相依性。
- 了解如何在此範例中將 DAG 程式碼上傳至 Amazon S3 儲存貯體中的 `dags` 資料夾新增或更新 DAGs。
- 探索範例指令碼和其他 pymssql 模組範例。
- 進一步了解如何使用 Apache Airflow 參考指南中的 mssql_operator 在特定 Microsoft SQL 資料庫中執行 SQL 程式碼。

搭配使用 Amazon MWAA 與 Amazon EKS

下列範例示範如何使用 Amazon Managed Workflows for Apache Airflow 搭配 Amazon EKS。

主題

- 版本
- 先決條件
- 建立 Amazon EC2 的公有金鑰
- 建立叢集
- 建立mwaa命名空間
- 建立 mwaa 命名空間的角色
- 建立並連接 Amazon EKS 叢集的 IAM 角色

- 建立 requirements.txt 檔案
- 建立 Amazon EKS 的身分映射
- 建立 kubeconfig
- 建立 DAG
- 將 DAG 和 kube_config.yaml 新增至 Amazon S3 儲存貯體
- 啟用並觸發範例

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用本主題中的範例，您需要下列項目：

- Amazon MWAA 環境。
- eksctl。若要進一步了解，請參閱安裝 eksctl。
- kubectl。若要進一步了解，請參閱安裝和設定 kubectl。在某些情況下，這會與 eksctl 一起安裝。
- 在您建立 Amazon MWAA 環境的區域中的 EC2 金鑰對。若要進一步了解，請參閱建立或匯入金鑰對。

> **Note**
>
> 當您使用 `eksctl`命令時，您可以包含 `--profile`來指定預設以外的設定檔。

建立 Amazon EC2 的公有金鑰

使用下列命令，從您的私有金鑰對建立公有金鑰。

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

若要進一步了解，請參閱擷取金鑰對的公有金鑰。

建立叢集

使用下列命令來建立叢集。如果您想要叢集的自訂名稱，或在不同的區域中建立它，請取代名稱和區域值。您必須在建立 Amazon MWAA 環境的相同區域中建立叢集。取代子網路的值，以符合您用於 Amazon MWAA 的 Amazon VPC 網路中的子網路。取代 的值`ssh-public-key`，以符合您使用的金鑰。您可以使用相同區域中來自 Amazon EC2 的現有金鑰，或在建立 Amazon MWAA 環境的相同區域中建立新的金鑰。

```
eksctl create cluster \
--name mwaa-eks \
--region us-west-2 \
--version 1.18 \
--nodegroup-name linux-nodes \
--nodes 3 \
--nodes-min 1 \
--nodes-max 4 \
--with-oidc \
--ssh-access \
--ssh-public-key MyPublicKey \
--managed \
--vpc-public-subnets "subnet-1111111111111111, subnet-2222222222222222" \
--vpc-private-subnets "subnet-33333333333333333, subnet-44444444444444444"
```

完成建立叢集需要一些時間。完成後，您可以驗證叢集已成功建立，並使用下列命令設定 IAM OIDC 提供者：

```
eksctl utils associate-iam-oidc-provider \
--region us-west-2 \
--cluster mwaa-eks \
--approve
```

建立`mwaa`命名空間

確認叢集已成功建立後，請使用下列命令來建立 Pod 的命名空間。

```
kubectl create namespace mwaa
```

建立 mwaa 命名空間的角色

建立命名空間後，請在 EKS 上為可在 MWAA 命名空間中執行 Pod 的 Amazon MWAA 使用者建立角色和角色繫結關係。如果您使用不同的命名空間名稱，請將 中的 mwaa 取代為您使用-n *mwaa*的名稱。

```
cat << EOF | kubectl apply -f - -n mwaa
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: mwaa-role
rules:
  - apiGroups:
  - ""
  - "apps"
  - "batch"
  - "extensions"
resources:
  - "jobs"
  - "pods"
  - "pods/attach"
   - "pods/exec"
  - "pods/log"
  - "pods/portforward"
  - "secrets"
  - "services"
verbs:
  - "create"
  - "delete"
  - "describe"
  - "get"
  - "list"
  - "patch"
  - "update"
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: mwaa-role-binding
  subjects:
    - kind: User
  name: mwaa-service
  roleRef:
```

```
    kind: Role
  name: mwaa-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

執行下列命令，確認新角色可以存取 Amazon EKS 叢集。如果您未使用 *mwaa*，請務必使用正確的名稱：

```
kubectl get pods -n mwaa --as mwaa-service
```

您會收到一則訊息，指出：

```
No resources found in mwaa namespace.
```

建立並連接 Amazon EKS 叢集的 IAM 角色

您必須建立 IAM 角色，然後將其繫結至 Amazon EKS (k8s) 叢集，以便用於透過 IAM 進行身分驗證。此角色僅用於登入叢集，且沒有任何主控台或 API 呼叫的許可。

使用 中的步驟為 Amazon MWAA 環境建立新的角色Amazon MWAA 執行角色。不過，與其建立和連接該主題中所述的政策，請連接下列政策：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
${MWAA_ENV_NAME}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:log-group:airflow-
${MWAA_ENV_NAME}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
```

```
            "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/
${EKS_CLUSTER_NAME}"
        }
    ]
}
```

建立角色之後，請編輯 Amazon MWAA 環境，以使用您建立做為環境執行角色的角色。若要變更角色，請編輯要使用的環境。您可以在許可下選取執行角色。

已知問題：

- 子路徑無法向 Amazon EKS 驗證的角色 ARNs 存在已知問題。解決方法是手動建立服務角色，而不是使用 Amazon MWAA 本身建立的服務角色。若要進一步了解，請參閱 aws-auth configmap 中包含路徑的 角色無法運作
- 如果 IAM 中沒有 Amazon MWAA 服務清單，您需要選擇替代服務政策，例如 Amazon EC2，然後更新角色的信任政策以符合下列項目：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "airflow-env.amazonaws.com",
          "airflow.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

若要進一步了解，請參閱如何搭配 IAM 角色使用信任政策。

建立 requirements.txt 檔案

若要使用本節中的範例程式碼，請確定您已將下列其中一個資料庫選項新增至 `requirements.txt`。若要進一步了解，請參閱 安裝 Python 相依性。

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

建立 Amazon EKS 的身分映射

針對您在下列命令中建立的角色使用 ARN，為 Amazon EKS 建立身分映射。將 Region *us-east-1* 變更為您建立環境的區域。取代角色的 ARN，最後以您環境的執行角色取代 *mwaa-execution-role*。

```
eksctl create iamidentitymapping \
--region us-east-1 \
--cluster mwaa-eks \
--arn arn:aws:iam::123456789012:role/mwaa-execution-role \
--username mwaa-service
```

建立 kubeconfig

使用下列命令來建立 kubeconfig：

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

如果您在執行時使用特定設定檔update-kubeconfig，則需要移除新增至 kube_config.yaml 檔案的 env:區段，以便其與 Amazon MWAA 搭配使用。若要這麼做，請從 檔案刪除下列項目，然後儲存它：

```
env:
 - name: AWS_PROFILE
 value: profile_name
```

建立 DAG

使用下列程式碼範例來建立 Python 檔案，例如 mwaa_pod_example.py DAG 的 。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
 KubernetesPodOperator

default_args = {
```

```
   'owner': 'aws',
   'depends_on_past': False,
   'start_date': datetime(2019, 2, 20),
   'provide_context': True
}

dag = DAG(
   'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                        namespace="mwaa",
                        image="ubuntu:18.04",
                        cmds=["bash"],
                        arguments=["-c", "ls"],
                        labels={"foo": "bar"},
                        name="mwaa-pod-test",
                        task_id="pod-task",
                        get_logs=True,
                        dag=dag,
                        is_delete_operator_pod=False,
                        config_file=kube_config_path,
                        in_cluster=False,
                        cluster_context='aws'
                        )
```

將 DAG 和 **kube_config.yaml** 新增至 Amazon S3 儲存貯體

將您建立的 DAG 和 kube_config.yaml 檔案放入 Amazon MWAA 環境的 Amazon S3 儲存貯體。您可以使用 Amazon S3 主控台或 將檔案放入儲存貯體 AWS Command Line Interface。

啟用並觸發範例

在 Apache Airflow 中，啟用範例，然後觸發該範例。

成功執行並完成之後，請使用下列命令來驗證 Pod：

```
kubectl get pods -n mwaa
```

您可以取得類似下列的輸出：

```
NAME READY STATUS RESTARTS AGE
mwaa-pod-test-aa11bb22cc3344445555666677778888 0/1 Completed 0 2m23s
```

然後，您可以使用下列命令來驗證 Pod 的輸出。將名稱值取代為上一個命令傳回的值：

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc3344445555666677778888
```

使用 連線至 Amazon ECS **ECSOperator**

本主題說明如何使用 從 Amazon MWAA ECSOperator連線至 Amazon Elastic Container Service (Amazon ECS) 容器。在下列步驟中，您將新增必要的許可到環境的執行角色、使用 CloudFormation 範本建立 Amazon ECS Fargate 叢集，最後建立並上傳連線至新叢集的 DAG。

主題

- 版本
- 先決條件
- 許可
- 建立 Amazon ECS 叢集
- 範例程式碼

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

若要使用此頁面上的範例程式碼，您需要下列項目：

- Amazon MWAA 環境。

許可

- 您環境的執行角色需要許可，才能在 Amazon ECS 中執行任務。您可以將 AmazonECS_FullAccess 受 AWS管政策連接至您的執行角色，或建立下列政策並將其連接至您的執行角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ecs-tasks.amazonaws.com"
                }
            }
        }
    ]
}
```

- 除了新增在 Amazon ECS 中執行任務所需的許可之外，您還必須修改 Amazon MWAA 執行角色中的 CloudWatch Logs 政策陳述式，以允許存取 Amazon ECS 任務日誌群組，如下所示。Amazon ECS 日誌群組是由 CloudFormation 範本在 中建立the section called “建立 Amazon ECS 叢集”。

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
```

```
      "logs:GetQueryResults"
    ],
    "Resource": [
      "arn:aws:logs:us-east-1:123456789012:log-group:airflow-environment-name-*",
      "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
    ]
}
```

如需 Amazon MWAA 執行角色以及如何連接政策的詳細資訊，請參閱 執行角色。

建立 Amazon ECS 叢集

使用下列 CloudFormation 範本，您將建置 Amazon ECS Fargate 叢集以搭配 Amazon MWAA 工作流程使用。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的建立任務定義。

1. 使用下列程式碼建立 JSON 檔案，並將其儲存為 `ecs-mwaa-cfn.json`。

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "This template deploys an ECS Fargate cluster with an Amazon
 Linux image as a test for MWAA.",
    "Parameters": {
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "Select a VPC that allows instances access to ECR, as
 used with MWAA."
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "Select at two private subnets in your selected VPC, as
 used with MWAA."
        },
        "SecurityGroups": {
            "Type": "List<AWS::EC2::SecurityGroup::Id>",
            "Description": "Select at least one security group in your selected
 VPC, as used with MWAA."
        }
    },
    "Resources": {
        "Cluster": {
            "Type": "AWS::ECS::Cluster",
```

```
            "Properties": {
                "ClusterName": {
                    "Fn::Sub": "${AWS::StackName}-cluster"
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "Properties": {
                "LogGroupName": {
                    "Ref": "AWS::StackName"
                },
                "RetentionInDays": 30
            }
        },
        "ExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "ecs-tasks.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        },
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Fn::Sub": "${AWS::StackName}-task"
                },
                "Cpu": 2048,
                "Memory": 4096,
                "NetworkMode": "awsvpc",
```

```
                "ExecutionRoleArn": {
                    "Ref": "ExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Fn::Sub": "${AWS::StackName}-container"
                        },
                        "Image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/
amazonlinux:latest",
                        "PortMappings": [
                            {
                                "Protocol": "tcp",
                                "ContainerPort": 8080,
                                "HostPort": 8080
                            }
                        ],
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-region": {
                                    "Ref": "AWS::Region"
                                },
                                "awslogs-group": {
                                    "Ref": "LogGroup"
                                },
                                "awslogs-stream-prefix": "ecs"
                            }
                        }
                    }
                ],
                "RequiresCompatibilities": [
                    "FARGATE"
                ]
            }
        },
        "Service": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "ServiceName": {
                    "Fn::Sub": "${AWS::StackName}-service"
                },
                "Cluster": {
                    "Ref": "Cluster"
```

```
                },
                "TaskDefinition": {
                    "Ref": "TaskDefinition"
                },
                "DesiredCount": 1,
                "LaunchType": "FARGATE",
                "PlatformVersion": "1.3.0",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "Subnets": {
                            "Ref": "SubnetIds"
                        },
                        "SecurityGroups": {
                            "Ref": "SecurityGroups"
                        }
                    }
                }
            }
        }
    }
}
```

2. 在您的命令提示字元中，使用下列 AWS CLI 命令來建立新的堆疊。您必須將`SecurityGroups`和 值取代`SubnetIds`為 Amazon MWAA 環境安全群組和子網路的值。

```
aws cloudformation create-stack \
--stack-name my-ecs-stack --template-body file://ecs-mwaa-cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=your-mwaa-security-group \
ParameterKey=SubnetIds,ParameterValue=your-mwaa-subnet-1\\,your-mwaa-subnet-1 \
--capabilities CAPABILITY_IAM
```

或者，您可以使用下列 shell 指令碼。指令碼會使用 get-environment AWS CLI 命令擷取您環境安全群組和子網路的必要值，然後相應地建立堆疊。若要執行指令碼，請執行下列動作。

a. 複製指令碼，並將指令碼儲存為與 CloudFormation 範本`ecs-stack-helper.sh`相同的目錄。

```
#!/bin/bash

joinByString() {
  local separator="$1"
```

```
  shift
  local first="$1"
  shift
  printf "%s" "$first" "${@/#/$separator}"
}

response=$(aws mwaa get-environment --name $1)

securityGroupId=$(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SecurityGroupIds[]')
subnetIds=$(joinByString '\,' $(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SubnetIds[]'))

aws cloudformation create-stack --stack-name $2 --template-body file://ecs-
cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=$securityGroupId \
ParameterKey=SubnetIds,ParameterValue=$subnetIds \
--capabilities CAPABILITY_IAM
```

b. 使用以下命令執行指令碼。將 environment-name和 取代stack-name為您的資訊。

```
chmod +x ecs-stack-helper.sh
./ecs-stack-helper.bash environment-name stack-name
```

如果成功，您會參考以下顯示新 CloudFormation 堆疊 ID 的輸出。

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

CloudFormation 堆疊完成並 AWS 佈建 Amazon ECS 資源後，您就可以建立和上傳 DAG。

範例程式碼

1. 開啟命令提示，然後導覽至存放 DAG 程式碼的目錄。例如：

```
cd dags
```

2. 複製下列程式碼範例的內容，並在本機儲存為 mwaa-ecs-operator.py，然後將新的 DAG 上傳至 Amazon S3。

```
from http import client
from airflow import DAG
from airflow.providers.amazon.aws.operators.ecs import ECSOperator
from airflow.utils.dates import days_ago
import boto3

CLUSTER_NAME="mwaa-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
 information.
CONTAINER_NAME="mwaa-ecs-test-container" #Replace value for CONTAINER_NAME with
 your information.
LAUNCH_TYPE="FARGATE"

with DAG(
    dag_id = "ecs_fargate_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1)
) as dag:
    client=boto3.client('ecs')
    services=client.list_services(cluster=CLUSTER_NAME,launchType=LAUNCH_TYPE)

 service=client.describe_services(cluster=CLUSTER_NAME,services=services['serviceArns'])

    ecs_operator_task = ECSOperator(
        task_id = "ecs_operator_task",
        dag=dag,
        cluster=CLUSTER_NAME,
        task_definition=service['services'][0]['taskDefinition'],
        launch_type=LAUNCH_TYPE,
        overrides={
            "containerOverrides":[
                {
                    "name":CONTAINER_NAME,
                    "command":["ls", "-l", "/"],
                },
            ],
        },

        network_configuration=service['services'][0]['networkConfiguration'],
        awslogs_group="mwaa-ecs-zero",
```

```
        awslogs_stream_prefix=f"ecs/{CONTAINER_NAME}",
    )
```

> **Note**
>
> 在範例 DAG 中，對於 awslogs_group，您可能需要使用 Amazon ECS 任務日誌群組的名稱來修改日誌群組。此範例假設名為 的日誌群組mwaa-ecs-zero。對於 awslogs_stream_prefix，請使用 Amazon ECS 任務日誌串流字首。此範例假設日誌串流字首 ecs。

3. 執行下列 AWS CLI 命令，將 DAG 複製到您環境的儲存貯體，然後使用 Apache Airflow UI 觸發 DAG。

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，您會在 ecs_fargate_dag DAG 中的 任務日誌ecs_operator_task中取得類似下列的輸出：

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:300}} INFO - Running ECS Task -
Task definition: arn:aws:ecs:us-west-2:123456789012:task-definition/mwaa-ecs-test-
task:1 - on cluster mwaa-ecs-test-cluster
[2022-01-01, 12:00:00 UTC] {{ecs-operator-test.py:302}} INFO - ECSOperator
 overrides:
{'containerOverrides': [{'name': 'mwaa-ecs-test-container', 'command': ['ls', '-l',
 '/']}]}
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:379}} INFO - ECS task ID is:
 e012340b5e1b43c6a757cf012c635935
[2022-01-01, 12:00:00 UTC] {{ecs.py:313}} INFO - Starting ECS Task Log Fetcher
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] total
 52
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 bin -> usr/bin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x   2 root root 4096 Apr  9  2019 boot
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   5 root root  340 Jul 19 17:54 dev
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   1 root root 4096 Jul 19 17:54 etc
```

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 home
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 lib -> usr/lib
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    9 Jun 13 18:51 lib64 -> usr/lib64
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:51 local
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 media
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 mnt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 opt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x 103 root root    0 Jul 19 17:54 proc
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
x-\-\-   2 root root 4096 Apr  9  2019 root
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:52 run
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    8 Jun 13 18:51 sbin -> usr/sbin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 srv
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x  13 root root    0 Jul 19 17:54 sys
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 drwxrwxrwt   2 root root 4096 Jun 13 18:51 tmp
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  13 root root 4096 Jun 13 18:51 usr
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  18 root root 4096 Jun 13 18:52 var
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:328}} INFO - ECS Task has been successfully
 executed
```

搭配 Amazon MWAA 使用 dbt

本主題示範如何搭配 Amazon MWAA 使用 dbt 和 Postgres。在下列步驟中，您將新增必要的相依性到您的 `requirements.txt`，並將範例 dbt 專案上傳至您環境的 Amazon S3 儲存貯體。然後，您將使用範例 DAG 來驗證 Amazon MWAA 已安裝相依性，最後使用 `BashOperator`來執行 dbt 專案。

主題

- 版本
- 先決條件
- 相依性
- 將 dbt 專案上傳至 Amazon S3
- 使用 DAG 驗證 dbt 相依性安裝
- 使用 DAG 執行 dbt 專案

版本

您可以使用此頁面上的程式碼範例搭配 Python 3.10 中的 Apache Airflow v2 和 Python 3.11 中的 Apache Airflow v3。 https://peps.python.org/pep-0619/ https://peps.python.org/pep-0664/

先決條件

在您可以完成下列步驟之前，您將需要下列項目：

- 使用 Apache Airflow 2.2.2 版的 Amazon MWAA 環境。此範例已撰寫，並使用 v2.2.2 進行測試。您可能需要修改範例，以與其他 Apache Airflow 版本搭配使用。
- 範例 dbt 專案。若要開始使用 dbt 搭配 Amazon MWAA，您可以建立分支，並從 dbt-labs GitHub 儲存庫複製 dbt 入門專案。 GitHub

相依性

若要將 Amazon MWAA 與 dbt 搭配使用，請將下列啟動指令碼新增至您的環境。若要進一步了解，請參閱搭配 Amazon MWAA 使用啟動指令碼。

```
#!/bin/bash

  if [[ "${MWAA_AIRFLOW_COMPONENT}" != "worker" ]]
    then
```

```
    exit 0
fi

echo "-----------------------------"
echo "Installing virtual Python env"
echo "-----------------------------"

pip3 install --upgrade pip

echo "Current Python version:"
python3 --version
echo "..."

sudo pip3 install --user virtualenv
sudo mkdir python3-virtualenv
cd python3-virtualenv
sudo python3 -m venv dbt-env
sudo chmod -R 777 *

echo "-----------------------------"
echo "Activating venv in"
$DBT_ENV_PATH
   echo "-----------------------------"

source dbt-env/bin/activate
pip3 list

echo "-----------------------------"
echo "Installing libraries..."
echo "-----------------------------"

# do not use sudo, as it will install outside the venv
pip3 install dbt-redshift==1.6.1 dbt-postgres==1.6.1

echo "-----------------------------"
echo "Venv libraries..."
echo "-----------------------------"

pip3 list
dbt --version

echo "-----------------------------"
echo "Deactivating venv..."
echo "-----------------------------"
```

```
  deactivate
```

在下列各節中，您將上傳 dbt 專案目錄至 Amazon S3，並執行 DAG，以驗證 Amazon MWAA 是否已成功安裝所需的 dbt 相依性。

將 dbt 專案上傳至 Amazon S3

若要將 dbt 專案與 Amazon MWAA 環境搭配使用，您可以將整個專案目錄上傳至環境的`dags`資料夾。當環境更新時，Amazon MWAA 會將 dbt 目錄下載至本機`usr/local/airflow/dags/`資料夾。

將 dbt 專案上傳至 Amazon S3

1. 導覽至您複製 dbt 入門專案的目錄。
2. 執行下列 Amazon S3 AWS CLI command，使用 `--recursive` 參數以遞迴方式將專案內容複製到您環境的`dags`資料夾。命令會建立名為 的子目錄`dbt`，可用於所有 dbt 專案。如果子目錄已存在，專案檔案會複製到現有目錄，而且不會建立新的目錄。命令也會在此特定入門專案的 dbt 目錄中建立子目錄。

   ```
   aws s3 cp dbt-starter-project s3://amzn-s3-demo-bucket/dags/dbt/dbt-starter-project
    --recursive
   ```

   您可以針對專案子目錄使用不同的名稱，在父`dbt`目錄中組織多個 dbt 專案。

使用 DAG 驗證 dbt 相依性安裝

下列 DAG 使用 `BashOperator`和 bash 命令來驗證 Amazon MWAA 是否已成功安裝 中指定的 dbt 相依性`requirements.txt`。

```
from airflow import DAG
   from airflow.operators.bash_operator import BashOperator
   from airflow.utils.dates import days_ago

   with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
   cli_command = BashOperator(
   task_id="bash_command",
   bash_command="/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt --version"
   )
```

執行下列動作來存取任務日誌，並確認已安裝 dbt 及其相依性。

1. 導覽至 Amazon MWAA 主控台，然後從可用環境清單中選擇 Open Airflow UI。
2. 在 Apache Airflow UI 上，從清單中尋找 dbt-installation-test DAG，然後在Last Run欄中選擇日期以開啟最後一個成功任務。
3. 使用圖形檢視，選擇bash_command任務以開啟任務執行個體詳細資訊。
4. 選擇日誌以開啟任務日誌，然後驗證日誌是否成功列出我們在 中指定的 dbt 版本requirements.txt。

使用 DAG 執行 dbt 專案

下列 DAG 使用 BashOperator將上傳到 Amazon S3 的 dbt 專案從本機usr/local/airflow/dags/目錄複製到可寫入/tmp目錄，然後執行 dbt 專案。bash 命令會假設名為 的入門 dbt 專案dbt-starter-project。根據專案目錄的名稱修改目錄名稱。

```
from airflow import DAG
   from airflow.operators.bash_operator import BashOperator
   from airflow.utils.dates import days_ago

   import os

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   # assumes all files are in a subfolder of DAGs called dbt

   with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
   cli_command = BashOperator(
   task_id="bash_command",
   bash_command="source /usr/local/airflow/python3-virtualenv/dbt-env/bin/activate;\
   cp -R /usr/local/airflow/dags/dbt /tmp;\
   echo 'listing project files:';\
   ls -R /tmp;\
   cd /tmp/dbt/mwaa_dbt_test_project;\
   /usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt run --project-dir /tmp/dbt/
mwaa_dbt_test_project --profiles-dir ..;\
   cat /tmp/dbt_logs/dbt.log;\
   rm -rf /tmp/dbt/mwaa_dbt_test_project"
```

```
    )
```

AWS 部落格和教學課程

- 使用 Amazon EKS 和 Amazon MWAA for Apache Airflow v2.x

Amazon Managed Workflows for Apache Airflow 的最佳實務

本指南說明使用 Amazon Managed Workflows for Apache Airflow 時建議的最佳實務。

主題

- Amazon MWAA 上 Apache Airflow 的效能調校
- 在 requirements.txt 中管理 Python 相依性

Amazon MWAA 上 Apache Airflow 的效能調校

本主題說明如何使用 調整 Amazon Managed Workflows for Apache Airflow 環境的效能在 Amazon MWAA 上使用 Apache Airflow 組態選項。

內容

- 新增 Apache Airflow 組態選項
- Apache Airflow 排程器
  - Parameters
  - 限制
- DAG 資料夾
  - Parameters
- DAG 檔案
  - Parameters
- 任務
  - Parameters

新增 Apache Airflow 組態選項

使用下列程序將 Airflow 組態選項新增至您的環境。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。

5. 在 Airflow 組態選項窗格中選擇新增自訂組態。
6. 從下拉式清單中選擇組態並輸入值，或輸入自訂組態並輸入值。
7. 針對您要新增的每個組態，選擇新增自訂組態。
8. 選擇儲存。

若要進一步了解，請參閱 在 Amazon MWAA 上使用 Apache Airflow 組態選項。

Apache Airflow 排程器

Apache Airflow 排程器是 Apache Airflow 的核心元件。排程器的問題可防止剖析 DAGs 和排程任務。如需 Apache Airflow 排程器調校的詳細資訊，請參閱 Apache Airflow 文件網站上的微調排程器效能。

Parameters

本節說明 Apache Airflow 排程器 (Apache Airflow v2 和更新版本） 可用的組態選項及其使用案例。

Apache Airflow v3

Configuration	使用案例
celery.sync_parallelism Celery Executor 用來同步任務狀態的程序數目。 預設值：1	您可以使用此選項來限制 Celery Executor 使用的程序，以防止佇列衝突。根據預設，值會設定為 1 ，以防止將任務日誌交付至 CloudWatch Logs 時發生錯誤。將值設定為 0表示使用最大數量的程序，但在交付任務日誌時可能會導致錯誤。
scheduler.scheduler_idle_sleep_time 排程器「迴圈」中連續 DAG 檔案處理之間的等待秒數。 預設值：1	您可以使用此選項來釋放排程器上的 CPU 用量，方法是增加排程器在完成擷取 DAG 剖析結果後休眠的時間、尋找和佇列任務，以及在執行器中執行排入佇列的任務。增加此值會耗用 `dag_processor.parsing_processes` Apache Airflow v2 和 Apache Airflow v3 在 環境中執行的排程器執行緒數目。這可以減少排程器剖析 DAGs 的容量，並增加 DAGs 在 Web 伺服器中填入所需的時間。

Configuration	使用案例
scheduler.max_dagruns_to_create_per_loop 為每個排程器「迴圈」建立 DagRuns 的 DAGs 數目上限。 預設值：10	您可以使用此選項，藉由減少排程器「迴圈」的 DagRuns 數目上限，釋放用於排程任務的資源。
dag_processor.parsing_processes 排程器可以平行執行以排程 DAGs執行緒數目。 預設：使用 `(2 * number of vCPUs) - 1`	您可以使用此選項，藉由減少排程器平行執行以剖析 DAGs 的程序數目來釋放資源。如果 DAG 剖析影響任務排程，建議您將此數字保持較低。您必須指定小於您環境中 vCPU 計數的值。若要進一步了解，請參閱 限制。

Apache Airflow v2

Configuration	使用案例
celery.sync_parallelism Celery Executor 用來同步任務狀態的程序數目。 預設值：1	您可以使用此選項來限制 Celery Executor 使用的程序，以防止佇列衝突。根據預設，值會設定為 1 ，以防止將任務日誌交付至 CloudWatch Logs 時發生錯誤。將值設定為 0表示使用最大數量的程序，但在交付任務日誌時可能會導致錯誤。
scheduler.idle_sleep_time 排程器「迴圈」中連續 DAG 檔案處理之間的等待秒數。 預設值：1	您可以使用此選項來釋放排程器上的 CPU 用量，方法是增加排程器在完成擷取 DAG 剖析結果後休眠的時間、尋找和佇列任務，以及在執行器中執行排入佇列的任務。增加此值會耗用 `scheduler.parsing_processes` Apache Airflow v2 和 Apache Airflow v3 在環境中執行的排程器執行緒數目。這可以減少排程器剖析 DAGs 的容量，並增加 DAGs 在 Web 伺服器中填入所需的時間。

Configuration	使用案例
scheduler.max_dagruns_to_create_per_loop 為每個排程器「迴圈」建立 DagRuns 的 DAGs 數目上限。 預設值：10	您可以使用此選項，藉由減少排程器「迴圈」的 DagRuns 數目上限，釋放用於排程任務的資源。
scheduler.parsing_processes 排程器可以平行執行以排程 DAGs執行緒數目。 預設：使用 `(2 * number of vCPUs) - 1`	您可以使用此選項，藉由減少排程器平行執行以剖析 DAGs 的程序數目來釋放資源。如果 DAG 剖析影響任務排程，建議您將此數字保持較低。您必須指定小於您環境中 vCPU 計數的值。若要進一步了解，請參閱 限制。

限制

本節說明調整排程器的預設參數時應考慮的限制。

scheduler.parsing_processes、scheduler.max_threads （僅限 v2)

環境類別的每個 vCPU 允許兩個執行緒。環境類別的排程器必須至少保留一個執行緒。如果您注意到任務排程延遲，您可能需要增加環境類別。例如，大型環境的排程器具有 4 個 vCPU Fargate 容器執行個體。這表示執行緒7總數上限為 ，可用於其他程序。也就是說，兩個執行緒會乘以四個 vCPUs，減去排程器本身的一個 vCPU。您在 `scheduler.max_threads` (僅限 v2) 中指定的值，且`scheduler.parsing_processes`不得超過 環境類別可用的執行緒數目，如下所示：

- mw1.small – 不得超過其他程序的1執行緒。剩餘的執行緒會保留給排程器。
- mw1.medium – 不得超過其他程序的3執行緒。剩餘的執行緒會保留給排程器。
- mw1.large – 不得超過其他程序的7執行緒。剩餘的執行緒會保留給排程器。

DAG 資料夾

Apache Airflow 排程器會持續掃描您環境中DAGs 資料夾。任何包含`plugins.zip`的檔案，或包含「airflow」匯入陳述式的 Python (`.py`) 檔案。然後，任何產生的 Python DAG 物件都會放入 DagBag

中，以供排程器處理該檔案，以決定需要排程哪些任務。無論檔案是否包含任何可行的 DAG 物件，都會進行大型檔案剖析。

Parameters

本節說明 DAGs (Apache Airflow v2 及更新版本） 可用的組態選項及其使用案例。

Apache Airflow v3

Configuration	使用案例
dag_processor.refresh_interval 必須掃描 DAGs是否有新檔案的秒數。 預設：300 秒	您可以使用此選項，透過增加剖析 DAGs秒數來釋放資源。如果您在 中遇到長時間的剖析時間`total_parse_time metrics` ，建議您增加此值，這可能是因為 DAGs資料夾中有大量檔案所致。
dag_processor.min_file_process_interval 排程器剖析 DAG 並反映 DAG 更新的秒數。 預設：30 秒	您可以使用此選項，透過增加排程器在剖析 DAG 之前等待的秒數來釋放資源。例如，如果您指定 的值`30`，則會每 30 秒剖析一次 DAG 檔案。我們建議您將此數字保持高，以減少環境中的 CPU 用量。

Apache Airflow v2

Configuration	使用案例
scheduler.dag_dir_list_interval 必須掃描 DAGs是否有新檔案的秒數。 預設：300 秒	您可以使用此選項，透過增加剖析 DAGs秒數來釋放資源。如果您在 中遇到長時間的剖析時間`total_parse_time metrics` ，建議您增加此值，這可能是因為 DAGs資料夾中有大量檔案所致。
scheduler.min_file_process_interval	您可以使用此選項，透過增加排程器在剖析 DAG 之前等待的秒數來釋放資源。例如，如果

Configuration	使用案例
排程器剖析 DAG 並反映 DAG 更新的秒數。 預設：30 秒	您指定 的值30，則會每 30 秒剖析一次 DAG 檔案。我們建議您將此數字保持高，以減少環境中的 CPU 用量。

DAG 檔案

做為 Apache Airflow 排程器迴圈的一部分，會剖析個別 DAG 檔案以擷取 DAG Python 物件。在 Apache Airflow v2 和更新版本中，排程器會同時剖析最多 個剖析程序。`scheduler.min_file_process_interval` (v2) 或 `dag_processor.min_file_process_interval`(v3) 中指定的秒數必須先通過，才能再次剖析相同的檔案。

Parameters

本節說明 Apache Airflow DAG 檔案 (Apache Airflow v2 及更新版本） 可用的組態選項及其使用案例。

Apache Airflow v3

Configuration	使用案例
dag_processor.dag_file_processor_timeout DagFileProcessor 逾時處理 DAG 檔案之前的秒數。 預設：50 秒	您可以使用此選項，透過增加 DagFilePr ocessor 逾時之前所需的時間來釋放資源。如果您在 DAG 處理日誌中遇到逾時，導致沒有載入可行DAGs，建議您增加此值。
core.dagbag_import_timeout 匯入 Python 檔案逾時之前的秒數。 預設：30 秒	您可以使用此選項，藉由在匯入 Python 檔案以擷取 DAG 物件時，增加排程器逾時之前所花費的時間來釋放資源。此選項會做為排程器「迴圈」的一部分處理，且必須包含小於 中指定值的值`dag_processor.dag_file_processor_timeout` 。
core.min_serialized_dag_update_interval	

Configuration	使用案例
更新資料庫中序列化 DAGs的最小秒數。 預設：30	您可以使用此選項，透過增加資料庫中序列化 DAGs更新後的秒數來釋放資源。如果您有大量 DAGs 或複雜的 DAGs，建議您增加此值。隨著 DAGs 序列化，增加此值可減少排程器和資料庫的負載。
core.min_serialized_dag_fetch_interval 當已載入 DagBag 時，從資料庫重新擷取序列化 DAG 的秒數。 預設值：10	您可以使用此選項，透過增加重新擷取序列化 DAG 的秒數來釋放資源。值必須大於 中指定的值core.min_serialized_dag_update_interval ，以降低資料庫「寫入」速率。隨著 DAGs序列化，增加此值可減少 Web 伺服器和資料庫的負載。

Apache Airflow v2

Configuration	使用案例
core.dag_file_processor_timeout DagFileProcessor 逾時處理 DAG 檔案之前的秒數。 預設：50 秒	您可以使用此選項，透過增加 DagFileProcessor 逾時之前所需的時間來釋放資源。如果您在 DAG 處理日誌中遇到逾時，導致沒有載入可行DAGs，建議您增加此值。
core.dagbag_import_timeout 匯入 Python 檔案逾時之前的秒數。 預設：30 秒	您可以使用此選項，藉由在匯入 Python 檔案以擷取 DAG 物件時，增加排程器逾時之前所花費的時間來釋放資源。此選項會做為排程器「迴圈」的一部分處理，且必須包含小於 中指定值的值core.dag_file_processor_timeout 。
core.min_serialized_dag_update_interval	您可以使用此選項，透過增加資料庫中序列化 DAGs更新後的秒數來釋放資源。如果您有大

Configuration	使用案例
更新資料庫中序列化 DAGs的最小秒數。 預設：30	量 DAGs 或複雜的 DAGs，建議您增加此值。隨著 DAGs 序列化，增加此值可減少排程器和資料庫的負載。
core.min_serialized_dag_fetch_interval 當已載入 DagBag 時，從資料庫重新擷取序列化 DAG 的秒數。 預設值：10	您可以使用此選項，透過增加重新擷取序列化 DAG 的秒數來釋放資源。值必須大於 中指定的值`core.min_serialized_dag_update_interval`，以降低資料庫「寫入」速率。隨著 DAGs序列化，增加此值可減少 Web 伺服器和資料庫的負載。

任務

Apache Airflow 排程器和工作者都參與佇列和取消佇列任務。排程器會將準備從無狀態排程的剖析任務轉換為已排程狀態。執行器也會在 Fargate 的排程器容器上執行，將這些任務排入佇列，並將其狀態設定為已排入佇列。當工作者有容量時，會從佇列中取得任務，並將狀態設定為執行中，然後根據任務是否成功，將其狀態變更為成功或失敗。

Parameters

本節說明 Apache Airflow 任務可用的組態選項及其使用案例。

Amazon MWAA 覆寫的預設組態選項會以##標示。

Apache Airflow v3

Configuration	使用案例
core.parallelism 可具有 Running 狀態的任務執行個體數目上限。 預設：根據 動態設定`(maxWorkers * maxCeleryWorkers) / schedulers * 1.5`。	您可以使用此選項，透過增加可同時執行的任務執行個體數量來釋放資源。指定的值必須是可用工作者的數量乘以工作者的任務密度。我們建議您只有在遇到大量任務卡在「執行中」或「佇列」狀態時，才變更此值。

Configuration	使用案例
core.execute_tasks_new_python_interpreter 決定 Apache Airflow 是否透過強制執行父程序或建立新的 Python 程序來執行任務。 預設：`True`	設定為 時`True`，Apache Airflow 會將您對外掛程式所做的變更辨識為新的 Python 程序，以便建立以執行任務。
celery.worker_concurrency Amazon MWAA 會覆寫此選項的 Airflow 基本安裝，以擴展工作者作為其自動擴展元件的一部分。 預設：不適用	`################`
celery.worker_autoscale 工作者的任務並行。 預設值： • mw1.micro - 3，0 • mw1.small - 5，0 • mw1.medium - 10，0 • mw1.large - 20，0 • mw1.xlarge - 40，0 • mw1.2xlarge - 80，0	您可以使用此選項，透過減少工作者的任務`minimum`並行`maximum`來釋放資源。工作者最多接受設定的`maximum`並行任務，無論是否有足夠資源可以這樣做。如果任務排程時沒有足夠的資源，任務會立即失敗。我們建議將此值變更為資源密集型任務，方法是將值減少為小於預設值，以允許每個任務更多的容量。

Apache Airflow v2

Configuration	使用案例

Configuration	使用案例
core.parallelism 可具有 Running 狀態的任務執行個體數目上限。 預設：根據 動態設定(maxWorkers * maxCeleryWorkers) / schedulers * 1.5。	您可以使用此選項，透過增加可同時執行的任務執行個體數量來釋放資源。指定的值必須是可用工作者的數量乘以工作者的任務密度。我們建議您只有在遇到大量任務卡在「執行中」或「佇列」狀態時，才變更此值。
core.dag_concurrency 允許為每個 DAG 同時執行的任務執行個體數目。 預設：10000	您可以使用此選項，透過增加允許同時執行的任務執行個體數量來釋放資源。例如，如果您有 100 個具有 10 個平行任務DAGs，而且您希望所有 DAGs 同時執行，則可以計算最大平行處理，將可用工作者數量乘以 中的工作者任務密度celery.worker_concurrency ，再除以 DAGs 數量。
core.execute_tasks_new_python_interpreter 決定 Apache Airflow 是否透過強制執行父程序或建立新的 Python 程序來執行任務。 預設：True	設定為 時True，Apache Airflow 會將您對外掛程式所做的變更辨識為新的 Python 程序，以便建立以執行任務。
celery.worker_concurrency Amazon MWAA 會覆寫此選項的 Airflow 基本安裝，以擴展工作者作為其自動擴展元件的一部分。 預設：不適用	################

Configuration	使用案例
celery.worker_autoscale 工作者的任務並行。 預設值： • mw1.micro - 3，0 • mw1.small - 5，0 • mw1.medium - 10，0 • mw1.large - 20，0 • mw1.xlarge - 40，0 • mw1.2xlarge - 80，0	您可以使用此選項，透過減少工作者的任務minimum並行maximum來釋放資源。工作者最多接受設定的maximum並行任務，無論是否有足夠資源可以這樣做。如果任務排程時沒有足夠的資源，任務會立即失敗。我們建議將此值變更為資源密集型任務，方法是將值減少為小於預設值，以允許每個任務更多的容量。

在 requirements.txt 中管理 Python 相依性

本主題說明如何在 Amazon Managed Workflows for Apache Airflow 環境的 `requirements.txt` 檔案中安裝和管理 Python 相依性。

內容

- 使用 Amazon MWAA CLI 公用程式測試 DAGs
- 使用 PyPi.org 要求檔案格式安裝 Python 相依性
  - 選項一：Python 套件索引中的 Python 相依性
  - 選項二：Python wheel (.whl)
    - 在 Amazon S3 儲存貯體上使用 plugins.zip 檔案
    - 使用 URL 上託管的 WHL 檔案
    - 從 DAG 建立 WHL 檔案
  - 選項三：託管在私有 PyPi/PEP-503 相容儲存庫上的 Python 相依性
- 在 Amazon MWAA 主控台上啟用日誌
- 在 CloudWatch Logs 主控台上存取日誌
- 在 Apache Airflow UI 中存取錯誤

  - 登入 Apache Airflow
- 範例requirements.txt案例

使用 Amazon MWAA CLI 公用程式測試 DAGs

- 命令列界面 (CLI) 公用程式會在本機複寫 Amazon Managed Workflows for Apache Airflow 環境。
- CLI 會在本機建置類似於 Amazon MWAA 生產映像的 Docker 容器映像。您可以使用它來執行本機 Apache Airflow 環境，以在部署到 Amazon MWAA 之前開發和測試 DAGs、自訂外掛程式和相依性。
- 若要執行 CLI，請參閱 GitHub 上的 aws-mwaa-docker-images。

使用 PyPi.org 要求檔案格式安裝 Python 相依性

下一節說明根據 PyPi.org 需求檔案格式安裝 Python 相依性的不同方式。

選項一：Python 套件索引中的 Python 相依性

下一節說明如何從 `requirements.txt` 檔案中的 Python 套件索引指定 Python 相依性。

Apache Airflow v3

1. 在本機測試。在建立`requirements.txt`檔案之前，反覆新增其他程式庫，以尋找套件及其版本的正確組合。若要執行 Amazon MWAA CLI 公用程式，請參閱 GitHub 上的 aws-mwaa-docker-images。
2. 檢閱 Apache Airflow 套件額外項目。若要存取 Amazon MWAA 上為 Apache Airflow v3 安裝的套件清單，請參閱 GitHub 網站上的 aws-mwaa-docker-images`requirements.txt`。
3. 新增限制條件陳述式。在檔案頂端新增 Apache Airflow v3 環境的限制條件`requirements.txt`檔案。Apache Airflow 限制條件檔案會指定 Apache Airflow 發行時可用的提供者版本。

   在下列範例中，將 *`{environment-version}`* 取代為您環境的版本編號，並將 *`{Python-version}`* 取代為您環境相容的 Python 版本。

   如需有關與 Apache Airflow 環境相容的 Python 版本的資訊，請參閱 Apache Airflow 版本。

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

如果限制條件檔案判斷`xyz==1.0`套件與您環境中的其他套件不相容， `pip3 install` 無法防止不相容的程式庫安裝到您的環境。如果任何套件的安裝失敗，您可以在 CloudWatch Logs 的對應日誌串流中存取每個 Apache Airflow 元件（排程器、工作者和 Web 伺服器）的錯誤日誌。如需日誌類型的詳細資訊，請參閱 the section called “存取 Airflow 日誌”。

4. Apache Airflow 套件。新增套件額外項目和版本 (==)。這有助於防止相同名稱但不同版本的套件安裝在您的環境中。

```
apache-airflow[package-extra]==2.5.1
```

5. Python 程式庫。在 `requirements.txt` 檔案中新增套件名稱和版本 (==)。這有助於防止 PyPi.org 未來的重大更新自動套用。

```
library == version
```

Example Boto3 和 psycopg2-binary

此範例僅供示範之用。boto 和 psycopg2-binary 程式庫包含在 Apache Airflow v3 的基本安裝中，不需要在`requirements.txt`檔案中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的套件沒有版本，Amazon MWAA 會從 PyPi.org 安裝最新版本的套件。此版本可能會與您 中的其他套件衝突`requirements.txt`。

Apache Airflow v2

1. 在本機測試。在建立`requirements.txt`檔案之前，反覆新增其他程式庫，以尋找套件及其版本的正確組合。若要執行 Amazon MWAA CLI 公用程式，請參閱 GitHub 上的 aws-mwaa-docker-images。
2. 檢閱 Apache Airflow 套件額外項目。若要存取 Amazon MWAA 上為 Apache Airflow v2 安裝的套件清單，請存取 GitHub 網站上的 aws-mwaa-docker-images`requirements.txt`。
3. 新增限制條件陳述式。在檔案頂端新增 Apache Airflow v2 環境的限制條件`requirements.txt`檔案。Apache Airflow 限制條件檔案會指定 Apache Airflow 發行時可用的提供者版本。

從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制條件，Amazon MWAA 會為您指定一個限制條件，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。

在下列範例中，將 *{environment-version}* 取代為您環境的版本編號，並將 *{Python-version}* 取代為您環境相容的 Python 版本。

如需有關與 Apache Airflow 環境相容的 Python 版本的資訊，請參閱 Apache Airflow 版本。

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

如果限制條件檔案判斷`xyz==1.0`套件與您環境中的其他套件不相容， `pip3 install` 無法防止不相容的程式庫安裝到您的環境。如果任何套件的安裝失敗，您可以在 CloudWatch Logs 的對應日誌串流中存取每個 Apache Airflow 元件（排程器、工作者和 Web 伺服器）的錯誤日誌。如需日誌類型的詳細資訊，請參閱 the section called “存取 Airflow 日誌”。

4. Apache Airflow 套件。新增套件額外項目和版本 (==)。這有助於防止相同名稱但不同版本的套件安裝在您的環境中。

```
apache-airflow[package-extra]==2.5.1
```

5. Python 程式庫。在 `requirements.txt` 檔案中新增套件名稱和版本 (==)。這有助於防止 PyPi.org 未來的重大更新自動套用。

```
library == version
```

Example Boto3 和 psycopg2-binary

此範例僅供示範之用。boto 和 psycopg2-binary 程式庫包含在 Apache Airflow v2 基本安裝中，不需要在`requirements.txt`檔案中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的套件沒有版本，Amazon MWAA 會從 PyPi.org 安裝最新版本的套件。此版本可能會與您 中的其他套件衝突`requirements.txt`。

選項二：Python wheel (.whl)

Python wheel 是一種套件格式，旨在使用編譯的成品來運送程式庫。輪子套件做為在 Amazon MWAA 中安裝相依性的方法，有幾個好處：

- 更快速的安裝 – WHL 檔案會以單一 ZIP 的形式複製到容器，然後在本機安裝，無需下載每個檔案。
- 較少衝突 – 您可以事先判斷套件的版本相容性。因此， 不需要pip遞迴處理相容的版本。
- 更高的彈性 – 使用外部託管程式庫時，下游需求可能會變更，導致 Amazon MWAA 環境中容器之間的版本不相容。透過不依賴相依性的外部來源， 上的每個容器都有相同的程式庫，無論每個容器何時執行個體化。

我們建議您使用下列方法來從 中的 Python wheel 封存檔 (`.whl`) 安裝 Python 相依性`requirements.txt`。

方法

- 在 Amazon S3 儲存貯體上使用 plugins.zip 檔案
- 使用 URL 上託管的 WHL 檔案
- 從 DAG 建立 WHL 檔案

在 Amazon S3 儲存貯體上使用 **`plugins.zip`** 檔案

Apache Airflow 排程器、工作者和 webserver （適用於 Apache Airflow 2.2.2 版及更新版本）會在 中為您的環境在 AWS受管 Fargate 容器上啟動期間搜尋自訂外掛程式。`/usr/local/airflow/plugins/*`此程序會在 Python 相依性和 Apache Airflow 服務啟動`pip3 install -r requirements.txt`的 Amazon MWAA 之前開始。`plugins.zip` 檔案可用於您不想要在環境執行期間持續變更的任何檔案，或者您不想將存取權授予寫入 DAGs的使用者。例如，Python 程式庫 wheel 檔案、憑證 PEM 檔案和組態 YAML 檔案。

下一節說明如何在 Amazon S3 儲存貯體的 `plugins.zip` 檔案中安裝輪子。

1. 下載必要的 WHL 檔案 您可以`pip download`與 Amazon MWAA aws-mwaa-docker-images 或其他 Amazon Linux 2 容器`requirements.txt`上的現有 搭配使用，以解析和下載必要的 Python wheel 檔案。

```
pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
cd "$AIRFLOW_HOME/plugins"
zip "$AIRFLOW_HOME/plugins.zip" *
```

2. 在 中指定路徑**requirements.txt**。使用 指定您 requirements.txt 頂端的外掛程式目錄，`--find-links`並指示pip不要使用 從其他來源安裝 `--no-index`，如下列程式碼所列：

```
--find-links /usr/local/airflow/plugins
--no-index
```

Example requirements.txt 中的 wheel

下列範例假設您已在 Amazon S3 儲存貯體根目錄中的`plugins.zip`檔案中上傳輪子。例如：

```
--find-links /usr/local/airflow/plugins
--no-index

numpy
```

Amazon MWAA 會從 `plugins` 資料夾擷取`numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl`輪子，並將其安裝在您的環境中。

使用 URL 上託管的 WHL 檔案

下一節說明如何安裝託管在 URL 上的滾輪。URL 必須可公開存取，或從您為 Amazon MWAA 環境指定的自訂 Amazon VPC 中存取。

- 提供 URL。將 URL 提供給 中的滾輪`requirements.txt`。

Example公有 URL 上的車輪封存

下列範例會從公有網站下載輪子。

```
--find-links https://files.pythonhosted.org/packages/
--no-index
```

Amazon MWAA 會從您指定的 URL 擷取輪子，並將其安裝在您的環境中。

> **Note**
>
> URLs。

從 DAG 建立 WHL 檔案

如果您有使用 Apache Airflow 2.2.2 版或更新版本的私有 Web 伺服器，而且由於您的環境無法存取外部儲存庫，因此無法安裝需求，您可以使用下列 DAG 來接受現有的 Amazon MWAA 需求，並將其封裝在 Amazon S3 上：

```
from airflow import DAG
 from airflow.operators.bash_operator import BashOperator
 from airflow.utils.dates import days_ago

 S3_BUCKET = 'my-s3-bucket'
 S3_KEY = 'backup/plugins_whl.zip'

 with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
 cli_command = BashOperator(
 task_id="bash_command",
 bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/requirements/
requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3 cp /tmp/
plugins.zip s3://amzn-s3-demo-bucket/{S3_KEY}"
 )
```

執行 DAG 之後，請`plugins.zip`選擇性地使用此新檔案做為您的 Amazon MWAA，並與其他外掛程式一起封裝。然後，以 `--find-links /usr/local/airflow/plugins` 和 更新您的`requirements.txt`前綴，`--no-index`而不新增 `--constraint`。

此方法可讓您離線使用相同的程式庫。

選項三：託管在私有 PyPi/PEP-503 相容儲存庫上的 Python 相依性

下一節說明如何安裝在具有身分驗證的私有 URL 上託管的 Apache Airflow 額外項目。

1. 將您的使用者名稱和密碼新增為 Apache Airflow 組態選項。例如：
   - `foo.user` : *YOUR_USER_NAME*
   - `foo.pass` : *YOUR_PASSWORD*
2. 建立您的 `requirements.txt` 檔案。將下列範例中的預留位置替換為您的私有 URL，以及您新增為 Apache Airflow 組態選項的使用者名稱和密碼。例如：

   ```
   --index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
   ```

3. 將任何其他程式庫新增至您的 `requirements.txt` 檔案。例如：

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
my-private-package==1.2.3
```

在 Amazon MWAA 主控台上啟用日誌

Amazon MWAA 環境的執行角色需要許可，才能將日誌傳送至 CloudWatch Logs。若要更新執行角色的許可，請參閱 Amazon MWAA 執行角色。

您可以在 INFO、ERROR、 或 CRITICAL層級啟用 Apache Airflow WARNING日誌。當您選擇日誌層級時，Amazon MWAA 會傳送該層級和所有較高嚴重性層級的日誌。例如，如果您在 INFO層級啟用日誌，Amazon MWAA 會將INFO日誌和 ERROR、 WARNING和 CRITICAL日誌層級傳送至 CloudWatch Logs。我們建議在排程器的 INFO層級啟用 Apache Airflow 日誌，以存取 收到的日誌`requirements.txt`。

在 CloudWatch Logs 主控台上存取日誌

您可以存取排程器的 Apache Airflow 日誌來排程工作流程和剖析dags資料夾。下列步驟說明如何在 Amazon MWAA 主控台上開啟排程器的日誌群組，以及在 CloudWatch Logs 主控台上存取 Apache Airflow 日誌。

存取 的日誌 **`requirements.txt`**

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在監控窗格中選擇 Airflow 排程器日誌群組。
4. 在`requirements_install_ip`日誌串流中選擇日誌。
5. 請參閱 環境上安裝的套件清單`/usr/local/airflow/.local/bin`。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 檢閱套件清單，以及是否有任何套件在安裝期間發生錯誤。如果發生錯誤，您可能會收到類似以下的錯誤：

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

在 Apache Airflow UI 中存取錯誤

您也可以檢查 Apache Airflow UI，以識別錯誤是否與另一個問題相關。使用 Amazon MWAA 上的 Apache Airflow 時最常遇到的錯誤是：

```
Broken DAG: No module named x
```

如果您在 Apache Airflow UI 中發現此錯誤，則檔案中可能會缺少必要的相依性`requirements.txt`。

登入 Apache Airflow

您需要 AWS 帳戶 in AWS Identity and Access Management (IAM) 存取 Apache Airflow UI 的Apache Airflow UI 存取政策：AmazonMWAAWebServerAccess許可。

存取您的 Apache Airflow UI

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇開啟氣流使用者介面。

範例`requirements.txt`案例

您可以在 中混合和比對不同的格式`requirements.txt`。下列範例使用不同方式的組合來安裝額外項目。

Example PyPi.org 上的額外項目和公有 URL

從 PyPi.org 指定套件時，除了公有 URL 上的套件，例如自訂 PEP 503 相容儲存庫 URLs，您還需要使用 `--index-url`選項。 PyPi.org,

```
aws-batch == 0.6
    phoenix-letter >= 0.3

    --index-url http://dist.repoze.org/zope2/2.10/simple
    zopelib
```

Amazon Managed Workflows for Apache Airflow 的監控和指標

監控是維護 Amazon Managed Workflows for Apache Airflow 和您的 AWS 解決方案可靠性、可用性和效能的重要部分。我們建議您從 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。本主題說明 AWS Amazon MWAA 環境監控和回應潛在事件的資源。

> **Note**
>
> Apache Airflow 指標和記錄受標準 Amazon CloudWatch 定價約束。

如需監控 Apache Airflow 的詳細資訊，請參閱 Apache Airflow 文件網站上的記錄和監控。

章節

- Amazon MWAA 的監控概觀
- 在 中存取稽核日誌 AWS CloudTrail
- 在 Amazon CloudWatch 中存取 Airflow 日誌
- 在 Amazon MWAA 上監控儀表板和警示
- CloudWatch 中的 Apache Airflow 環境指標
- Amazon MWAA 的容器、佇列和資料庫指標

Amazon MWAA 的監控概觀

此頁面說明用來監控 Amazon Managed Workflows for Apache Airflow 環境 AWS 的服務。

內容

- Amazon CloudWatch 概觀
- AWS CloudTrail 概觀

Amazon CloudWatch 概觀

CloudWatch 是 AWS 服務的指標儲存庫，可用來根據服務發佈的指標和維度擷取統計資料。您可以使用這些指標來設定警示、計算統計資料，然後在儀表板中呈現資料，協助您在 Amazon CloudWatch 主控台中評估環境的運作狀態。

Apache Airflow 已設定為將 Amazon Managed Workflows for Apache Airflow 環境的 StatsD 指標傳送至 Amazon CloudWatch。

若要進一步了解，請參閱什麼是 Amazon CloudWatch？。

AWS CloudTrail 概觀

CloudTrail 是一種稽核服務，可提供 Amazon MWAA AWS 中使用者、角色或服務所採取動作的記錄。您可以使用 CloudTrail 所收集的資訊，判斷對 Amazon MWAA 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及稽核日誌中提供的其他詳細資訊。

若要進一步了解，請參閱什麼是 AWS CloudTrail？。

在 中存取稽核日誌 AWS CloudTrail

AWS CloudTrail 當您建立 AWS 帳戶 時， 會在您的 上啟用 。CloudTrail 會記錄 IAM 實體或服務所採取的活動 AWS ，例如 Amazon Managed Workflows for Apache Airflow，這會記錄為 CloudTrail 事件。您可以在 CloudTrail 主控台中檢視、搜尋和下載過去 90 天的事件歷史記錄。CloudTrail 會擷取 Amazon MWAA 主控台上的所有事件，以及對 Amazon MWAA APIs 的所有呼叫。它不會擷取唯讀動作，例如 `GetEnvironment`或 `PublishMetrics`動作。此頁面說明如何使用 CloudTrail 監控 Amazon MWAA 的事件。

內容

- 在 CloudTrail 中建立追蹤
- 使用 CloudTrail 事件歷史記錄存取事件
- 的範例追蹤 CreateEnvironment
- 後續步驟？

在 CloudTrail 中建立追蹤

您需要建立追蹤來存取 中事件的持續記錄 AWS 帳戶，包括 Amazon MWAA 的事件。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。如果您未建立追蹤，您仍然可以在 CloudTrail 主

控台中存取可用的事件歷史記錄。例如，使用 CloudTrail 收集的資訊，您可以判斷對 Amazon MWAA 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。若要進一步了解，請參閱為您的 建立追蹤 AWS 帳戶。

使用 CloudTrail 事件歷史記錄存取事件

您可以在 CloudTrail 主控台中檢視事件歷史記錄，對過去 90 天內的操作和安全事件進行故障診斷。例如，您可以存取 AWS 帳戶 與每個區域 中資源 （例如 IAM 使用者或其他 AWS 資源） 建立、修改或刪除相關的事件。若要進一步了解，請參閱使用 CloudTrail 事件歷史記錄存取事件。

1. 開啟 CloudTrail 主控台。
2. 選擇事件歷史記錄。
3. 選取您要檢視的事件，然後選擇比較事件詳細資訊。

的範例追蹤 `CreateEnvironment`

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。事件代表來自任何來源的單一請求，並包含所請求動作的相關資訊，例如動作的日期和時間，或請求參數。CloudTrail 日誌檔案不是公有 API 呼叫的排序堆疊追蹤，也不會以任何特定順序列出。下列範例是 `CreateEnvironment`動作的日誌項目，因為缺少許可而遭到拒絕。中的值`AirflowConfigurationOptions`已針對隱私權進行修訂。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "00123456ABC7DEF8HIJK",
    "arn": "arn:aws:sts::012345678901:assumed-role/root/myuser",
    "accountId": "012345678901",
    "accessKeyId": "",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "00123456ABC7DEF8HIJK",
        "arn": "arn:aws:iam::012345678901:role/user",
        "accountId": "012345678901",
        "userName": "user"
      },
      "webIdFederationData": {},
```

```
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-10-07T15:51:52Z"
      }
    }
  },
  "eventTime": "2020-10-07T15:52:58Z",
  "eventSource": "airflow.amazonaws.com",
  "eventName": "CreateEnvironment",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/7.26.5",
  "errorCode": "AccessDenied",
  "requestParameters": {
    "SourceBucketArn": "arn:aws:s3:::my-bucket",
    "ExecutionRoleArn": "arn:aws:iam::012345678901:role/AirflowTaskRole",
    "AirflowConfigurationOptions": "***",
    "DagS3Path": "sample_dag.py",
    "NetworkConfiguration": {
      "SecurityGroupIds": [
      "sg-01234567890123456"
      ],
      "SubnetIds": [
        "subnet-01234567890123456",
        "subnet-65432112345665431"
      ]
    },
    "Name": "test-cloudtrail"
  },
  "responseElements": {
    "message": "Access denied."
  },
  "requestID": "RequestID",
  "eventID": "EventID",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678901"
}
```


後續步驟？

- 了解如何為 CloudTrail 支援的 AWS 服務和整合中的 CloudTrail 日誌中收集的事件資料設定其他服務。 CloudTrail

- 了解如何在設定 CloudTrail 的 Amazon SNS 通知中，在 CloudTrail 將新的日誌檔案發佈至 Amazon S3 儲存貯體時收到通知 Amazon SNS CloudTrail。

在 Amazon CloudWatch 中存取 Airflow 日誌

Amazon MWAA 可以將 Apache Airflow 日誌傳送至 Amazon CloudWatch。您可以從單一位置存取多個環境的日誌，輕鬆識別 Apache Airflow 任務延遲或工作流程錯誤，而無需額外的第三方工具。必須在 Amazon Managed Workflows for Apache Airflow 主控台上啟用 Apache Airflow 日誌，才能存取 CloudWatch 中的 Apache Airflow DAG 處理、任務、Web 伺服器、工作者日誌。

內容

- 定價
- 開始之前
- 日誌類型
- 啟用 Apache Airflow 日誌
- 存取 Apache Airflow 日誌
- 排程器日誌範例
- 後續步驟？

定價

- 需支付標準 CloudWatch Logs 費用。如需詳細資訊，請參閱 CloudWatch 定價。

開始之前

- 您必須擁有可存取 CloudWatch 中日誌的角色。如需詳細資訊，請參閱 存取 Amazon MWAA 環境。

日誌類型

Amazon MWAA 會為您啟用的每個 Airflow 記錄選項建立日誌群組，並將日誌推送至與環境相關聯的 CloudWatch Logs 群組。日誌群組會以下列格式命名：`YourEnvironmentName-`**`LogType`**。例如，如果您的環境名為 `Airflow-v202-Public`，Apache Airflow 任務日誌會傳送至 `Airflow-v202-Public-`**`Task`**。

日誌類型	Description
YourEnvironmentName-**DAGProcessing**	DAG 處理器管理員（處理 DAG 檔案之排程器的一部分）的日誌。
YourEnvironmentName-**Scheduler**	Airflow 排程器產生的日誌。
YourEnvironmentName-**Task**	DAG 產生的任務日誌。
YourEnvironmentName-**WebServer**	Airflow Web 界面產生的日誌。
YourEnvironmentName-**Worker**	在工作流程和 DAG 執行中產生的日誌。

啟用 Apache Airflow 日誌

您可以在 INFO、ERROR、 或 CRITICAL層級啟用 Apache Airflow WARNING日誌。當您選擇日誌層級時，Amazon MWAA 會傳送該層級和所有較高嚴重性層級的日誌。例如，如果您在 INFO層級啟用日誌，Amazon MWAA 會將INFO日誌和 ERROR、 WARNING和 CRITICAL日誌層級傳送至 CloudWatch Logs。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 選擇編輯。
4. 選擇下一步。
5. 選擇下列一或多個記錄選項：
    a. 在監控窗格中選擇 Airflow 排程器日誌群組。
    b. 在監控窗格中選擇 Airflow Web 伺服器日誌群組。
    c. 在監控窗格中選擇 Airflow 工作者日誌群組。
    d. 在監控窗格中選擇 Airflow DAG 處理日誌群組。
    e. 在監控窗格中選擇 Airflow 任務日誌群組。
    f. 在日誌層級中選擇日誌層級。
6. 選擇下一步。
7. 選擇儲存。

存取 Apache Airflow 日誌

下一節說明如何在 CloudWatch 主控台中存取 Apache Airflow 日誌。

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在監控窗格中選擇日誌群組。
4. 在日誌串流中選擇日誌。

排程器日誌範例

您可以存取排程器的 Apache Airflow 日誌來排程工作流程和剖析dags資料夾。下列步驟說明如何在 Amazon MWAA 主控台上開啟排程器的日誌群組，以及在 CloudWatch Logs 主控台上存取 Apache Airflow 日誌。

存取 的日誌 **requirements.txt**

1. 在 Amazon MWAA 主控台上開啟環境頁面。
2. 選擇環境。
3. 在監控窗格中選擇 Airflow 排程器日誌群組。
4. 在requirements_install_ip日誌串流中選擇日誌。
5. 請參閱 環境上安裝的套件清單/usr/local/airflow/.local/bin。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 檢閱套件清單，以及是否有任何套件在安裝期間發生錯誤。如果發生錯誤，您可能會收到類似以下的錯誤：

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

後續步驟？

- 了解如何使用 Amazon CloudWatch 警示在 中設定 CloudWatch 警示。 Amazon CloudWatch
- 了解如何使用 CloudWatch 儀表板在 中建立 CloudWatch 儀表板。

在 Amazon MWAA 上監控儀表板和警示

您可以在 Amazon CloudWatch 中建立自訂儀表板，並為特定指標新增警示，以監控 Amazon Managed Workflows for Apache Airflow 環境的運作狀態。當警示在儀表板上時，它會在處於 ALARM 狀態時變成紅色，讓您更輕鬆地主動監控 Amazon MWAA 環境的運作狀態。

Apache Airflow 公開多個程序的指標，包括 DAG 程序數量、DAG 包大小、目前正在執行的任務、任務失敗和成功。當您建立環境時，Airflow 會自動將 Amazon MWAA 環境的指標傳送至 CloudWatch。此頁面說明如何為 Amazon MWAA 環境的 CloudWatch 中的 Airflow 指標建立運作狀態儀表板。

內容

- 指標
- 警示狀態概觀
- 自訂儀表板和警示範例
  - 關於這些指標
  - 關於儀表板
  - 使用教學 AWS 課程
  - 使用 CloudFormation
- 刪除指標和儀表板
- 後續步驟？

指標

您可以為 Apache Airflow 版本可用的任何指標建立自訂儀表板和警示。每個指標對應至 Apache Airflow 金鑰效能指標 (KPI)。若要存取指標清單，請參閱：

- CloudWatch 中的 Apache Airflow 環境指標

警示狀態概觀

警示擁有以下可能的狀態：

- OK – 指標或表達式在定義的閾值內。
- ALARM – 指標或表達式在定義的閾值外。
- INSUFFICIENT_DATA – 警示剛開始無法使用指標，或資料不足無法讓指標判斷警示狀態。

自訂儀表板和警示範例

您可以建置自訂監控儀表板，以顯示 Amazon MWAA 環境所選指標的圖表。

關於這些指標

以下清單說明本節中教學課程和範本定義在自訂儀表板中建立的每個指標。

- QueuedTasks - 佇列狀態的任務數量。對應至 `executor.queued_tasks` Apache Airflow 指標。
- TasksPending - 執行器中待定的任務數量。對應至 `scheduler.tasks.pending` Apache Airflow 指標。

> **Note**
>
> 不適用於 Apache Airflow v2.2 和更新版本。

- RunningTasks - 在執行器中執行的任務數量。對應至 `executor.running_tasks` Apache Airflow 指標。
- SchedulerHeartbeat - Apache Airflow 在排程器任務上執行的簽入次數。對應至 `scheduler_heartbeat` Apache Airflow 指標。
- TotalParseTime - 掃描和匯入所有 DAG 檔案一次所需的秒數。對應至 `dag_processing.total_parse_time` Apache Airflow 指標。

關於儀表板

下圖顯示本節教學課程和範本定義所建立的監控儀表板。

使用教學 AWS 課程

您可以使用下列 AWS 教學課程，自動為目前部署的任何 Amazon MWAA 環境建立運作狀態儀表板。它也會在所有 Amazon MWAA 環境中為運作狀態不佳的工作者和排程器活動訊號失敗建立 CloudWatch 警示。

- Amazon MWAA 的 CloudWatch Dashboard Automation

使用 CloudFormation

您可以使用本節中的 CloudFormation 範本定義，在 CloudWatch 中建立監控儀表板，然後在 CloudWatch 主控台上新增警示，以在指標超過特定閾值時接收通知。若要使用此範本定義建立堆疊，請參閱在 CloudFormation 主控台上建立堆疊。若要將警示新增至儀表板，請參閱使用警示。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Creates MWAA Cloudwatch Dashboard
Parameters:
  DashboardName:
    Description: Enter the name of the CloudWatch Dashboard
    Type: String
  EnvironmentName:
    Description: Enter the name of the MWAA Environment
    Type: String
Resources:
  BasicDashboard:
    Type: AWS::CloudWatch::Dashboard
    Properties:
      DashboardName: !Ref DashboardName
      DashboardBody:
        Fn::Sub: '{
              "widgets": [
                  {
                      "type": "metric",
                      "x": 0,
                      "y": 0,
                      "width": 12,
                      "height": 6,
                      "properties": {
                          "view": "timeSeries",
                          "stacked": true,
                          "metrics": [
                              [
```

```
                        "AmazonMWAA",
                        "QueuedTasks",
                        "Function",
                        "Executor",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "QueuedTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 0,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "RunningTasks",
                        "Function",
                        "Executor",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "RunningTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 12,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
```

```
                        "view": "timeSeries",
                        "stacked": true,
                        "metrics": [
                            [
                                "AmazonMWAA",
                                "SchedulerHeartbeat",
                                "Function",
                                "Scheduler",
                                "Environment",
                                "${EnvironmentName}"
                            ]
                        ],
                        "region": "${AWS::Region}",
                        "title": "SchedulerHeartbeat ${EnvironmentName}",
                        "period": 300
                    }
                },
                {
                    "type": "metric",
                    "x": 12,
                    "y": 0,
                    "width": 12,
                    "height": 6,
                    "properties": {
                        "view": "timeSeries",
                        "stacked": true,
                        "metrics": [
                            [
                                "AmazonMWAA",
                                "TasksPending",
                                "Function",
                                "Scheduler",
                                "Environment",
                                "${EnvironmentName}"
                            ]
                        ],
                        "region": "${AWS::Region}",
                        "title": "TasksPending ${EnvironmentName}",
                        "period": 300
                    }
                },
                {
                    "type": "metric",
                    "x": 0,
```

```
                    "y": 12,
                    "width": 24,
                    "height": 6,
                    "properties": {
                        "view": "timeSeries",
                        "stacked": true,
                        "region": "${AWS::Region}",
                        "metrics": [
                            [
                                "AmazonMWAA",
                                "TotalParseTime",
                                "Function",
                                "DAG Processing",
                                "Environment",
                                "${EnvironmentName}"
                            ]
                        ],
                        "title": "TotalParseTime  ${EnvironmentName}",
                        "period": 300
                    }
                }
            ]
        }'
```

刪除指標和儀表板

如果您刪除 Amazon MWAA 環境，也會刪除對應的儀表板。CloudWatch 指標會儲存十五 (15) 個月，且無法刪除。CloudWatch 主控台會將指標搜尋限制在上次擷取指標後兩 (2) 週，以確保為您的 Amazon MWAA 環境顯示最新的執行個體。若要進一步了解，請參閱 Amazon CloudWatch FAQs。

後續步驟？

- 了解如何建立 DAG，以查詢您環境的 Amazon Aurora PostgreSQL 中繼資料資料庫，並將自訂指標發佈至 中的 CloudWatch使用 DAG 在 CloudWatch 中寫入自訂指標。

CloudWatch 中的 Apache Airflow 環境指標

Apache Airflow v2 和 v3 已設定為收集 StatsD 指標，並將 Amazon Managed Workflows for Apache Airflow 環境的 StatsD 指標傳送至 Amazon CloudWatch。Apache Airflow 傳送的指標完整清單可在 Apache Airflow 參考指南中的指標頁面上取得。此頁面說明 CloudWatch 中可用的 Apache Airflow 指標，以及如何在 CloudWatch 主控台中存取指標。

內容

- 條款
- 維度
- 在 CloudWatch 主控台中存取指標
- CloudWatch 中可用的 Apache Airflow 指標
  - Apache Airflow 計數器
  - Apache Airflow 量測器
  - Apache Airflow 計時器
- 選擇報告哪些指標
- 後續步驟？

條款

命名空間

命名空間是 AWS 服務的 CloudWatch 指標的容器。對於 Amazon MWAA，命名空間為 AmazonMWAA。

CloudWatch 指標

CloudWatch 指標代表 CloudWatch 特有的一組按時間順序排列的資料點。

Apache Airflow 指標

Apache Airflow 特定的指標。

維度

維度是一組名稱值對，是指標身分的一部分。

單位

統計資料具有度量單位。對於 Amazon MWAA，單位包括計數、秒和毫秒。對於 Amazon MWAA，單位是根據原始 Airflow 指標中的單位設定。

維度

本節說明 CloudWatch 中 Apache Airflow 指標的 CloudWatch Dimensions 分組。

維度	Description
DAG	指出特定的 Apache Airflow DAG 名稱。
DAG 檔案名稱	指出特定的 Apache Airflow DAG 檔案名稱。
函式	此維度用於改善 CloudWatch 中指標的分組。
任務	指出排程器執行的 Apache Airflow 任務。值一律為 Job。
運算子	指出特定的 Apache Airflow 運算子。
集區	指出特定的 Apache Airflow 工作者集區。
任務	指出特定的 Apache Airflow 任務。
HostName	指出特定執行中 Apache Airflow 程序的主機名稱。

在 CloudWatch 主控台中存取指標

本節說明如何存取 CloudWatch 中特定 DAG 的效能指標。

存取維度的效能指標

1. 在 CloudWatch 主控台上開啟指標頁面。
2. 選取您的 AWS 區域。
3. 選擇 AmazonMWAA 命名空間。
4. 在所有指標索引標籤中，選取維度。例如 DAG、Environment。

5. 選擇維度的 CloudWatch 指標。例如，TaskInstanceSuccesses 或 TaskInstanceDuration。選擇繪製所有搜尋結果的圖形。
6. 選擇圖形化指標索引標籤，以存取 Apache Airflow 指標的效能統計資料，例如 DAG、環境、任務。

CloudWatch 中可用的 Apache Airflow 指標

本節說明傳送至 CloudWatch 的 Apache Airflow 指標和維度。

Apache Airflow 計數器

本節中的 Apache Airflow 指標包含有關 Apache Airflow 計數器的資料。

CloudWatch 指標	Apache Airflow 指標	單位	維度	
SLAMissed **Note** 僅適用於 Apache Airflow v2.4.3 至 v2.10.3。	sla_missed	計數	函數、排程器	
FailedSLACallback **Note** 僅適用於 Apache Airflow v2.4.3 至 v2.10.3。	sla_callback_notification_failure	計數	函數、排程器	
更新	dataset.updates	計數	函數、排程器	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
Note 適用於 Apache Airflow 2.6.3 版及更新版本。				
孤立 Note 適用於 Apache Airflow 2.6.3 版及更新版本。	dataset.orphaned	計數	函數、排程器	
FailedCeleryTaskExecution Note 適用於 Apache Airflow 2.4.3 版及更新版本。	celery.execute_command.failure	計數	函數、Celery	
FilePathQueueUpdateCount Note 適用於 Apache Airflow 2.6.3 版及更新版本。	dag_processing.file_path_queue_update_count	計數	函數、排程器	
CriticalSectionBusy	scheduler.critical_section_busy	計數	函數、排程器	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
DagBagSize	dagbag_size	計數	函數、DAG 處理	
DagCallbackExceptions	dag.callback_exceptions	計數	DAG，全部	
FailedSLAEmailAttempts **Note** 不適用於 Apache Airflow v3.0.6 和更新版本。	sla_email_notification_failure	計數	函數、排程器	
TaskInstanceFinished	ti.finish.{dag_id}.{task_id}.{state}	計數	DAG、{dag_id} 任務，{task_id} 狀態，{state}	
JobEnd	{job_name}_end	計數	任務，{job_name}	
JobHeartbeatFailure	{job_name}_heartbeat_failure	計數	任務，{job_name}	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
JobStart	{job_name}_start	計數	任務，{job_name}	
ManagerStalls	dag_processing.manager_stalls	計數	函數、DAG 處理	
OperatorFailures	Operator_failures_{operator_name}	計數	運算子，{operator_name}	
OperatorSuccesses	Operator_successes_{operator_name}	計數	運算子，{operator_name}	
OtherCallbackCount Note Apache Airflow 2.6.3 版及更新版本中提供。	dag_processing.other_callback_count	計數	函數、排程器	
Processes	dag_processing.processes	計數	函數、DAG 處理	
SchedulerHeartbeat	scheduler_heartbeat	計數	函數、排程器	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
StartedTaskInstances	ti.start.{dag_id}.{task_id}	計數	DAG，全部 任務，全部	
SlaCallbackCount	dag_processing.sla_callback_count Note 適用於 Apache Airflow 2.6.3 版及更新版本。	計數	函數、排程器	
TasksKilledExternally	scheduler.tasks.killed_externally	計數	函數、排程器	
TaskTimeoutError	celery.task_timeout_error	計數	函數、Celery	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TaskInstanceCreatedUsingOperator	task_instance_created-{operator_name}	計數	運算子，{operator_name}	
TaskInstancePreviouslySucceeded	previously_succeeded	計數	DAG，全部 任務，全部	
TaskInstanceFailures	ti_failures	計數	DAG，全部 任務，全部	
TaskInstanceSuccesses	ti_successes	計數	DAG，全部 任務，全部	
TaskRemovedFromDAG	task_removed_from_dag.{dag_id}	計數	DAG、{dag_id}	
TaskRestoredToDAG	task_restored_to_dag.{dag_id}	計數	DAG、{dag_id}	
TriggersSucceeded Note 適用於 Apache Airflow 2.7.2 版及更新版本。	triggers.succeeded	計數	函數、觸發	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TriggersFailed Note 適用於 Apache Airflow 2.7.2 版及更新版本。	triggers.failed	計數	函數、觸發	
TriggersBlockedMainThread Note 適用於 Apache Airflow 2.7.2 版及更新版本。	triggers.blocked_main_thread	計數	函數、觸發	
TriggerHeartbeat Note 適用於 Apache Airflow 2.8.1 版及更新版本。	triggerer_heartbeat	計數	函數、觸發器	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TaskInstanceCreatedUsingOperator	airflow.task_instance_created_{operator_name} Note 適用於 Apache Airflow 2.7.2 版及更新版本。	計數	運算子、{operator_name}	
ZombiesKilled	zombies_killed	計數	DAG，全部 任務，全部	

Apache Airflow 量測器

本節中的 Apache Airflow 指標包含有關 Apache Airflow 量測計的資料。

CloudWatch 指標	Apache Airflow 指標	單位	維度	
DAGFileRefreshError	dag_file_refresh_error	計數	函數、DAG 處理	

CloudWatch 指標	Apache Airflow 指標	單位	維度
ImportErrors	dag_processing.import_errors	計數	函數、DAG 處理
Exception Failures	smart_sensor_operator.exception_failures	計數	函數、智慧型感應器運算子
ExecutedTasks	smart_sensor_operator.executed_tasks	計數	函數、智慧型感應器運算子
InfraFailures	smart_sensor_operator.infra_failures	計數	函數、智慧型感應器運算子
LoadedTasks	smart_sensor_operator.loaded_tasks	計數	函數、智慧型感應器運算子
TotalParseTime	dag_processing.total_parse_time	秒鐘	函數、DAG 處理
TriggeredDagRuns Note Apache Airflow 2.6.3 版及更新版本中提供。	dataset.triggered_dagruns	計數	函數、排程器

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TriggersRunning Note Apache Airflow 2.7.2 版及更新版本中提供。	triggers.running.*{hostname}*	計數	函數、觸發 HostName、*{hostname}*	
PoolDeferredSlots Note Apache Airflow 2.7.2 版及更新版本中提供。	pool.deferred_slots。{pool_name}	計數	集區，{pool_name}	
DAGFileProcessingLastRunSecondsAgo	dag_processing.last_run.seconds_ago.{dag_filename}	秒鐘	DAG 檔案名稱，{dag_filename}	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
OpenSlots	executor.open_slots	計數	函數、執行器	
OrphanedTasksAdopted	scheduler.orphaned_tasks.adopted	計數	函數、排程器	
OrphanedTasksCleared	scheduler.orphaned_tasks.cleared	計數	函數、排程器	
PokedExceptions	smart_sensor_operator.poked_exception	計數	函數、智慧型感應器運算子	
PokedSuccess	smart_sensor_operator.poked_success	計數	函數、智慧型感應器運算子	
PokedTasks	smart_sensor_operator.poked_tasks	計數	函數、智慧型感應器運算子	
PoolFailures	pool.open_slots.{pool_name}	計數	集區，{pool_name}	
PoolStarvingTasks	pool.starving_tasks.{pool_name}	計數	集區，{pool_name}	
PoolOpenSlots	pool.open_slots.{pool_name}	計數	集區，{pool_name}	
PoolQueuedSlots	pool.queued_slots.{pool_name}	計數	集區，{pool_name}	

CloudWatch 指標	Apache Airflow 指標	單位	維度
PoolRunningSlots	pool.running_slots.{pool_name}	計數	集區，{pool_name}
ProcessorTimeouts	dag_processing.processor_timeouts	計數	函數、DAG 處理
QueuedTasks	executor.queued_tasks	計數	函數、執行器
RunningTasks	executor.running_tasks	計數	函數、執行器
TasksExecutable	scheduler.tasks.executable	計數	函數、排程器
TasksPending Note 不適用於 Apache Airflow v2.2 和更新版本。	scheduler.tasks.pending	計數	函數、排程器
TasksRunning	scheduler.tasks.running	計數	函數、排程器
TasksStarving	scheduler.tasks.starving	計數	函數、排程器

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TasksWithoutDagRun	scheduler.tasks.without_dagrun	計數	函數、排程器	
DAGFileProcessingLastNumOfDbQueries Note 適用於 Apache Airflow 2.10.1 版及更新版本。	dag_processing.last_num_of_db_queries.{dag_filename}	計數	DAG 檔案名稱，{dag_filename}	
PoolScheduledSlots Note 適用於 Apache Airflow 2.10.1 版及更新版本。	pool.scheduled_slots.{pool_name}	計數	集區，{pool_name}	

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TaskCpuUsage Note Apache Airflow 2.10.1 版及更新版本中提供。	cpu.usage.{dag_id}.{task_id}	百分比	DAG、{dag_id} 任務，{task_id}	
TaskMemoryUsage Note Apache Airflow 2.10.1 版及更新版本中提供。	mem.usage.{dag_id}.{task_id}	百分比	DAG、{dag_id} 任務，{task_id}	

Apache Airflow 計時器

本節中的 Apache Airflow 指標包含有關 Apache Airflow 計時器的資料。

CloudWatch 指標	Apache Airflow 指標	單位	維度	
CollectDBDags	collect_db_dags	毫秒	函數、DAG 處理	
CriticalSectionDuration	scheduler.critical_section_duration	毫秒	函數、排程器	
CriticalSectionQueryDuration Note 適用於 Apache Airflow 2.5.1 版及更新版本。	scheduler.critical_section_query_duration	毫秒	函數、排程器	
DAGDependencyCheck	dagrun.dependency-check.{dag_id}	毫秒	DAG、{dag_id}	
DAGDurationFailed	dagrun.duration.failed.{dag_id}	毫秒	DAG、{dag_id}	
DAGDurationSuccess	dagrun.duration.success.{dag_id}	毫秒	DAG、{dag_id}	

CloudWatch 指標	Apache Airflow 指標	單位	維度
DAGFilePr ocessingL astDuration	dag_proce ssing.las t_duration.{dag_fi lename}	秒鐘	DAG 檔案名稱，{dag_filename}
DAGSchedu leDelay	dagrun.sc hedule_delay. {dag_id}	毫秒	DAG、{dag_id}
FirstTask SchedulingDelay	dagrun.{d ag_id}.first_task_ scheduling_delay	毫秒	DAG、{dag_id}
Scheduler LoopDuration Note 適用於 Apache Airflow 2.5.1 版及更新版本。	scheduler .schedule r_loop_duration	毫秒	函數、排程器
TaskInsta nceDuration	dag.{dag_id}. {task_id}.duration	毫秒	DAG、{dag_id} 任務，{task_id}

CloudWatch 指標	Apache Airflow 指標	單位	維度	
TaskInstanceQueuedDuration	dag.{dag_id}.{t queued_duration **Note** 適用於 Apache Airflow 2.7.2 版及更新版本。	毫秒	DAG、{dag_id} 任務，{task_id}	
TaskInstanceScheduledDuration **Note** 適用於 Apache Airflow 2.7.2 版及更新版本。	dag.{dag_id}{ta .scheduled_duration	毫秒	DAG、{dag_id} 任務，{task_id}	

選擇報告哪些指標

您可以使用下列 Amazon MWAA 組態選項，選擇向 CloudWatch 發出或由 Apache Airflow 封鎖的 Apache Airflow 指標：

- **metrics.metrics_allow_list** — 逗號分隔的字首清單，可用來選取環境向 CloudWatch 發出的指標。如果您希望 Apache Airflow 不傳送所有可用的指標，而是選取元素子集，請使用此選項。例如 `scheduler,executor,dagrun`。
- **metrics.metrics_block_list** — 逗號分隔字首清單，用於篩選以清單元素開頭的指標。例如 `scheduler,executor,dagrun`。

如果您同時設定 `metrics.metrics_allow_list`和 `metrics.metrics_block_list`，Apache Airflow 會忽略 `metrics.metrics_block_list`。如果您設定 `metrics.metrics_block_list` 但未設定 `metrics.metrics_allow_list`，Apache Airflow 會篩選掉您在 中指定的元素`metrics.metrics_block_list`。

> **Note**
>
> `metrics.metrics_allow_list` 和 `metrics.metrics_block_list`組態選項僅適用於 Apache Airflow v2.6.3 和更新版本。對於舊版 Apache Airflow，請`metrics.statsd_block_list`改用 `metrics.statsd_allow_list`和 。

後續步驟？

- 探索用於在 PublishMetrics 發佈環境運作狀態指標的 Amazon MWAA API 操作。

Amazon MWAA 的容器、佇列和資料庫指標

除了 Apache Airflow 指標之外，您還可以使用 CloudWatch 監控 Amazon Managed Workflows for Apache Airflow 環境的基礎元件，其會收集原始資料並將資料處理為可讀且幾近即時的指標。透過這些環境指標，您將更清楚了解關鍵效能指標，協助您適當地調整環境大小，並偵錯工作流程的問題。這些指標適用於 Amazon MWAA 上所有支援的 Apache Airflow 版本。

Amazon MWAA 將為每個 Amazon Elastic Container Service (Amazon ECS) 容器和 Amazon Aurora PostgreSQL 執行個體提供 CPU 和記憶體使用率，以及為最舊訊息數量和存留期提供 Amazon Simple Queue Service (Amazon SQS) 指標、為資料庫連線提供 Amazon Relational Database Service (Amazon RDS) 指標、磁碟佇列深度、寫入操作、延遲和輸送量，以及 Amazon RDS Proxy 指標。這些指標也包含基礎工作者、其他工作者、排程器和 Web 伺服器的數量。

這些統計資料會保留 15 個月，因此您可以存取歷史資訊，並更清楚排程失敗的原因，以及對基礎問題進行疑難排解。您也可以設定監控特定閾值的警示，並在達到這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

主題

- 條款
- 維度
- 在 CloudWatch 主控台中存取指標
- 指標清單

條款

命名空間

命名空間是 AWS 服務的 CloudWatch 指標的容器。對於 Amazon MWAA，命名空間為 `AWS/MWAA`。

CloudWatch 指標

CloudWatch 指標代表 CloudWatch 特有的一組按時間順序排列的資料點。

維度

維度是一組名稱值對，是指標身分的一部分。

單位

統計資料具有度量單位。對於 Amazon MWAA，單位包含計數。

維度

本節說明 CloudWatch 中 Amazon MWAA 指標的 CloudWatch 維度分組。

維度	Description
叢集	Amazon MWAA 環境用來執行 Apache Airflow 元件的最少三個 Amazon ECS 容器的指標：排程器、工作者和 Web 伺服器。

維度	Description
佇列	將排程器與工作者分離的 Amazon SQS 佇列指標。當工作者讀取訊息時，它們會被視為進行中，不適用於其他工作者。如果在 12 小時可見性逾時之前未刪除訊息，其他工作者即可讀取訊息。
資料庫	Amazon MWAA 使用的 Aurora 叢集指標。這包括主要資料庫執行個體的指標，以及支援讀取操作的僅供讀取複本。Amazon MWAA 會同時發佈 READER 和 WRITER 執行個體的資料庫指標。

在 CloudWatch 主控台中存取指標

本節說明如何在 CloudWatch 中存取 Amazon MWAA 指標。

存取維度的效能指標

1. 在 CloudWatch 主控台上開啟指標頁面。
2. 選取您的 AWS 區域。
3. 選擇 AWS/MWAA 命名空間。
4. 在所有指標索引標籤中，選擇維度。例如，叢集。
5. 選擇維度的 CloudWatch 指標。例如，NumSchedulers 或 CPUUtilization。然後，選擇繪製所有搜尋結果的圖形。
6. 選擇圖形化指標索引標籤以存取效能指標。

指標清單

下表列出 Amazon MWAA 的叢集、佇列和資料庫服務指標。若要存取直接從 Amazon ECS、Amazon SQS 或 Amazon RDS 發出的指標描述，請選擇個別的文件連結。

主題

- 叢集指標
- 資料庫指標

- 佇列指標
- Application Load Balancer 指標

叢集指標

下列指標適用於每個排程器、基礎工作者、其他工作者和 Web 伺服器。如需每個叢集指標的詳細資訊和說明，請參閱《Amazon ECS 開發人員指南》中的可用指標和維度。

命名空間	指標	單位
AWS/MWAA	CPUUtilization	百分比
AWS/MWAA	MemoryUtilization	百分比

評估其他工作者和 Web 伺服器容器的數量

您可以使用叢集維度中提供的元件指標，如下列程序所述，來評估環境在特定時間點使用多少額外工作者或 Web 伺服器。您可以透過繪製 CPUUtilization 或 MemoryUtilization 指標的圖形，並將統計資料類型設定為範例計數來執行此操作。產生的值是AdditionalWorker元件RUNNING的任務總數。了解您環境使用的其他工作者執行個體數量，可協助您判斷環境如何擴展，並且您可以使用 來最佳化其他工作者的數量。

Workers

使用 評估其他工作者的數量 AWS 管理主控台

1. 選擇 AWS/MWAA 命名空間。
2. 在所有指標索引標籤中，選擇叢集維度。
3. 在叢集維度中，針對 AdditionalWorker，選擇 CPUUtilization 或 MemoryUtilization 指標。
4. 在圖形化指標索引標籤上，將期間設定為 1 分鐘，將統計資料設定為範例計數。

webservers

使用 評估其他 Web 伺服器的數量 AWS 管理主控台

1. 選擇 AWS/MWAA 命名空間。

2. 在所有指標索引標籤中，選擇叢集維度。
3. 在叢集維度中，針對 AdditionalWebservers，選擇 CPUUtilization 或 MemoryUtilization 指標。
4. 在圖形化指標索引標籤上，將期間設定為 1 分鐘，將統計資料設定為範例計數。

如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的服務RUNNING任務計數。

資料庫指標

下列指標適用於與 Amazon MWAA 環境相關聯的每個資料庫執行個體。

命名空間	指標	單位
AWS/MWAA	CPUUtilization	百分比
AWS/MWAA	DatabaseConnections	計數
AWS/MWAA	DiskQueueDepth	計數
AWS/MWAA	FreeableMemory	位元組
AWS/MWAA	VolumeWriteIOPS	每五分鐘計數
AWS/MWAA	WriteIOPS	每秒計數
AWS/MWAA	WriteLatency	秒鐘
AWS/MWAA	WriteThroughput	每秒位元組數

佇列指標

如需下列佇列指標之單位和說明的詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的 Amazon SQS 的可用 CloudWatch 指標。

命名空間	指標	單位
AWS/MWAA	ApproximateAgeOfOldestTask	秒鐘
AWS/MWAA	RunningTasks	計數
AWS/MWAA	QueuedTasks	計數

Application Load Balancer 指標

Application Load Balancer 指標適用於您環境中執行的 Web 伺服器。Amazon MWAA 使用這些指標到 ，根據流量擴展您的 Web 伺服器。如需下列負載平衡器指標的單位和說明詳細資訊，請參閱《Application Load Balancer 使用者指南》中的 Application Load Balancer 的 CloudWatch 指標。

命名空間	指標	單位
AWS/MWAA	ActiveConnectionCount	計數

Amazon Managed Workflows for Apache Airflow 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您（客戶）之間的共同責任。共同責任模式將其描述為雲端的安全性，和雲端中的安全性：

- 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 也提供您可以安全使用的服務。在AWS 合規計劃中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用於 Amazon MWAA 的合規計劃，請參閱AWS 合規計劃的 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 Amazon Managed Workflows for Apache Airflow 時套用共同責任模型。使用它來設定 Amazon MWAA 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon MWAA 資源。

在本節中：

- Amazon Managed Workflows for Apache Airflow 中的資料保護
- AWS Identity and Access Management
- Amazon Managed Workflows for Apache Airflow 的合規驗證
- Amazon Managed Workflows for Apache Airflow 中的彈性
- Amazon MWAA 中的基礎設施安全性
- Amazon MWAA 中的組態和漏洞分析
- Amazon MWAA 的安全最佳實務

Amazon Managed Workflows for Apache Airflow 中的資料保護

AWS 共同責任模型適用於 Amazon Managed Workflows for Apache Airflow 中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責控制在此基礎設施上託管的內容。此內容包含您使用之 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱安全部落格上的AWS 共同責任模型和 GDPR AWS 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別使用者帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。建議使用 TLS 1.2 或更新版本。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 服務中的所有 AWS 預設安全控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的欄位中，例如名稱欄位。這包括當您使用 Amazon MWAA 或使用主控台 AWS CLI、API 或 AWS SDKs的其他 AWS 服務時。您在標籤或用於名稱的任意格式欄位中輸入的任何資料都可以用於帳單或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

Amazon MWAA 上的加密

下列主題說明 Amazon MWAA 如何保護靜態和傳輸中的資料。使用此資訊來了解 Amazon MWAA 如何與 整合 AWS KMS 來加密靜態資料，以及如何使用傳輸中的 Transport Layer Security (TLS) 通訊協定加密資料。

主題

- 靜態加密
- 傳輸中加密

靜態加密

在 Amazon MWAA 上，靜態資料是服務儲存至持久性媒體的資料。

您可以使用 AWS擁有的金鑰進行靜態資料加密，或者選擇性地在建立環境時提供客戶受管金鑰以進行其他加密。如果您選擇使用客戶管理的 KMS 金鑰，它必須與您搭配環境使用的其他 AWS 資源和服務位於相同的帳戶中。

若要使用客戶受管 KMS 金鑰，您必須將 CloudWatch 存取所需的政策陳述式連接至您的金鑰政策。當您為環境使用客戶管理的 KMS 金鑰時，Amazon MWAA 會代表您連接四個授權。如需授予 Amazon MWAA 連接至客戶受管 KMS 金鑰的詳細資訊，請參閱客戶受管金鑰以進行資料加密。

如果您未指定客戶管理的 KMS 金鑰，根據預設，Amazon MWAA 會使用 AWS 擁有的 KMS 金鑰來加密和解密您的資料。建議使用 AWS 擁有的 KMS 金鑰來管理 Amazon MWAA 上的資料加密。

> **Note**
>
> 您需為 Amazon MWAA 上 AWS 擁有或客戶管理之 KMS 金鑰的儲存和使用付費。如需詳細資訊，請參閱 AWS KMS 定價。

加密成品

您可以在建立 Amazon MWAA 環境時指定 AWS擁有的金鑰或客戶受管金鑰，以指定用於靜態加密的加密成品。Amazon MWAA 會將所需的授予新增至您指定的金鑰。

Amazon S3 – Amazon S3 資料會在物件層級使用伺服器端加密 (SSE) 進行加密。Amazon S3 加密和解密會在存放 DAG 程式碼和支援檔案的 Amazon S3 儲存貯體上進行。物件上傳到 Amazon S3 時會加密，並在下載到您的 Amazon MWAA 環境時解密。根據預設，如果您使用客戶管理的 KMS 金鑰，Amazon MWAA 會使用它來讀取和解密 Amazon S3 儲存貯體上的資料。

CloudWatch Logs – 如果您使用的是 AWS 擁有的 KMS 金鑰，傳送至 CloudWatch Logs 的 Apache Airflow 日誌會使用 SSE 搭配 CloudWatch Logs AWS 擁有的 KMS 金鑰加密。如果您使用客戶管理的 KMS 金鑰，則必須將金鑰政策新增至 KMS 金鑰，以允許 CloudWatch Logs 使用您的金鑰。

Amazon SQS – Amazon MWAA 會為您的環境建立一個 Amazon SQS 佇列。Amazon MWAA 會使用 SSE 搭配 AWS 擁有的 KMS 金鑰或您指定的客戶受管 KMS 金鑰來加密傳入和傳出佇列的資料。無論您使用的是 AWS 擁有還是客戶管理的 KMS 金鑰，都必須將 Amazon SQS 許可新增至執行角色。

Aurora PostgreSQL – Amazon MWAA 會為您的環境建立一個 PostgreSQL 叢集。Aurora PostgreSQL 會使用 SSE，使用 AWS 擁有或客戶管理的 KMS 金鑰來加密內容。如果您使用客戶管理的 KMS 金鑰，Amazon RDS 會為金鑰新增至少兩個授權：一個用於叢集，另一個用於資料庫執行個體。如果您選擇在多個環境中使用客戶管理的 KMS 金鑰，Amazon RDS 可以建立額外的授權。如需詳細資訊，請參閱 Amazon RDS 中的資料保護。

傳輸中加密

傳輸中的資料稱為資料，可在其通過網路時攔截。

Transport Layer Security (TLS) 會在您環境的 Apache Airflow 元件和其他與 Amazon MWAA 整合 AWS 的服務之間加密傳輸中的 Amazon MWAA 物件，例如 Amazon S3。如需 Amazon S3 加密的詳細資訊，請參閱使用加密保護資料。

使用客戶受管金鑰進行加密

您可以選擇性地為環境中的資料加密提供客戶受管金鑰。您必須在與 Amazon MWAA 環境執行個體和 Amazon S3 儲存貯體相同的區域中建立客戶受管 KMS 金鑰，以存放工作流程的資源。如果您指定的客戶受管 KMS 金鑰與您用於設定環境的金鑰位於不同的帳戶中，您必須使用其 ARN 指定金鑰以進行跨帳戶存取。如需建立金鑰的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的建立金鑰。

支援的內容

AWS KMS 功能	支援	
AWS KMS 金鑰 ID 或 ARN。	是	
AWS KMS 金鑰別名。	否	
AWS KMS 多區域金鑰。	否	

使用授予進行加密

本主題說明授予 Amazon MWAA 代表您連接至客戶受管 KMS 金鑰，以加密和解密您的資料。

運作方式

AWS KMS 客戶受管 KMS 金鑰支援兩種資源型存取控制機制：金鑰政策和授權。

當許可大部分為靜態且用於同步服務模式時，會使用金鑰政策。當需要更多動態和精細的許可時，例如當服務需要為自己或其他帳戶定義不同的存取許可時，就會使用授予。

Amazon MWAA 會使用四個授予政策並將其連接至您的客戶受管 KMS 金鑰。這是因為環境從 CloudWatch Logs、Amazon SQS 佇列、Aurora PostgreSQL 資料庫、Secrets Manager 秘密、Amazon S3 儲存貯體和 DynamoDB 資料表加密靜態資料所需的精細許可。

當您建立 Amazon MWAA 環境並指定客戶受管 KMS 金鑰時，Amazon MWAA 會將授予政策連接至客戶受管 KMS 金鑰。這些政策允許 中的 Amazon MWAA `airflow.us-east-1.amazonaws.com`使用您的客戶受管 KMS 金鑰，代表您加密 Amazon MWAA 擁有的資源。

Amazon MWAA 會代表您建立指定的 KMS 金鑰並附加其他授權。這包括刪除環境時淘汰授予的政策、將客戶受管 KMS 金鑰用於用戶端加密 (CSE)，以及需要存取 Secrets Manager 中受客戶受管金鑰保護之秘密的 AWS Fargate 執行角色。

授予政策

Amazon MWAA 會代表您將下列以資源為基礎的政策授予新增至客戶管理的 KMS 金鑰。這些政策允許承授者和委託人 (Amazon MWAA) 執行政策中定義的動作。

授予 1：用於建立資料平面資源

```
{
  "Name": "mwaa-grant-for-env-mgmt-role-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

授予 2：用於**ControllerLambdaExecutionRole**存取

```
{
  "Name": "mwaa-grant-for-lambda-exec-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

授予 3：用於**CfnManagementLambdaExecutionRole**存取

```
{
  "Name": " mwaa-grant-for-cfn-mgmt-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ]
}
```

授予 4：用於 Fargate 執行角色以存取後端秘密

```
{
  "Name": "mwaa-fargate-access-for-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

將金鑰政策連接至客戶受管金鑰

如果您選擇將自己的客戶受管 KMS 金鑰與 Amazon MWAA 搭配使用，則必須將下列政策連接至金鑰，以允許 Amazon MWAA 使用它來加密您的資料。

如果您用於 Amazon MWAA 環境的客戶受管 KMS 金鑰尚未設定為使用 CloudWatch，您必須更新金鑰政策以允許加密的 CloudWatch Logs。如需詳細資訊，請參閱使用 AWS Key Management Service 服務在 CloudWatch 中加密日誌資料。

下列範例代表 CloudWatch Logs 的金鑰政策。取代為區域提供的範例值。

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:*:*"
    }
  }
}
```

AWS Identity and Access Management

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 管理員可控制誰可以進行身分驗證（登入）和授權（具有許可），以使用 Amazon Managed Workflows for Apache Airflow 資源。IAM 是一種可免費使用 AWS 的服務。

本主題提供 Amazon MWAA 如何使用 AWS Identity and Access Management (IAM) 的基本概觀。若要了解如何管理 Amazon MWAA 的存取權，請參閱 管理對 Amazon MWAA 環境的存取。

目錄

- 目標對象
- 使用身分來驗證
- 使用政策管理存取權
- 允許使用者存取自己的許可
- 對 Amazon Managed Workflows for Apache Airflow 身分和存取進行故障診斷
- Amazon MWAA 如何與 IAM 搭配使用

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 對 Amazon Managed Workflows for Apache Airflow 身分和存取進行故障診斷)
- 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 MWAA 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 MWAA 身分型政策範例)

使用身分來驗證

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。

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。

存取控制清單 (ACL)

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。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他政策類型，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 決定是否在涉及多個政策類型時允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

允許使用者存取自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接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": "*"
        }
    ]
}
```

對 Amazon Managed Workflows for Apache Airflow 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 Amazon MWAA 和 IAM 時可能遇到的常見問題。

我無權在 Amazon MWAA 中執行動作

如果 AWS 管理主控台 告知您無權執行 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。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole`動作，您的政策必須更新，以允許您將角色傳遞給 Amazon MWAA。

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。

當名為 的 IAM `marymajor` 使用者嘗試使用主控台在 Amazon MWAA 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

```
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 以外的人員 AWS 帳戶 存取我的 Amazon MWAA 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 Amazon MWAA 是否支援這些功能，請參閱 Amazon MWAA 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。

Amazon MWAA 如何與 IAM 搭配使用

Amazon MWAA 使用 IAM 身分型政策將許可授予 Amazon MWAA 動作和資源。如需可用於控制 Amazon MWAA 資源存取的自訂 IAM 政策建議範例，請參閱 the section called “存取 Amazon MWAA 環境”。

若要取得 Amazon MWAA 和其他 AWS 服務如何與 IAM 搭配使用的高階存取權，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

Amazon MWAA 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。Amazon MWAA 支援特定動作、資源和條件金鑰。

下列步驟說明如何使用 IAM 主控台建立新的 JSON 政策。此政策提供對 Amazon MWAA 資源的唯讀存取權。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "airflow:ListEnvironments",
        "airflow:GetEnvironment",
        "airflow:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱《IAM 使用者指南》中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

若要了解您在 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。

動作

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。

政策陳述式必須包含 `Action` 或 `NotAction` 元素。`Action` 元素會列出政策允許的動作。`NotAction` 元素會列出不允許的動作。

為 Amazon MWAA 定義的動作會反映您可以使用 Amazon MWAA 執行的任務。Detective 中的政策動作具有以下前綴：`airflow:`。

您可以使用萬用字元 (*) 來指定多個動作。您可以授予以字詞結尾的所有動作的存取權，而不是單獨列出這些動作，例如 `environment`。

若要取得 Amazon MWAA 動作的清單，請參閱《IAM 使用者指南》中的 Amazon Managed Workflows for Apache Airflow 定義的動作。

Amazon MWAA 身分型政策範例

若要存取 Amazon MWAA 政策，請參閱 管理對 Amazon MWAA 環境的存取。

根據預設，IAM 使用者和角色沒有建立或修改 Amazon MWAA 資源的許可。他們也無法使用 AWS 管理主控台 AWS CLI或 AWS API 執行任務。

IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將此類政策附加至需要此類許可的 IAM 使用者或群組。

> **Important**
>
> 我們建議您使用 IAM 角色和臨時登入資料來提供 Amazon MWAA 資源的存取權。避免將許可政策直接連接到您的 IAM 使用者。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題

- 政策最佳實務
- 使用 Amazon MWAA 主控台
- 允許使用者存取自己的許可

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 Amazon MWAA 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 受管政策並轉向最低權限許可 – 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 Amazon MWAA 主控台

若要使用 Amazon MWAA 主控台，使用者或角色必須能夠存取與 API 中對應動作相符的相關動作。

若要存取 Amazon MWAA 政策，請參閱 管理對 Amazon MWAA 環境的存取。

允許使用者存取自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接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 Workflows for Apache Airflow 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS 合規計劃。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 AWS 安全文件。

Amazon Managed Workflows for Apache Airflow 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。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，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、容錯能力和擴展能力，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。

Amazon MWAA 中的基礎設施安全性

Amazon Managed Workflows for Apache Airflow 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Amazon MWAA。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

Amazon MWAA 中的組態和漏洞分析

組態和 IT 控制是 AWS 與身為我們客戶的您共同的責任。

Amazon Managed Workflows for Apache Airflow 會定期修補和升級您環境中的 Apache Airflow。確保您的 VPCs 使用適當的存取政策。

如需詳細資訊，請參閱下列資源：

- Amazon Managed Workflows for Apache Airflow 的合規驗證
- 共同的責任模型
- Amazon Web Services：安全程序概觀
- Amazon MWAA 中的基礎設施安全性
- Amazon MWAA 的安全最佳實務

Amazon MWAA 的安全最佳實務

Amazon MWAA 提供許多安全功能，供您在開發和實作自己的安全政策時考慮。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。

- 使用最低許可政策。僅將許可授予使用者執行任務所需的資源或動作。
- 使用 AWS CloudTrail 來監控您帳戶中的使用者活動。
- 確保 Amazon S3 儲存貯體政策和物件 ACLs 將許可授予相關聯 Amazon MWAA 環境的使用者，以將物件放入儲存貯體。這可確保具有將工作流程新增至儲存貯體之許可的使用者也具有在 Airflow 中執行工作流程的許可。
- 使用與 Amazon MWAA 環境相關聯的 Amazon S3 儲存貯體。Amazon S3 儲存貯體可以是任何名稱。請勿將其他物件存放在儲存貯體中，或使用儲存貯體搭配其他服務。

Apache Airflow 中的安全最佳實務

Apache Airflow 不是多租戶。雖然有存取控制措施可將某些功能限制為 Amazon MWAA 實作的特定使用者，但 DAG 建立者確實能夠撰寫 DAGs，以變更 Apache Airflow 使用者權限並與基礎中繼資料庫互動。

我們建議在 Amazon MWAA 上使用 Apache Airflow 時執行下列步驟，以確保您環境的中繼資料庫和 DAGs是安全的。

- 假設 Amazon MWAA 執行角色或 Apache Airflow 連線可存取的任何內容，也可以讓可寫入環境的使用者存取具有 DAG 寫入存取權的個別團隊，或能夠將檔案新增至 Amazon S3 `/dags` 資料夾。
- 請勿直接提供 Amazon S3 DAGs 資料夾存取。反之，請使用 CI/CD 工具將 DAGs 寫入 Amazon S3，並執行驗證步驟，確保 DAG 程式碼符合團隊的安全準則。
- 防止使用者存取您環境的 Amazon S3 儲存貯體。反之，請使用 DAG 工廠，以 YAML、JSON 或其他存放在與存放 DAGs Amazon MWAA Amazon S3 儲存貯體不同位置的定義檔案為基礎產生 DAGs。
- 在 Secrets Manager 中存放秘密。雖然這不會阻止可以寫入 DAGs的使用者讀取秘密，但會阻止他們修改您環境使用的秘密。

偵測 Apache Airflow 使用者權限的變更

您可以使用 CloudWatch Logs Insights 來偵測 DAGs 變更 Apache Airflow 使用者權限的發生情況。若要這樣做，您可以在其中一個 DAGs 變更 Apache Airflow 使用者權限時，使用 EventBridge 排程規則、Lambda 函數和 CloudWatch Logs Insights 來傳送通知至 CloudWatch 指標。

先決條件

若要完成下列步驟，您將需要下列項目：

- 在日誌層級啟用所有 Apache Airflow `INFO`日誌類型的 Amazon MWAA 環境。如需詳細資訊，請參閱 the section called “存取 Airflow 日誌”。

設定 Apache Airflow 使用者權限變更的通知

1. 建立 Lambda 函數，針對五個 Amazon MWAA `Scheduler`環境日誌群組 (`DAGProcessing`、、`Task`、 `WebServer`和) 執行下列 CloudWatch Logs Insights 查詢字串`Worker`。

   ```
   fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
    by @log
   ```

2. 建立依排程執行的 EventBridge 規則，並使用您在上一個步驟中建立的 Lambda 函數做為規則的目標。使用 Cron 或 Rate 表達式設定排程，以定期執行。

Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本

本主題說明 Amazon Managed Workflows for Apache Airflow 支援的 Apache Airflow 版本，以及升級至最新版本的最佳實務。

主題

- 關於 Amazon MWAA 版本
- 最新版本
- Apache Airflow 版本
- Apache Airflow 元件
- 升級 Apache Airflow 版本
- 降級 Apache Airflow 版本
- Apache Airflow 已棄用版本
- Apache Airflow 版本支援和常見問答集

關於 Amazon MWAA 版本

Amazon MWAA 會建置容器映像，將 Apache Airflow 版本與其他常見的二進位檔和 Python 程式庫綁定。映像針對您指定的版本使用 Apache Airflow 基本安裝。建立環境時，您可以指定要使用的映像版本。建立環境後，它會繼續使用指定的映像版本，直到您將其升級至更新版本為止。

最新版本

Amazon MWAA 支援多個 Apache Airflow 版本。如果您在建立環境時未指定映像版本，Amazon MWAA 會使用 Apache Airflow 的最新支援版本建立環境。

Apache Airflow 版本

Amazon Managed Workflows for Apache Airflow 支援下列 Apache Airflow 版本。

Note

- 自 2025 年 12 月 30 日起，Amazon MWAA 將終止對 Apache Airflow 版本 v2.4.3、v2.5.1 和 v2.6.3 的支援。如需詳細資訊，請參閱 Apache Airflow 版本支援和常見問答集。
- 從 Apache Airflow v2.2.2 開始，Amazon MWAA 支援直接在 Apache Airflow Web 伺服器上安裝 Python 需求、供應商套件和自訂外掛程式。
- 從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制，Amazon MWAA 會為您指定一個，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。

  如需在需求檔案中設定限制的詳細資訊，請參閱安裝 Python 相依性。

Apache Airflow 版本	Apache Airflow 發行日期	Amazon MWAA 可用性日期	Apache Airflow 限制條件	Python 版本
2.11.0 版	2025 年 5 月 20 日	2026 年 1 月 7 日	v2.11.0 限制條件檔案	Python 3.12
v3.0.6	2025 年 8 月 29 日	2025 年 10 月 1 日	v3.0.6 限制條件檔案	Python 3.12
v2.10.3	2024 年 11 月 4 日	2024 年 12 月 18 日	v2.10.3 限制條件檔案	Python 3.11
2.10.1 版	2024 年 9 月 5 日	2024 年 9 月 26 日	v2.10.1 限制條件檔案	Python 3.11
v2.9.2	2024 年 6 月 10 日	2024 年 7 月 9 日	v2.9.2 限制條件檔案	Python 3.11
2.8.1 版	2024 年 1 月 19 日	2024 年 2 月 23 日	v2.8.1 限制條件檔案	Python 3.11
2.7.2 版	2023 年 10 月 12 日	2023 年 11 月 6 日	v2.7.2 限制條件檔案	Python 3.11

如需遷移自我管理 Apache Airflow 部署或遷移現有 Amazon MWAA 環境的詳細資訊，包括備份中繼資料資料庫的說明，請參閱 Amazon MWAA 遷移指南。

Apache Airflow 元件

本節說明 Amazon MWAA 上每個 Apache Airflow 版本可用的 Apache Airflow 排程器和工作者數量，並提供金鑰 Apache Airflow 功能的清單，指出支援每個功能的版本。

排程器

Apache Airflow v2 和更新版本的排程器：

排程器（預設）	排程器（分鐘）	排程器（最大值）
2	2	5

工作程序

Apache Airflow v2 和更新版本的工作者：

工作者（預設）	工作者（分鐘）	工作者（上限）
10	1	25

升級 Apache Airflow 版本

Amazon MWAA 支援次要版本升級。這表示您可以將環境從 版本升級至 x.**1**.z x.**2**.z，但不能升級至新的主要版本，例如從 **1**.y.z升級至 **2**.y.z。

如需詳細資訊，以及更新工作流程資源，以及將環境升級至新版本的詳細說明，請參閱 the section called “變更版本”。

降級 Apache Airflow 版本

Amazon MWAA 支援次要版本降級至降級時仍然支援的較早版本。這表示您可以將環境從 版本降級x.**2**.z為 x.**1**.z，但不能降級為先前的主要版本，例如從 降級**2**.y.z為 **1**.y.z。

如需詳細資訊，以及更新工作流程資源，以及將環境升級至新版本的詳細說明，請參閱 the section called “變更版本”。

Apache Airflow 已棄用版本

下表列出 Amazon MWAA 中已棄用的 Apache Airflow 版本，以及每個版本的初始版本和end-of-support日期。如需遷移至較新版本的詳細資訊，請參閱 Amazon MWAA 遷移指南。

Apache Airflow 版本	Apache Airflow 發行日期	Amazon MWAA 可用性日期	Amazon MWAA end-of-support日期
v1.10.12	2020 年 8 月 25 日	2020 年 11 月 24 日	2024 年 2 月 21 日
2.0.2 版	2021 年 4 月 19 日	2021 年 5 月 25 日	2024 年 4 月 29 日
2.2.2 版	2021 年 11 月 15 日	2022 年 1 月 27 日	2024 年 6 月 27 日
v2.4.3	2022 年 11 月 14 日	2023 年 1 月 5 日	2025 年 12 月 30 日
v2.5.1	2023 年 1 月 20 日	2023 年 4 月 11 日	2025 年 12 月 30 日
v2.6.3	2023 年 7 月 10 日	2023 年 8 月 9 日	2025 年 12 月 30 日

Apache Airflow 版本支援和常見問答集

根據 Apache Airflow 社群發行程序和版本政策，Amazon MWAA 致力於在任何指定時間支援至少三個次要版本的 Apache Airflow。我們將在支援結束日期前至少 180 天，宣布特定 Apache Airflow 次要版本的支援結束日期。

常見問答集

問：Amazon MWAA 支援 Apache Airflow 版本多久？

答：Amazon MWAA 在第一次可用後支援 Apache Airflow 修補程式版本至少 12 個月。

問：在 Amazon MWAA 上的 Apache Airflow 版本支援結束時，是否會通知我？

答案：是。如果您帳戶中的任何 Amazon MWAA 環境執行接近終止支援的版本，Amazon MWAA 會在 Health 儀板表 終止支援日期的 之前傳送通知。

問：終止支援日期會發生什麼事？

答：在支援日期結束時，您無法再使用已棄用版本來建立新的 Amazon MWAA 環境。您可以繼續存取現有的 Amazon MWAA 環境，這些環境會執行相關聯、已棄用版本的 Apache Airflow，風險由您自行承擔。若要升級至 Amazon MWAA 上較新版本的 Apache Airflow，請參閱 Amazon MWAA 遷移指南。

Important

您有責任將 Amazon MWAA 版本保持在最新狀態。 會 AWS 要求所有客戶將其 Amazon MWAA 環境升級至最新版本，以受益於最新的安全性、隱私權和可用性防護措施。如果您在過了棄用日期的不支援版本或軟體上操作環境，稱為舊版，您會面臨更大的安全、隱私權和操作風險，包括停機時間事件。透過在舊版上操作 Amazon MWAA 環境，您確認您了解並故意承擔這些風險，而且您同意盡快完成升級至最新版本。在舊版上繼續操作您的環境受制於規範您使用 AWS 服務的協議。

舊版不會被視為正式提供，也 AWS 不再支援舊版。因此，如果 AWS 判斷舊版會對服務、AWS其附屬公司或任何其他第三方構成安全或責任風險，或有傷害風險， AWS 可能會隨時限制任何舊版的存取或使用。您繼續在舊版上執行工作負載的決定可能會導致您的內容無法使用、損毀或無法復原。在舊版上執行的環境受限於服務水準協議 (SLA) 例外狀況。

在舊版上執行的環境和相關軟體可能包含錯誤、錯誤、瑕疵和有害元件。因此，即使協議或服務條款中有任何相反的資訊， 仍會照原樣 AWS 提供舊版。

如需 AWS共同責任模型的詳細資訊，請參閱 AWS Well-Architected Framework 中的共同責任。

Amazon Managed Workflows for Apache Airflow 服務端點和配額

Amazon Managed Workflows for Apache Airflow 具有下列服務配額和端點。服務配額 (也稱為限制) 是 AWS 帳戶的服務資源或操作的最大數量。

內容

- 服務端點
- Service Quotas
- 增加配額

服務端點

若要存取 Amazon MWAA 的端點清單，請參閱 Amazon Managed Workflows for Apache Airflow 端點和配額。

Service Quotas

配額名稱	說明	預設配額	可調整
環境	每個區域每個帳戶的 Amazon MWAA 環境數目上限。	10	是
每個環境的工作者數	每個 Amazon MWAA 環境的工作者數量上限。	25	是
每個環境的 Web 伺服器	每個 Amazon MWAA 環境的 Web 伺服器數量上限。	5	是

增加配額

您可以提交配額增加請求，以請求增加可調整的配額。

Amazon MWAA 常見問答集

此頁面說明使用 Amazon Managed Workflows for Apache Airflow 時可能遇到的常見問題。

內容

- 支援的版本
  - Amazon MWAA 支援哪些 Apache Airflow v2？
  - 我可以使用哪個 Python 版本？
- 使用案例
  - 我可以將 Amazon MWAA 與 Amazon SageMaker Unified Studio 搭配使用嗎？
  - 何時可以使用 AWS Step Functions 與 Amazon MWAA？
- 環境規格
  - 每個環境可以使用多少任務儲存體？
  - 用於 Amazon MWAA 環境的預設作業系統是什麼？
  - 我可以為 Amazon MWAA 環境使用自訂映像嗎？
  - Amazon MWAA HIPAA 是否合規？
  - Amazon MWAA 是否支援 Spot 執行個體？
  - Amazon MWAA 是否支援自訂網域？
  - 我可以在我的環境中使用 SSH 嗎？
  - 為什麼 VPC 安全群組需要自我參考規則？
  - 我可以從 IAM 中的不同群組隱藏環境嗎？
  - 我可以在 Apache Airflow 工作者上存放臨時資料嗎？
  - 我可以指定超過 25 個 Apache Airflow 工作者嗎？
  - Amazon MWAA 是否支援共用 VPCs 或共用子網路？
  - 我可以建立或整合自訂 Amazon SQS 佇列，以管理 Apache Airflow 中的任務執行和工作流程協同運作嗎？
- 指標
  - 使用哪些指標來判斷是否擴展工作者？
  - 我可以在 CloudWatch 中建立自訂指標嗎？
- DAGs、運算子、連線和其他問題
  - 我可以使用 PythonVirtualenvOperator嗎？

- Amazon MWAA 需要多長時間才能辨識新的 DAG 檔案？
- Apache Airflow 為什麼不收取我的 DAG 檔案？
- 我可以requirements.txt從環境移除 plugins.zip或 嗎？
- 為什麼我的外掛程式不會顯示在 Apache Airflow v2.0.2 管理外掛程式選單中？
- 我可以使用 AWS 資料庫遷移服務 (DMS) 運算子嗎？
- 當我使用 AWS 登入資料存取 Airflow REST API 時，是否可以將限流限制提高到每秒超過 10 筆交易 (TPS)？
- Airflow 任務執行 API 伺服器在 Amazon MWAA 中的何處執行？

支援的版本

Amazon MWAA 支援哪些 Apache Airflow v2？

若要了解 Amazon MWAA 支援的內容，請參閱 Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本。

我可以使用哪個 Python 版本？

Amazon Managed Workflows for Apache Airflow 支援下列 Apache Airflow 版本。

> **Note**
>
> - 自 2025 年 12 月 30 日起，Amazon MWAA 將終止對 Apache Airflow 2.4.3 版、2.5.1 版和 2.6.3 版的支援。如需詳細資訊，請參閱 Apache Airflow 版本支援和常見問答集。
> - 從 Apache Airflow v2.2.2 開始，Amazon MWAA 支援直接在 Apache Airflow Web 伺服器上安裝 Python 需求、供應商套件和自訂外掛程式。
> - 從 Apache Airflow 2.7.2 版開始，您的需求檔案必須包含 `--constraint`陳述式。如果您未提供限制，Amazon MWAA 會為您指定一個，以確保您的需求中列出的套件與您正在使用的 Apache Airflow 版本相容。
>
>   如需在需求檔案中設定限制的詳細資訊，請參閱安裝 Python 相依性。

Apache Airflow 版本	Apache Airflow 發行日期	Amazon MWAA 可用性日期	Apache Airflow 限制條件	Python 版本
2.11.0 版	2025 年 5 月 20 日	2026 年 1 月 7 日	v2.11.0 限制條件檔案	Python 3.12
3.0.6 版	2025 年 8 月 29 日	2025 年 10 月 1 日	v3.0.6 限制條件檔案	Python 3.12
2.10.3 版	2024 年 11 月 4 日	2024 年 12 月 18 日	v2.10.3 限制條件檔案	Python 3.11
v2.10.1	2024 年 9 月 5 日	2024 年 9 月 26 日	v2.10.1 限制條件檔案	Python 3.11
2.9.2 版	2024 年 6 月 10 日	2024 年 7 月 9 日	v2.9.2 限制條件檔案	Python 3.11
2.8.1 版	2024 年 1 月 19 日	2024 年 2 月 23 日	v2.8.1 限制條件檔案	Python 3.11
2.7.2 版	2023 年 10 月 12 日	2023 年 11 月 6 日	v2.7.2 限制條件檔案	Python 3.11

如需遷移自我管理 Apache Airflow 部署或遷移現有 Amazon MWAA 環境的詳細資訊，包括備份中繼資料資料庫的說明，請參閱 Amazon MWAA 遷移指南。

使用案例

我可以將 Amazon MWAA 與 Amazon SageMaker Unified Studio 搭配使用嗎？

是。使用 Amazon SageMaker Unified Studio 工作流程，您可以在 Amazon SageMaker Unified Studio 中設定和執行一系列任務。Amazon SageMaker Unified Studio 工作流程使用 Apache Airflow 來建立資料處理程序的模型，並協調您的 Amazon SageMaker Unified Studio 程式碼成品。如需詳細資訊，請參閱工作流程一節。若要進一步了解 Amazon SageMaker，請參閱什麼是 Amazon SageMaker？

何時可以使用 AWS Step Functions 與 Amazon MWAA？

1. 您可以使用 Step Functions 來處理個別客戶訂單，因為 Step Functions 可以擴展以滿足一個訂單或一百萬個訂單的需求。
2. 如果您正在執行處理前一天訂單的隔夜工作流程，您可以使用 Step Functions 或 Amazon MWAA。Amazon MWAA 為您提供開放原始碼選項，以從您正在使用 AWS 的資源中抽象化工作流程。

環境規格

每個環境可以使用多少任務儲存體？

任務儲存體限制為 20 GB，並由 Amazon ECS Fargate 1.4 指定。RAM 的數量取決於您指定的環境類別。如需環境類別的詳細資訊，請參閱 設定 Amazon MWAA 環境類別。

用於 Amazon MWAA 環境的預設作業系統是什麼？

Amazon MWAA 環境是在執行 Amazon Linux 2 的 2.6 版及更舊版本的執行個體上建立，以及在執行 Amazon Linux 2023 的 2.7 版及更新版本的執行個體上建立。

我可以為 Amazon MWAA 環境使用自訂映像嗎？

不支援自訂映像。Amazon MWAA 使用建置在 Amazon Linux AMI 上的映像。Amazon MWAA `pip3 -r install`會針對您新增至環境 Amazon S3 儲存貯體的 requirements.txt 檔案中指定的需求執行 ，以安裝其他需求。

Amazon MWAA HIPAA 是否合規？

Amazon MWAA 符合健康保險流通與責任法案 (HIPAA) 的資格。如果您有 HIPAA 商業夥伴增補合約 (BAA) AWS，您可以使用 Amazon MWAA 在 2022 年 11 月 14 日當天或之後建立的環境中處理受保護醫療資訊 (PHI)。

Amazon MWAA 是否支援 Spot 執行個體？

Amazon MWAA 目前不支援 Apache Airflow 的隨需 Amazon EC2 Spot 執行個體類型。不過，Amazon MWAA 環境可以在 Amazon EMR 和 Amazon EC2 上觸發 Spot 執行個體。

Amazon MWAA 是否支援自訂網域？

若要為您的 Amazon MWAA 主機名稱使用自訂網域，請執行下列其中一項操作：

- 對於具有公有 Web 伺服器存取權的 Amazon MWAA 部署，您可以使用 Amazon CloudFront 搭配 Lambda@Edge 將流量導向您的環境，並將自訂網域名稱映射至 CloudFront。如需為公有環境設定自訂網域的詳細資訊和範例，請參閱 Amazon MWAA 範例 GitHub 儲存庫中公有 Web 伺服器範例的 Amazon MWAA 自訂網域。 GitHub
- 如需具有私有 Web 伺服器存取權的 Amazon MWAA 部署，請參閱 the section called “設定自訂網域”。

我可以在我的環境中使用 SSH 嗎？

雖然 Amazon MWAA 環境不支援 SSH，但可以使用 DAG 來執行使用 的 bash 命令`BashOperator`。例如：

```
from airflow import DAG
    from airflow.operators.bash_operator import BashOperator
    from airflow.utils.dates import days_ago
    with DAG(dag_id="any_bash_command_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
    task_id="bash_command",
    bash_command="{{ dag_run.conf['command'] }}"
    )
```

若要在 Apache Airflow UI 中觸發 DAG，請使用：

```
{ "command" : "your bash command"}
```

為什麼 VPC 安全群組需要自我參考規則？

透過建立自我參考規則，您將來源限制為 VPC 中的相同安全群組，而且並非開放給所有網路。若要進一步了解，請參閱 the section called “VPC 中的安全性”。

我可以從 IAM 中的不同群組隱藏環境嗎？

您可以在 中指定環境名稱來限制存取 AWS Identity and Access Management，但是，如果使用者可以存取一個環境，則無法在 AWS 主控台中使用存取篩選，他們可以存取所有環境。

我可以在 Apache Airflow 工作者上存放臨時資料嗎？

您的 Apache Airflow Operators 可以將臨時資料存放在工作者上。Apache Airflow 工作者可以在您環境的 Fargate 容器/tmp上存取 中的暫存檔案。

> **Note**
>
> 根據 Amazon ECS Fargate 1.4，總任務儲存體限制為 20 GB。無法保證後續任務會在相同的 Fargate 容器執行個體上執行，這可以使用不同的/tmp資料夾。

我可以指定超過 25 個 Apache Airflow 工作者嗎？

是。雖然您可以在 Amazon MWAA 主控台上指定最多 25 個 Apache Airflow 工作者，但您可以透過請求增加配額，在 環境中設定最多 50 個工作者。如需詳細資訊，請參閱請求提高配額。

Amazon MWAA 是否支援共用 VPCs 或共用子網路？

Amazon MWAA 不支援共用 VPCs 或共用子網路。您在建立環境時選取的 Amazon VPC 必須由嘗試建立環境的帳戶擁有。不過，您可以將流量從 Amazon MWAA 帳戶中的 Amazon VPC 路由到共用 VPC。如需將流量路由至共用 Amazon VPC 的詳細資訊和範例，請參閱《Amazon VPC Transit Gateways 指南》中的集中式傳出路由至網際網路。

我可以建立或整合自訂 Amazon SQS 佇列，以管理 Apache Airflow 中的任務執行和工作流程協同運作嗎？

否，您無法在 Amazon MWAA 中建立、修改或使用自訂 Amazon SQS 佇列。這是因為 Amazon MWAA 會自動為每個 Amazon MWAA 環境佈建和管理自己的 Amazon SQS 佇列。

指標

使用哪些指標來判斷是否擴展工作者？

Amazon MWAA 會監控 CloudWatch 中的 QueuedTasks 和 RunningTasks，以決定是否在您的環境中擴展 Apache Airflow 工作者。若要進一步了解，請參閱 監控和指標。

我可以在 CloudWatch 中建立自訂指標嗎？

不在 CloudWatch 主控台上。不過，您可以建立在 CloudWatch 中寫入自訂指標的 DAG。如需詳細資訊，請參閱 the section called “使用 DAG 撰寫自訂指標”。

DAGs、運算子、連線和其他問題

我可以使用 `PythonVirtualenvOperator`嗎？

Amazon MWAA `PythonVirtualenvOperator`上未明確支援 ，但您可以建立使用 的自訂外掛程式`PythonVirtualenvOperator`。如需範例程式碼，請參閱 the section called “自訂外掛程式以修補 PythonVirtualenvOperator ”。

Amazon MWAA 需要多長時間才能辨識新的 DAG 檔案？

DAGs 會定期從 Amazon S3 儲存貯體同步至您的環境。如果您新增新的 DAG 檔案，Amazon MWAA 大約需要 300 秒才能開始使用新的檔案。如果您更新現有的 DAG，Amazon MWAA 大約需要 30 秒才能辨識您的更新。

這些值、新 DAGs 的 300 秒，以及現有 DAGs 的更新 30 秒，`min_file_process_interval`分別對應至 Apache Airflow 組態選項 `dag_dir_list_interval`和 。

Apache Airflow 為什麼不收取我的 DAG 檔案？

以下是此問題的可能解決方案：

1. 檢查您的執行角色是否具有足夠的 Amazon S3 儲存貯體許可。若要進一步了解，請參閱 Amazon MWAA 執行角色。
2. 檢查 Amazon S3 儲存貯體是否已設定封鎖公開存取，並啟用版本控制。若要進一步了解，請參閱為 Amazon MWAA 建立 Amazon S3 儲存貯體。
3. 驗證 DAG 檔案本身。例如，請確定每個 DAG 都有唯一的 DAG ID。

我可以`requirements.txt`從環境移除 `plugins.zip`或 嗎？

目前，您無法在新增 plugins.zip 或 requirements.txt 後從環境中移除它們，但我們正在處理此問題。在此期間，解決方法是分別指向空白文字或 zip 檔案。若要進一步了解，請參閱 刪除 Amazon S3 上的檔案。

為什麼我的外掛程式不會顯示在 Apache Airflow v2.0.2 管理外掛程式選單中？

基於安全考量，Amazon MWAA 上的 Apache Airflow Webserver 的網路輸出有限，且不會直接在 2.0.2 版環境的 Apache Airflow Webserver 上安裝外掛程式和 Python 相依性。列出的外掛程式可讓 Amazon MWAA 在 AWS Identity and Access Management (IAM) 中驗證您的 Apache Airflow 使用者。

若要能夠直接在 Web 伺服器上安裝外掛程式和 Python 相依性，建議您使用 Apache Airflow v2.2 和更新版本建立新的環境。Amazon MWAA 會直接在適用於 Apache Airflow v2.2 和更新版本的 Web 伺服器上安裝 Python 相依性和自訂外掛程式。

我可以使用 AWS 資料庫遷移服務 (DMS) 運算子嗎？

Amazon MWAA 支援 DMS Operators。不過，此運算子無法用於對與 Amazon MWAA 環境相關聯的 Amazon Aurora PostgreSQL 中繼資料資料庫執行動作。

當我使用 AWS 登入資料存取 Airflow REST API 時，是否可以將限流限制提高到每秒超過 10 筆交易 (TPS)？

是，您可以。若要提高限流限制，請聯絡 AWS 客戶支援。

Airflow 任務執行 API 伺服器在 Amazon MWAA 中的何處執行？

Amazon MWAA 會在 Webserver 元件中執行 Airflow 任務執行 API 伺服器。任務執行 APIs僅適用於 Apache Airflow v3 和更新版本。如需 Amazon MWAA 架構的詳細資訊，請參閱 Architecture。

對 Amazon Managed Workflows for Apache Airflow 進行故障診斷

本章說明在 Amazon Managed Workflows for Apache Airflow 上使用 Apache Airflow 時可能遇到的常見問題和錯誤，以及解決這些錯誤的建議步驟。

內容

- 故障診斷：DAGs、運算子、連線和其他問題
  - 連線
    - 我無法連線至 Secrets Manager
    - 如何在執行角色政策中設定secretsmanager:ResourceTag/<tag-key>秘密管理員條件或資源限制？
    - 我無法連線至 Snowflake
    - 我無法在 Airflow UI 中找到我的連線
  - Web 伺服器
    - 我在存取 Web 伺服器時收到 5xx 錯誤
    - 我收到The scheduler does not seem to be running錯誤
  - 任務
    - 我讓任務停滯或未完成
    - 我在 Airflow v3 中收到沒有日誌的任務失敗
  - CLI
    - 在 CLI 中觸發 DAG 時收到「503」錯誤
    - 為什麼 dags backfill Apache Airflow CLI 命令會失敗？ 是否有解決方法？
  - 運算子
    - 我使用 S3Transform 運算子收到PermissionError: [Errno 13] Permission denied錯誤
- 故障診斷：建立和更新 Amazon MWAA 環境
  - 更新 requirements.txt
    - 我指定了新版本的 requirements.txt，需要超過 20 分鐘的時間來更新我的環境
  - 外掛程式
    - Amazon MWAA 是否支援實作自訂 UI？
  - 建立儲存貯體

    - 我無法選取 S3 封鎖公開存取設定的選項
- 建立 環境
    - 我嘗試建立環境，並且停滯在 Creating 狀態
    - 我嘗試建立環境，但狀態顯示為 Create failed
    - 我嘗試選取 VPC 並收到Network Failure錯誤
    - 我嘗試建立環境並收到服務、分割區或資源「必須傳遞」錯誤
    - 我嘗試建立環境，並顯示狀態為 ，Available但當我嘗試存取 Airflow UI 時，會顯示 Empty Reply from Server或 502 Bad Gateway錯誤
    - 我嘗試建立環境，而我的使用者名稱是一組隨機字元名稱
- 更新環境
    - 我嘗試變更環境類別，但更新失敗
- 存取環境
    - 我無法存取 Apache Airflow UI
- 故障診斷：CloudWatch Logs 和 CloudTrail 錯誤
    - 日誌
        - 我找不到任務日誌，或收到Reading remote log from Cloudwatch log_group錯誤
        - 任務在沒有任何日誌的情況下失敗
        - 我在 CloudTrail 中收到ResourceAlreadyExistsException錯誤
        - 我在 CloudTrail 中收到Invalid request錯誤
        - 我在 Apache Airflow 日誌Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory中取得
        - 我收到排程器日誌中的 psycopg2 'server 意外關閉連線'
        - 我在 DAG 處理日誌Executor reports task instance %s finished (%s) although the task says its %s中取得
        - 我在任務日誌Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.中取得

故障診斷：DAGs、運算子、連線和其他問題

本頁面上的主題說明您在 Amazon Managed Workflows for Apache Airflow 環境中可能遇到的 Apache Airflow v2 和 v3 Python 相依性、自訂外掛程式、DAGs、運算子、連線、任務和 Web 伺服器問題的解決方案。

內容

- 連線
  - 我無法連線至 Secrets Manager
  - 如何在執行角色政策中設定secretsmanager:ResourceTag/<tag-key>秘密管理員條件或資源限制？
  - 我無法連線至 Snowflake
  - 我無法在 Airflow UI 中找到我的連線
- Web 伺服器
  - 我在存取 Web 伺服器時收到 5xx 錯誤
  - 我收到The scheduler does not seem to be running錯誤
- 任務
  - 我讓任務停滯或未完成
  - 我在 Airflow v3 中收到沒有日誌的任務失敗
- CLI
  - 在 CLI 中觸發 DAG 時收到「503」錯誤
  - 為什麼 dags backfill Apache Airflow CLI 命令會失敗？ 是否有解決方法？
- 運算子
  - 我使用 S3Transform 運算子收到PermissionError: [Errno 13] Permission denied錯誤

連線

下列主題說明使用 Apache Airflow 連線或使用另一個 AWS 資料庫時可能收到的錯誤。

我無法連線至 Secrets Manager

建議下列步驟：

1. 了解如何在 中建立 Apache Airflow 連線和變數的私密金鑰the section called “設定 Secrets Manager”。
2. 了解如何在 中使用 Apache Airflow 變數 (`test-variable`) 的私密金鑰針對 AWS Secrets Manager Apache Airflow 變數在 中使用私密金鑰。
3. 了解如何在 中使用 Apache Airflow 連線 (`myconn`) 的私密金鑰在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線。

如何在執行角色政策中設定secretsmanager:ResourceTag/<tag-key>秘密管理員條件或資源限制？

> **Note**
>
> 適用於 Apache Airflow 2.0 版及更早版本。

目前，由於 Apache Airflow 中的已知問題，您無法在環境執行角色中使用條件金鑰或其他資源限制來限制對 Secrets Manager 秘密的存取。

我無法連線至 Snowflake

建議下列步驟：

1. 在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。
2. 將下列項目新增至您環境的 requirements.txt。

```
apache-airflow-providers-snowflake==1.3.0
```

3. 將下列匯入新增至您的 DAG：

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

確保 Apache Airflow 連線物件包含下列鍵值對：

1. Conn ID：nowflake_conn
2. Conn 類型：Snowflake
3. 主機：<my account>.<my region if not us-west-2>.snowflakecomputing.com
4. 結構描述：<my 結構描述>
5. 登入：<我的使用者名稱>
6. 密碼：*******
7. 連接埠：<port， if any>
8. 額外：

```
{
```

```
      "account": "<my account>",
      "warehouse": "<my warehouse>",
      "database": "<my database>",
      "region": "<my region if not using us-west-2 otherwise omit this line>"
      }
```

例如：

```
>>> import json
    >>> from airflow.models.connection import Connection
    >>> myconn = Connection(
    ...     conn_id='snowflake_conn',
    ...     conn_type='Snowflake',
    ...     host='123456789012.us-east-1.snowflakecomputing.com',
    ...     schema='YOUR_SCHEMA'
    ...     login='YOUR_USERNAME',
    ...     password='YOUR_PASSWORD',
    ...     port='YOUR_PORT'
    ...     extra=json.dumps(dict(account='123456789012', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='us-east-1')),
    ... )
```

我無法在 Airflow UI 中找到我的連線

Apache Airflow 在 Apache Airflow UI 中提供連線範本。無論連線類型為何，它都會使用此字串來產生連線 URI 字串。如果 Apache Airflow UI 中無法使用連線範本，則可以使用替代連線範本來產生連線 URI 字串，例如使用 HTTP 連線範本。

建議下列步驟：

1. 在 的 Apache Airflow UI 中存取 Amazon MWAA 提供的連線類型安裝在 Amazon MWAA 環境上的 Apache Airflow 提供者套件。
2. 存取命令，在 的 CLI 中建立 Apache Airflow 連線Apache Airflow CLI 命令參考。
3. 了解如何針對 Amazon MWAA 上的 Apache Airflow UI 中無法使用的連線類型，交替使用 Apache Airflow UI 中的連線範本連線類型概觀。

Web 伺服器

下列主題說明 Amazon MWAA 上 Apache Airflow Web 伺服器可能收到的錯誤。

我在存取 Web 伺服器時收到 5xx 錯誤

建議下列步驟：

1. 檢查 Apache Airflow 組態選項。確認您指定為 Apache Airflow 組態選項的鍵值對 AWS Secrets Manager已正確設定，例如 。若要進一步了解，請參閱 the section called “我無法連線至 Secrets Manager”。
2. 檢查 `requirements.txt`。確認 中列出的 Airflow "extras" `requirements.txt` 套件和其他程式庫與您的 Apache Airflow 版本相容。
3. 探索在`requirements.txt`檔案中指定 Python 相依性的方法，請參閱 在 requirements.txt 中管理 Python 相依性。

我收到`The scheduler does not seem to be running`錯誤

如果排程器似乎未執行，或幾小時前收到最後一個「心跳」，您的 DAGs 可能不會列在 Apache Airflow 中，也不會排定新任務。

建議下列步驟：

1. 確認您的 VPC 安全群組允許傳入存取連接埠 5432。需要此連接埠才能連線至您環境的 Amazon Aurora PostgreSQL 中繼資料資料庫。新增此規則後，請給 Amazon MWAA 幾分鐘的時間，錯誤可能會消失。若要進一步了解，請參閱 the section called “VPC 中的安全性”。

   > **Note**
   >
   > - Aurora PostgreSQL 中繼資料庫是 Amazon MWAA 服務架構的一部分，不適用於您的 AWS 帳戶。
   > - 資料庫相關錯誤通常是排程器失敗的徵狀，而不是根本原因。

2. 如果排程器未執行，可能是因為相依性安裝失敗或排程器過載等多種因素。透過存取 CloudWatch Logs 中的對應日誌群組，確認您的 DAGs、外掛程式和需求正常運作。若要進一步了解，請參閱監控和指標。

任務

下列主題說明 環境中 Apache Airflow 任務可能收到的錯誤。

我讓任務停滯或未完成

如果您的 Apache Airflow 任務「卡住」或未完成，建議您執行下列步驟：

1. 定義了大量 DAGs。減少 DAGs 數量並執行環境更新（例如變更日誌層級），以強制重設。

   a. 無論是否啟用，氣流都會剖析 DAGs。如果您使用的是環境容量的 50% 以上，您可能會開始讓 Apache Airflow 排程器負擔過重。這會導致 CloudWatch 指標中的大量總剖析時間，或 CloudWatch Logs 中的長 DAG 處理時間。還有其他方法可以最佳化本指南範圍之外的 Apache Airflow 組態。

   b. 若要進一步了解我們建議調整環境效能的最佳實務，請參閱 the section called “Apache Airflow 的效能調校”。

2. 佇列中可能會有大量任務。這通常顯示為大型且不斷增加None的 狀態任務數量，或在 Queued Tasks和/或 CloudWatch Tasks Pending 中顯示為大型任務數量。發生這種情況的原因如下：

   a. 如果執行的任務超過環境的容量，和/或在自動調整規模之前排入佇列的大量任務，則有時間偵測任務並部署額外的工作者。

   b. 如果執行的任務數量多於環境的容量，建議您減少 DAGs 同時執行的任務數量，和/或增加 Apache Airflow 工作者人數下限。

   c. 如果有許多任務在自動擴展有時間偵測和部署其他工作者之前排入佇列，我們建議交錯任務部署和/或增加 Apache Airflow 工作者人數下限。

   d. 您可以使用 AWS Command Line Interface (AWS CLI) 中的 update-environment 命令來變更在您環境中執行的工作者數量下限或上限。

   ```
   aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-
   workers 10
   ```

   e. 若要進一步了解我們建議調整環境效能的最佳實務，請參閱 the section called “Apache Airflow 的效能調校”。

3. 如果您的任務卡在「執行中」狀態，您也可以清除任務或將其標記為成功或失敗。這可讓環境的自動擴展元件縮減環境上執行的工作者數量。下圖描述了絞線任務的範例。

   - 選擇絞線任務的圓圈，然後選取清除（如圖所示）。這可讓 Amazon MWAA 縮減工作者規模；否則，Amazon MWAA 無法判斷要啟用或停用哪些 DAGs，如果仍有佇列的任務，則無法縮減規模。

4. 請參閱《Apache Airflow 參考指南》中的概念，進一步了解 Apache Airflow 任務生命週期。

我在 Airflow v3 中收到沒有日誌的任務失敗

如果您的 Apache Airflow 3 任務在沒有日誌的情況下失敗，請遵循下列步驟：

- 如果工作者日誌出現錯誤，例如任務失敗時`Task handler raised error: WorkerLostError('Worker exited prematurely: exitcode 15 Job: 12.')`，這表示指派給任務的分叉工作者程序可能意外終止。

  若要解決此問題，請考慮使用相同的最小值和最大值來設定 celery.worker_autoscale。例如：

  ```
  celery.worker_autoscale=5,5  # for mw1.small
  celery.worker_autoscale=10,10 # for mw1.medium
  celery.worker_autoscale=20,20 # for mw1.large
  ```

  這可確保工作者集區大小保持固定，防止工作者意外終止。

CLI

下列主題說明您在 中執行 Airflow CLI 命令時可能收到的錯誤 AWS Command Line Interface。

在 CLI 中觸發 DAG 時收到「503」錯誤

Airflow CLI 在 Apache Airflow Webserver 上執行，其並行有限。一般而言，最多可以同時執行 4 個 CLI 命令。

為什麼 **`dags backfill`** Apache Airflow CLI 命令會失敗？ 是否有解決方法？

> **Note**
>
> 以下僅適用於 Apache Airflow v2.0.2 環境。

與其他 Apache Airflow CLI 命令一樣， `backfill`命令會在本機剖析所有 DAGs，再處理任何 DAGs，無論 CLI 操作套用到哪個 DAG。在使用 Apache Airflow 2.0.2 版的 Amazon MWAA 環境中，因為在 CLI 命令執行時，外掛程式和需求尚未安裝在 Web 伺服器上、剖析操作失敗，而且未叫用 `backfill`操作。如果您在環境中沒有任何需求或外掛程式，`backfill`操作將會成功。

若要能夠執行 backfill CLI 命令，建議您在 bash 運算子中叫用它。在 bash 運算子中， 從工作者backfill啟動，允許 DAGs 成功剖析，因為所有必要的要求和 plguins 都可用並安裝。使用下列範例建立具有 的 DAGBashOperator，以執行 backfill。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="backfill_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="airflow dags backfill my_dag_id"
    )
```

運算子

下列主題說明使用 Operators 時可能收到的錯誤。

我使用 S3Transform 運算子收到**PermissionError: [Errno 13] Permission denied**錯誤

如果您嘗試使用 S3Transform 運算子執行 Shell 指令碼，且收到PermissionError: [Errno 13] Permission denied錯誤，建議您執行下列步驟。下列步驟假設您擁有現有的 plugins.zip 檔案。如果您要建立新的 plugins.zip，請參閱 安裝自訂外掛程式。

1. 在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。
2. 建立您的「轉換」指令碼。

   ```
   #!/bin/bash
   cp $1 $2
   ```

3. （選用） macOS 和 Linux 使用者可能需要執行下列命令，以確保指令碼可執行。

   ```
   chmod 777 transform_test.sh
   ```

4. 將指令碼新增至您的 plugins.zip。

   ```
   zip plugins.zip transform_test.sh
   ```

5. 請遵循上傳 plugins.zip 至 Amazon S3 中的步驟。
6. 請遵循 Amazon MWAA 主控台上指定 plugins.zip 版本中的步驟。
7. 建立下列 DAG。

```
from airflow import DAG
      from airflow.providers.amazon.aws.operators.s3_file_transform import
 S3FileTransformOperator
      from airflow.utils.dates import days_ago
      import os

      DAG_ID = os.path.basename(__file__).replace(".py", "")

      with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
      file_transform = S3FileTransformOperator(
      task_id='file_transform',
      transform_script='/usr/local/airflow/plugins/transform_test.sh',
      source_s3_key='s3://amzn-s3-demo-bucket/files/input.txt',
      dest_s3_key='s3://amzn-s3-demo-bucket/files/output.txt'
      )
```

8. 請遵循將 DAG 程式碼上傳至 Amazon S3 中的步驟。

故障診斷：建立和更新 Amazon MWAA 環境

本頁面上的主題包含您在建立和更新 Amazon Managed Workflows for Apache Airflow 環境時可能遇到的錯誤，以及如何解決這些錯誤。

內容

- 更新 requirements.txt
  - 我指定了新版本的 requirements.txt，需要超過 20 分鐘的時間來更新我的環境
- 外掛程式
  - Amazon MWAA 是否支援實作自訂 UI？
- 建立儲存貯體
  - 我無法選取 S3 封鎖公開存取設定的選項
- 建立 環境
  - 我嘗試建立環境，並且停滯在 Creating 狀態

- 我嘗試建立環境，但狀態顯示為 Create failed
- 我嘗試選取 VPC 並收到Network Failure錯誤
- 我嘗試建立環境並收到服務、分割區或資源「必須傳遞」錯誤
- 我嘗試建立環境，並顯示狀態為 ，Available但當我嘗試存取 Airflow UI 時，會顯示 Empty Reply from Server或 502 Bad Gateway錯誤
- 我嘗試建立環境，而我的使用者名稱是一組隨機字元名稱
- 更新環境
  - 我嘗試變更環境類別，但更新失敗
- 存取環境
  - 我無法存取 Apache Airflow UI

更新 `requirements.txt`

下列主題說明您在更新 時可能收到的錯誤`requirements.txt`。

我指定了新版本的 `requirements.txt`，需要超過 20 分鐘的時間來更新我的環境

如果您的環境需要超過 20 分鐘才能安裝新版本的`requirements.txt`檔案，則環境更新會失敗，而 Amazon MWAA 會轉返至容器映像的最後一個穩定版本。

1. 檢查套件版本。建議您一律為 中的 Python 相依性指定特定版本 (==) 或最大版本 (<=)`requirements.txt`。
2. 檢查 Apache Airflow 日誌。如果您啟用 Apache Airflow 日誌，請在 CloudWatch 主控台的日誌群組頁面上確認您的日誌群組已成功建立。如果您取得空白日誌，最常見的原因是缺少寫入日誌之 CloudWatch 或 Amazon S3 執行角色的許可。若要進一步了解，請參閱 執行角色。
3. 檢查 Apache Airflow 組態選項。如果您使用的是 Secrets Manager，請確認您指定為 Apache Airflow 組態選項的鍵值對已正確設定。若要進一步了解，請參閱 the section called “設定 Secrets Manager”。
4. 檢查 VPC 網路組態。若要進一步了解，請參閱 the section called “環境卡住”。
5. 檢查執行角色許可。執行角色是具有許可政策的 AWS Identity and Access Management (IAM) 角色，授予 Amazon MWAA 代表您叫用其他服務 AWS （ 例如 Amazon S3、CloudWatch、Amazon SQS、Amazon ECR) 資源的許可。您的客戶受管金鑰或 AWS擁有的金鑰也需要被允許存取。若要進一步了解，請參閱 執行角色。
6. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。

外掛程式

下列主題說明您在設定或更新 Apache Airflow 外掛程式時可能遇到的問題。

Amazon MWAA 是否支援實作自訂 UI？

從 Apache Airflow v2.2.2 開始，Amazon MWAA 支援在 Apache Airflow Web 伺服器上安裝外掛程式，並實作自訂 UI。如果您的 Amazon MWAA 環境執行 Apache Airflow 2.0.2 版或更新版本，您將無法實作自訂 UI。

如需版本管理和升級現有環境的詳細資訊，請參閱 版本。

建立儲存貯體

下列主題說明您在建立 Amazon S3 儲存貯體時可能收到的錯誤。

我無法選取 S3 封鎖公開存取設定的選項

Amazon MWAA 環境的執行角色需要 Amazon S3 儲存貯體上 `GetBucketPublicAccessBlock`動作的許可，才能驗證儲存貯體封鎖的公開存取。建議下列步驟：

1. 依照步驟將 JSON 政策連接至您的執行角色。
2. 連接下列 JSON 政策：

```
{
  "Effect":"Allow",
  "Action":[
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Resource":[
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ]
}
```

   使用 Amazon S3*3 ######## amzn-s3-demo-bucket* 中的範例預留位置。
3. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。

建立 環境

下列主題說明您在建立環境時可能收到的錯誤。

我嘗試建立環境，並且停滯在 **Creating** 狀態

建議下列步驟：

1. 使用公有路由檢查 VPC 網路。如果您使用具有網際網路存取的 Amazon VPC，請確認下列事項：
    - 您的 Amazon VPC 已設定為允許 Amazon MWAA 環境使用的不同 AWS 資源之間的網路流量，如 中所定義the section called “關於聯網”。例如，您的 VPC 安全群組必須允許自我參考規則中的所有流量，或選擇性地指定 HTTPS 連接埠範圍 443 和 TCP 連接埠範圍 5432 的連接埠範圍。
2. 檢查具有私有路由的 VPC 網路。如果您使用的是沒有網際網路存取的 Amazon VPC，請確認下列事項：
    - 您的 Amazon VPC 已設定為允許 Amazon MWAA 環境不同 AWS 資源之間的網路流量，如中所定義the section called “關於聯網”。例如，您的兩個私有子網路不得具有 NAT 閘道（或 NAT 執行個體）的路由表，也不得具有網際網路閘道。
3. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。

我嘗試建立環境，但狀態顯示為 **Create failed**

建議下列步驟：

1. 檢查 VPC 網路組態。若要進一步了解，請參閱 the section called “環境卡住”。
2. 檢查使用者許可。Amazon MWAA 會在建立環境之前，針對使用者的登入資料執行試轉。您的 AWS 帳戶 可能沒有在 AWS Identity and Access Management (IAM) 中建立環境部分資源的許可。例如，如果您選擇私有網路 Apache Airflow 存取模式，您的管理員 AWS 帳戶 必須已為您的環境授予 AmazonMWAAFullConsoleAccess 存取控制政策的存取權，這可讓您的帳戶建立 VPC 端點。
3. 檢查執行角色許可。執行角色是具有許可政策的 AWS Identity and Access Management (IAM) 角色，授予 Amazon MWAA 代表您叫用其他服務 AWS（例如 Amazon S3、CloudWatch、Amazon SQS、Amazon ECR) 資源的許可。您的客戶受管金鑰或 AWS擁有的金鑰也需要被允許存取。若要進一步了解，請參閱 執行角色。

4. 檢查 Apache Airflow 日誌。如果您啟用 Apache Airflow 日誌，請在 CloudWatch 主控台的日誌群組頁面上確認您的日誌群組已成功建立。如果您取得空白日誌，最常見的原因是缺少寫入日誌之 CloudWatch 或 Amazon S3 執行角色的許可。若要進一步了解，請參閱 執行角色。
5. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。
6. 如果您使用沒有網際網路存取的 Amazon VPC，請確保您已建立 Amazon S3 閘道端點，並授予 Amazon ECR 存取 Amazon S3 所需的最低權限。若要進一步了解如何建立 Amazon S3 閘道端點，請參閱下列各項：
   - 在沒有網際網路存取的情況下建立 Amazon VPC 網路
   - 《Amazon Elastic Container Registry 使用者指南》中的建立 Amazon S3 閘道端點

我嘗試選取 VPC 並收到**Network Failure**錯誤

建議下列步驟：

- 如果您在建立環境`Network Failure`時嘗試選取 Amazon VPC 時發生錯誤，請關閉任何執行中的瀏覽器內代理程式，然後再試一次。

我嘗試建立環境並收到服務、分割區或資源「必須傳遞」錯誤

建議下列步驟：

- 您可能會收到此錯誤，因為您為 Amazon S3 儲存貯體指定的 URI 在 URI 結尾包含 '/'。我們建議移除路徑中的 '/'。值必須採用下列格式：

```
s3://amzn-s3-demo-bucket
```

我嘗試建立環境，並顯示狀態為 ，**Available**但當我嘗試存取 Airflow UI 時，會顯示 **Empty Reply from Server**或 **502 Bad Gateway**錯誤

建議下列步驟：

1. 檢查 VPC 安全群組組態。若要進一步了解，請參閱 the section called “環境卡住”。
2. 確認您在 中列出的任何 Apache Airflow 套件`requirements.txt`對應至您在 Amazon MWAA 上執行的 Apache Airflow 版本。若要進一步了解，請參閱 安裝 Python 相依性。

3. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。

我嘗試建立環境，而我的使用者名稱是一組隨機字元名稱

- Apache Airflow 的使用者名稱上限為 64 個字元。如果您的 AWS Identity and Access Management (IAM) 角色超過此長度，則會使用雜湊演算法來減少它，同時保持唯一。

更新環境

下列主題說明您在更新環境時可能收到的錯誤。

我嘗試變更環境類別，但更新失敗

如果您將環境更新為不同的環境類別 （ 例如將 變更為 `mw1.medium` `mw1.small`)，且更新環境的請求失敗，則環境狀態會進入 `UPDATE_FAILED` 狀態，且環境會回復為 ，並根據環境的先前穩定版本計費。

建議下列步驟：

1. 在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。
2. 若要執行故障診斷指令碼來檢查 Amazon MWAA 環境的 Amazon VPC 網路設定和組態，請參閱 GitHub AWS 支援工具中的驗證環境指令碼。

存取環境

下列主題說明您在存取 環境時可能收到的錯誤。

我無法存取 Apache Airflow UI

建議下列步驟：

1. 檢查使用者許可。您可能尚未獲得許可政策的存取權，可用於存取 Apache Airflow UI。若要進一步了解，請參閱 the section called “存取 Amazon MWAA 環境”。
2. 檢查網路存取。這可能是因為您選擇了私有網路存取模式。如果 Apache Airflow UI 的 URL 格式如下`387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-`

`west-2.airflow.amazonaws.com`，表示您使用 Apache Airflow Web 伺服器的私有路由。您可以將 Apache Airflow 存取模式更新為公有網路存取模式，或建立機制來存取 Apache Airflow Web 伺服器的 VPC 端點。若要進一步了解，請參閱 the section called “管理對 VPC 端點的存取”。

故障診斷：CloudWatch Logs 和 CloudTrail 錯誤

本頁面上的主題包含 Amazon CloudWatch Logs 的解決方案，以及您在 Amazon Managed Workflows for Apache Airflow 環境中可能遇到的 AWS CloudTrail 錯誤。

內容

- 日誌
  - 我找不到任務日誌，或收到Reading remote log from Cloudwatch log_group錯誤
  - 任務在沒有任何日誌的情況下失敗
  - 我在 CloudTrail 中收到ResourceAlreadyExistsException錯誤
  - 我在 CloudTrail 中收到Invalid request錯誤
  - 我在 Apache Airflow 日誌Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory中取得
  - 我收到排程器日誌中的 psycopg2 'server 意外關閉連線'
  - 我在 DAG 處理日誌Executor reports task instance %s finished (%s) although the task says its %s中取得
  - 我在任務日誌Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.中取得

日誌

下列主題說明您在存取 Apache Airflow 日誌時可能收到的錯誤。

我找不到任務日誌，或收到**`Reading remote log from Cloudwatch log_group`**錯誤

Amazon MWAA 已設定 Apache Airflow 直接從 Amazon CloudWatch Logs 讀取和寫入日誌。如果工作者無法啟動任務，或無法寫入任何日誌，您會參考錯誤：

```
*** Reading remote log from Cloudwatch log_group: airflow-environmentName-Task
 log_stream: DAG_ID/TASK_ID/timestamp/n.log.Could not read remote logs from log_group:
 airflow-environmentName-Task log_stream: DAG_ID/TASK_ID/time/n.log.
```

- 建議下列步驟：

  a. 確認您已在環境的 INFO層級啟用任務日誌。如需詳細資訊，請參閱 在 Amazon CloudWatch 中存取 Airflow 日誌。

  b. 確認環境執行角色具有正確的許可政策。

  c. 確認您的運算子或任務正常運作、有足夠的資源來剖析 DAG，以及有適當的 Python 程式庫可供載入。若要驗證您是否有正確的相依性，請嘗試消除匯入，直到您找到造成問題的匯入為止。我們建議使用 aws-mwaa-docker-images 測試您的 Python 相依性。

任務在沒有任何日誌的情況下失敗

如果任務在工作流程中失敗，而且找不到失敗任務的任何日誌，請檢查您是否在預設引數中設定 queue 參數，如下所列。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

# Setting queue argument to default.
default_args = {
 "start_date": days_ago(1),
 "queue": "default"
}

with DAG(dag_id="any_command_dag", schedule_interval=None, catchup=False,
 default_args=default_args) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="{{ dag_run.conf['command'] }}"
    )
```

若要解決此問題，queue請從程式碼中移除 ，然後再次叫用 DAG。

我在 CloudTrail 中收到**ResourceAlreadyExistsException**錯誤

```
"errorCode": "ResourceAlreadyExistsException",
```

```
    "errorMessage": "The specified log stream already exists",
    "requestParameters": {
        "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
        "logStreamName": "scheduler_cross-account-eks.py.log"
    }
```

某些 Python 需求，例如將 Amazon MWAA 用來與 CloudWatch 通訊的`watchtower`程式庫`apache-airflow-backport-providers-amazon`轉返至較舊版本。建議下列步驟：

- 將下列程式庫新增至您的 `requirements.txt`

```
watchtower==1.0.6
```

我在 CloudTrail 中收到**`Invalid request`**錯誤

```
Invalid request provided: Provided role does not have sufficient permissions for s3
 location airflow-xxx-xxx/dags
```

如果您使用相同的 CloudFormation 範本建立 Amazon MWAA 環境和 Amazon S3 儲存貯體，則需要在 CloudFormation 範本中新增`DependsOn`區段。這兩個資源 (MWAA 環境和 MWAA 執行政策) 在其中具有相依性 CloudFormation。建議下列步驟：

- 將下列**`DependsOn`**陳述式新增至您的 CloudFormation 範本。

```
...
  MaxWorkers: 5
  NetworkConfiguration:
    SecurityGroupIds:
      - !GetAtt SecurityGroup.GroupId
    SubnetIds: !Ref subnetIds
  WebserverAccessMode: PUBLIC_ONLY
DependsOn: MwaaExecutionPolicy

 MwaaExecutionPolicy:
 Type: AWS::IAM::ManagedPolicy
 Properties:
   Roles:
    - !Ref MwaaExecutionRole
  PolicyDocument:
    Version: 2012-10-17
```

```
    Statement:
      - Effect: Allow
        Action: airflow:PublishMetrics
        Resource:
...
```

如需範例，請參閱 Amazon Managed Workflows for Apache Airflow 的快速入門教學課程。

我在 Apache Airflow 日誌**Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory**中取得

- 建議下列步驟：
  - 如果您使用的是 Apache Airflow v2，請將 新增`core.lazy_load_plugins : False`為 Apache Airflow 組態選項。若要進一步了解，請參閱使用組態選項載入 2 中的外掛程式。

我收到排程器日誌中的 psycopg2 'server 意外關閉連線'

如果您收到類似以下的錯誤，您的 Apache Airflow 排程器可能已耗盡資源。

```
2021-06-14T10:20:24.581-05:00    sqlalchemy.exc.OperationalError:
 (psycopg2.OperationalError) server closed the connection unexpectedly
2021-06-14T10:20:24.633-05:00    This probably means the server terminated abnormally
2021-06-14T10:20:24.686-05:00    before or while processing the request.
```

建議下列步驟：

- 考慮升級至 Apache Airflow v2.0.2，您可以使用它來指定最多 5 個排程器。

我在 DAG 處理日誌**Executor reports task instance %s finished (%s) although the task says its %s**中取得

如果您收到類似下列的錯誤，長時間執行的任務可能已達到 Amazon MWAA 的任務時間限制。任何一個 Airflow 任務的 Amazon MWAA 限制為 12 小時，以防止任務卡在佇列中並封鎖自動擴展等活動。

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
 %s) Was the task killed externally
```

建議下列步驟：

- 考慮將任務分成多個較短的執行中任務。Airflow 通常具有模型，因此運算子是非同步的。它會叫用外部系統上的活動，並輪詢 Apache Airflow Sensors 以檢查何時完成。如果感應器故障，則可以安全地重試，而不會影響操作員的功能。

我在任務日誌`Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.`中取得

如果您收到類似下列的錯誤，您環境的執行角色可能不包含建立任務日誌之日誌串流的許可政策。

```
Could not read remote logs from log_group: airflow-*{*environmentName}-Task
 log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

建議下列步驟：

- 使用其中一個範例政策來修改您環境的執行角色the section called “執行角色”。

您可能也在`requirements.txt`檔案中指定了與 Apache Airflow 版本不相容的提供者套件。例如，如果您使用的是 Apache Airflow v2.0.2，您可能已指定套件，例如 apache-airflow-providers-databricks 套件，其僅與 Airflow 2.1+ 相容。

建議下列步驟：

1. 如果您使用的是 Apache Airflow v2.0.2，請修改 `requirements.txt` 檔案並新增 `apache-airflow[databricks]`。這會安裝與 Apache Airflow v2.0.2 相容的正確 Databricks 套件版本。
2. 在 GitHub 上使用 aws-mwaa-docker-images，在本機測試您的 DAGs、自訂外掛程式和 Python 相依性。

Amazon MWAA 使用者指南歷史記錄

下表說明 Amazon MWAA 服務文件的重要新增項目，自 2020 年 11 月開始。若要接收有關本文件更新的通知，請訂閱 RSS 摘要。

變更	描述	日期
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow 2.11.0 版。此更新包含更新後提供者套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow 2.11.0 版的詳細資訊。 • 版本 • the section called “特定版本提供者套件”	2026 年 1 月 7 日
新增 Apache Airflow v3 支援	更新文件以包含對 Apache Airflow v3 的支援 • 程式碼範例 • 版本	2025 年 10 月 1 日
IPv6 更新	新增 IPv6 支援的相關資訊。 • the section called “關於聯網”	2025 年 8 月 26 日
環境更新	`GRACEFUL FORCED` 如果您在環境處於 `MAINTENANCE` 狀態時執行更新，則新增從 `workerReplacementStrategy` 變更為 的注意事項。 • the section called “更新環境”	2025 年 8 月 6 日

版本棄用資訊	更新版本棄用主題，以包含 Apache Airflow v2.4.3、Apache Airflow v2.5.1 和 Apache Airflow v2.6.3 的棄用通知和時間表。 • the section called “Apache Airflow 已棄用版本”	2025 年 6 月 24 日
新增了新的環境類別：mw1.micro	Amazon MWAA 現在提供新的環境類別：mw1.micro。 • the section called “設定環境類別” • the section called “Apache Airflow 的效能調校”	2024 年 11 月 19 日
支援更簡單的方法來存取 Apache Airflow REST API	Amazon MWAA 現在提供使用 AWS 登入資料與 Apache Airflow REST API 互動的簡化方法。 • the section called “使用 Apache Airflow REST API” • the section called “Apache Airflow Rest API 存取”	2024 年 10 月 23 日
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow 2.10.1 版。此更新包含更新提供者套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow v2.10.1 的詳細資訊。 • 版本 • the section called “特定版本提供者套件”	2024 年 9 月 26 日

新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow 2.9.2 版。此更新包含更新後提供者套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow 2.9.2 版的詳細資訊。 • 版本 • the section called “特定版本提供者套件”	2024 年 7 月 9 日
Amazon MWAA 支援設定自訂 Web 伺服器網域名稱	Amazon MWAA 支援為沒有網際網路存取的私有環境設定自訂 Web 伺服器網域名稱。此更新包含下列新主題，說明設定新的自訂網域。 • the section called “設定自訂網域”	2024 年 6 月 18 日
Amazon MWAA 支援 Web 伺服器自動擴展和 Apache Airflow REST API	Amazon MWAA 現在支援 Web 伺服器的自動擴展，以及存取和使用 Apache Airflow REST API 的功能。 • the section called “設定 Web 伺服器自動擴展” • the section called “使用 Apache Airflow REST API”	2024 年 5 月 16 日
改善自動擴展行為的描述	更新了下列主題，以反映當工作者以 Fargate 工作者規模縮減的形式接收新任務時，新的 Amazon MWAA 自動擴展行為。 • the section called “設定工作者自動擴展”	2024 年 5 月 10 日

支援較大的執行個體大小	Amazon MWAA 現在支援兩個較大的執行個體大小選項，適用於較大的工作負載：`mw1.xlarge` 、 和 `mw1.2xlarge` • the section called “環境功能”	2024 年 4 月 16 日
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow v2.8.1。此更新包含更新後提供者套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow 2.8.1 版的詳細資訊。 • 版本 • the section called “特定版本提供者套件”	2024 年 2 月 22 日
支援共用的 Amazon VPC	Amazon MWAA 支援為使用 Amazon OpenSearch Service 的組織建立跨帳戶環境，以使用擁有者帳戶中的中央共用 Amazon VPC 來管理 Amazon MWAA 資源。在此啟動過程中，Amazon MWAA 可讓您選擇建立和管理自己的 Amazon VPC 端點。 • the section called “管理您自己的 Amazon VPC 端點”	2023 年 11 月 15 日

新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow 2.7.2 版。此更新包含更新提供者套件的相關資訊，以及有關在 Amazon MWAA 上使用 Apache Airflow 2.7.2 版的詳細資訊。 • 版本 • the section called “特定版本提供者套件”	2023 年 11 月 6 日
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow v2.6.3。此更新包含更新供應商套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow 2.6.3 版的詳細資訊， • 版本	2023 年 8 月 9 日
版本棄用資訊	更新版本棄用主題，以包含 Apache Airflow v2.0.2 和 Apache Airflow v2.2.2 的棄用通知和時間表。 • the section called “Apache Airflow 已棄用版本”	2023 年 7 月 31 日
新的主題和使用案例	Amazon MWAA 支援次要版本升級。此更新包含下列新主題，說明如何升級環境，並確保您的工作流程資源與您升級到的 Apache Airflow 版本相容： • the section called “變更版本”	2023 年 6 月 5 日

更新主題	更新了客戶受管 IAM 政策，將 Amazon MWAA 的完整主控台和 API 存取權授予使用者。更新說明為什麼您必須提供 許可`iam:PassRole` ，以允許使用者將角色傳遞給 Amazon MWAA。Amazon MWAA 使用這些許可來代表使用者執行動作。 • the section called “存取 Amazon MWAA 環境”	2023 年 4 月 12 日
新指引	更新有關將 AWS Secrets Manager 設定為 Amazon MWAA 後端的主題，以提供使用查詢模式的指引。使用查詢模式可縮小 Apache Airflow 搜尋的秘密，並減少 Amazon MWAA 呼叫 Secrets Manager 以擷取連線和變數的 API 次數。這可降低使用 Secrets Manager 做為後端的相關成本。 • 建立 Secrets Manager 後端做為 Apache Airflow 組態選項	2023 年 4 月 12 日
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow v2.5.1。此更新包含更新後提供者套件的相關資訊，以及在 Amazon MWAA 上使用 Apache Airflow 2.5.1 版的詳細資訊， • 版本	2023 年 4 月 11 日

新的主題和使用案例	新增了有關搭配 Amazon MWAA 環境使用啟動指令碼的新主題。本主題說明為現有環境設定啟動指令碼、使用它來安裝 Linux 執行期，以及設定環境變數。 • the section called “使用啟動指令碼”	2023 年 4 月 3 日
已更新私有 Web 伺服器存取的章節	已更新私有 Web 伺服器存取的下列主題。更新闡明，在具有私有 Web 伺服器存取權的環境中，您必須使用 Python wheel 封存檔 (`.whl`) 來封裝和安裝相依性。 • 私有 Web 伺服器存取模式	2023 年 2 月 24 日
新增有關已棄用 Apache Airflow 版本的資訊	以有關 Amazon MWAA 如何管理取代 Apache Airflow 版本的新資訊更新版本主題。移除有關升級至較新版本的 Apache Airflow 的章節，以及描述 Apache Airflow v1 和 Apache Airflow v2 之間變更的章節。如需遷移至 Apache Airflow 新版本的詳細資訊，請參閱 Amazon MWAA 遷移指南。 • the section called “Apache Airflow 已棄用版本” • the section called “Apache Airflow 版本支援和常見問答集”	2023 年 2 月 17 日

Amazon MWAA 容器指標中的修正	更新容器指標主題，並移除維Cluster度中不存在的一組錯誤指標。新增額外章節，說明如何透過繪製AdditionalWorker 元件的 CPUUtilization 或 MemoryUtilization 指標，並將統計資料類型設定為 ，來評估環境在特定時間使用的其他工作者數量Sample Count。 • the section called “評估其他工作者和 Web 伺服器容器的數量”	2023 年 1 月 20 日
新的 Apache Airflow 版本	Amazon MWAA 現在支援 Apache Airflow 2.4.3 版。此更新包含更新提供者套件的相關資訊、在 Amazon MWAA 上使用 Apache Airflow v2.4.3 的詳細資訊，以及 Amazon MWAA 上每個 Apache Airflow 版本支援哪些功能的合併資訊。 • 版本	2023 年 1 月 5 日

更新了服務連結角色的主題	更新 Amazon MWAA 用來代表您建立和管理 AWS 資源的服務連結角色相關資訊，包括如何在不再需要時刪除服務連結角色的相關資訊。這包括更新的服務連結角色許可政策，允許 Amazon MWAA 在AWS/MWAA命名空間中發佈其他 CloudWatch 指標。 • the section called “服務連結角色”	2022 年 11 月 18 日
有關服務指標的新主題	新增了主題，說明 Amazon MWAA 在 AWS/MWAA 命名空間中發出的服務指標。這些包括 Amazon ECS 叢集指標排程器、工作者和 Web 伺服器、允許 Amazon MWAA 解耦排程器和工作者之佇列的 Amazon SQS 指標，以及中繼資料資料庫的 Amazon RDS 指標。 • the section called “容器、佇列和資料庫指標”	2022 年 11 月 18 日
新主題	已新增修改限制條件檔案的新指引，以指定要搭配 Amazon MWAA 環境使用的提供者套件新版本。 • the section called “限制條件檔案”	2022 年 11 月 18 日

已更新常見問答集項目	更新 Amazon MWAA HIPAA 資格的相關資訊。 • the section called “HIPAA 合規”	2022 年 11 月 15 日
新主題	新增在 Amazon MWAA 執行角色信任政策中使用 aws:SourceArn 和 aws:SourceAccount 全域條件內容金鑰的新主題，以防止跨服務混淆代理人。 • the section called “預防跨服務混淆代理人”	2022 年 10 月 21 日
新的範例程式碼	新增了將自訂作業系統層級指標寫入 CloudWatch 的更新指示和 DAG 程式碼範例。 • the section called “使用 DAG 撰寫自訂指標”	2022 年 9 月 13 日
新的範例程式碼	新增了更新的指示和擷取 Apache Airflow CLI 字符的新 AWS Lambda Python 程式碼範例，然後在指定的 Amazon MWAA 環境中叫用 DAG。 • the section called “使用 Lambda 叫用 DAGs ”	2022 年 9 月 12 日
新的架構圖	新增了新的架構圖，示範具有公有和私有 Web 伺服器的 Amazon MWAA 環境。 • the section called “Apache Airflow 存取模式”	2022 年 9 月 12 日

新的範本程式碼	新增了更新的指示和擷取 Apache Airflow CLI 字符的新 DAG 程式碼範例，然後在不同的 Amazon MWAA 環境中叫用另一個 DAG。 • the section called “在不同的環境中叫用 DAGs ”	2022 年 8 月 16 日
新的範本程式碼	新增了更新的指示和新的 DAG，查詢環境的 Aurora PostgreSQL 以取得中繼資料資訊、將結果寫入 CSV 檔案，並將檔案存放在 Amazon S3 中。 • the section called “將環境中繼資料匯出至 Amazon S3”	2022 年 8 月 12 日
新的範本程式碼	新增了更新的指示和新的 DAG，可在執行時間重新整理 AWS CodeArtifact 權杖並將結果儲存在 Amazon S3 中。 • the section called “在執行時間重新整理 AWS CodeArtifact 權杖”	2022 年 8 月 3 日
新的範本程式碼	新增在 Amazon MWAA `ECSOperator` 中使用 的更新指示和 DAG 程式碼範例。 • the section called “使用 `ECSOperator` ”	2022 年 7 月 26 日

新的範本程式碼	新增在 Amazon MWAA SSHOperator 中使用 的更新指示和 DAG 程式碼範例。 • the section called “使用 SSHOperator ”	2022 年 7 月 15 日
新的範本程式碼	新增將 dbt Postgres 與 Amazon MWAA 搭配使用的新指示和 DAG 程式碼範例。 • the section called “搭配 Amazon MWAA 使用 dbt”	2022 年 6 月 17 日
新的主題和使用案例	新增了使用 Python wheel 檔案為具有公有和私有存取權的 Amazon MWAA 環境安裝相依性的新指示和 DAG 程式碼範例。 • 使用 Python wheel 管理相依性	2022 年 5 月 13 日
新的主題和使用案例	新增了有關選擇 Amazon MWAA 傳送至 CloudWatch 之 Apache Airflow 指標的新指引。 • 選擇報告哪些 Apache Airflow 指標	2022 年 4 月 19 日
新指南	Amazon MWAA 提供從自我管理部署以及現有 Amazon MWAA 環境遷移 Apache Airflow 工作流程的遷移指南。 • Amazon MWAA 遷移指南	2022 年 3 月 7 日

新的主題和使用案例	新增使用 Apache Airflow 的新安全最佳實務，包括偵測 Apache Airflow 使用者權限變更的解決方案。 • the section called “Apache Airflow 中的安全最佳實務”	2022 年 2 月 18 日
新的範本程式碼	新增使用 Pendulum 建立時區感知 DAGs 的新程式碼範例，並釐清如何使用自訂外掛程式變更建立 Apache Airflow 日誌的時區。 • the section called “變更 DAG 的時區”	2022 年 2 月 11 日
Apache Airflow 2.2.2 版啟動	Amazon Managed Workflows for Apache Airflow 現在支援 Apache Airflow v2.2.2。從 v2.2 開始，Amazon MWAA 將直接在 Apache Airflow Web 伺服器上安裝 Python 套件和自訂外掛程式，讓您更靈活地管理環境。如需詳細資訊，請參閱下列內容。 • Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本. • Apache Airflow 文件網站上的 Apache Airflow v2.2.2 變更日誌。	2022 年 1 月 27 日

新的教學課程	新增了新的教學課程，示範建立新的自訂 Apache Airflow 角色，並將角色指派給從 IAM 映射的 Apache Airflow 使用者，以限制使用者對指定 DAGs 子集的存取。 • 教學課程：限制 Amazon MWAA 使用者存取 DAGs的子集	2021 年 12 月 8 日
修正項目	已修正設定 值`scheduler.min_file_process_interval` 以最佳化 CPU 用量的最佳實務建議。新增 IAM 政策範例，授予對執行角色中 Secrets Manager 資源的存取權。新增使用 Secrets Manager 條件金鑰的疑難排解主題。 • 排程器剖析 DAGs 的效能調校 • 提供 Amazon MWAA 存取 Secrets Manager 私密金鑰的許可 • 在 Secrets Manager 的 Amazon MWAA 執行角色中設定條件索引鍵	2021 年 11 月 22 日

新的範例程式碼	新增下列新程式碼範例，用於修改使用自訂外掛程式處理 DAGs的時區，以及從 bash 運算子內叫用 `dags backfill` Apache Airflow CLI 命令的新故障診斷主題。 • the section called “變更 DAG 的時區” • 使用 bash 運算子的回填 CLI 命令	2021 年 11 月 1 日
修正項目	修正 Amazon ECS 運算子程式碼範例的問題，並釐清 Amazon MWAA 執行角色所需的額外許可，以允許環境存取 CloudWatch Logs 中的 Amazon ECS 任務日誌群組。 • Amazon ECS 運算子許可。	2021 年 10 月 26 日
新的範例程式碼	新增程式碼範例，查詢 Aurora PostgreSQL 資料庫以取得與 DAG 執行相關的資訊，並將結果寫入 Amazon S3 上儲存CSV的檔案。 • the section called “將環境中繼資料匯出至 Amazon S3”.	2021 年 10 月 1 日
修正項目	更正 Amazon MWAA 如何自動將新物件和變更物件從目標 Amazon S3 儲存貯體同步至排程器和工作者的相關資訊。 • DAG 資料夾的運作方式。	2021 年 10 月 1 日

現在支援	Amazon MWAA 現在支援 Apache Airflow 2.0+ 的其他供應商套件。若要進一步了解支援的套件，請參閱下列內容：	2021 年 9 月 24 日
新的命令和程序	新增在不使用網際網路存取的情況下使用 Amazon VPC 時建立 Amazon S3 閘道端點的其他指導和 AWS CLI 命令範例： • 建立沒有網際網路存取的 Amazon VPC 網路。	2021 年 9 月 24 日
新的主題和使用案例	新增下列變更： • 新增了在 中使用 Amazon Elastic Container Service 運算子的新程式碼範例the section called “使用 `ECSOperator` ”。 • 針對在 中設定 Apache Airflow 外掛程式時的問題，新增新的故障診斷主題the section called “外掛程式”。	2021 年 9 月 19 日

新的支援區域	Amazon MWAA 現已在下列區域提供： • 亞太區域 (孟買) – ap-south-1 • 亞太區域 (首爾) – ap-northeast-2 • 歐洲 (倫敦) – eu-west-2 • 歐洲 (巴黎) – eu-west-3 • 加拿大 (中部) – ca-central-1 • 南美洲 (聖保羅) – sa-east-1 如需區域可用性和服務端點的詳細資訊，請參閱下列內容： • 中的 Amazon MWAA 端點和配額AWS 一般參考。	2021 年 8 月 31 日
新的主題和使用案例	新增下列變更： • 更新範例政策，以允許 Amazon MWAA 在 中擷取帳戶層級的 Amazon S3 設定 (`s3:GetAccountPublicAccessBlock`)Amazon MWAA 執行角色。	2021 年 8 月 27 日

修正項目	新增下列變更： • 已修正 CloudFormation 範本，以針對 中的安全群組使用自我參考傳入規則建立 VPC 網路。 • 已修正 CloudFormation 範本，以針對 中的安全群組使用自我參考傳入規則Amazon Managed Workflows for Apache Airflow 的快速入門教學課程。	2021 年 8 月 27 日
新的主題和使用案例	新增下列變更： • 已將 DAG 裝飾項目新增至 Apache Airflow v2.0.2 支援的清單。 Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本	2021 年 8 月 20 日

新的主題和使用案例	新增下列變更： • 已新增`celery.sync_parallelism` 使用案例至 Amazon MWAA 上 Apache Airflow 的效能調校。 • 已將服務端點新增至配額頁面，並將名稱變更為 Amazon Managed Workflows for Apache Airflow 服務端點和配額。 • 根據 的使用者意見回饋，釐清聯網先決條件開始使用 Amazon Managed Workflows for Apache Airflow。 • 已將 `dags list-runs` 和 `dags next-execution` 移至 中不支援的 Airflow CLI 命令Apache Airflow CLI 命令參考。	2021 年 8 月 13 日
新的範例程式碼	新增下列變更： • 新增在 中設定、取得或刪除 Apache Airflow v2.0.2 變數的 bash 範例Apache Airflow CLI 命令參考。 • 已將 Apache Airflow v2.0.2 相依性和 Airflow 連線範例新增至 搭配使用 Amazon MWAA 與 Amazon RDS for Microsoft SQL Server。	2021 年 8 月 13 日

修正項目	新增下列變更： • 已根據 的使用者意見回饋修正 Python 程式碼範例使用建立 SSH 連線 SSHOperator 。	2021 年 8 月 13 日
新的主題和使用案例	新增下列變更： • variables set 移至中支援的 Airflow CLI 命令Apache Airflow CLI 命令參考。 • 安裝 Python 相依性 根據使用者意見回饋，將 2.0.2 版的變更摘要從 Airflow 版本頁面新增至 。 • Apache Airflow CLI 命令參考 根據使用者意見回饋，將 2.0.2 版的變更摘要從 Airflow 版本頁面新增至 。 • 連線類型概觀 根據使用者意見回饋，將 2.0.2 版中的變更摘要從 Airflow 版本頁面新增至 。 • 安裝自訂外掛程式 根據使用者意見回饋，將 2.0.2 版中的變更摘要從 Airflow 版本頁面新增至 。 • 新增或更新 DAGs 根據使用者意見回饋，將 2.0.2 版中的變更摘要從 Airflow 版本頁面新增至 。	2021 年 8 月 6 日

新的範本程式碼	新增下列變更： • 已將 Apache Airflow 2.0.2 版範本程式碼新增至 使用 DAG 在 CLI 中匯入變數。 • 已將 Apache Airflow 2.0.2 版範本程式碼新增至 使用 Lambda 函數叫用 DAGs。	2021 年 8 月 6 日
新的主題和使用案例	新增下列變更： • 在 的 Airflow UI 中新增了「我找不到我的連線」的故障診斷主題對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 Amazon VPCsAmazon MWAA 支援的清單新增至關於 Amazon MWAA 上的聯網。	2021 年 7 月 29 日
修正項目	新增下列變更： • 已根據使用者在 列印 Web 登入字符的意見回饋，修正 Python 程式碼範例建立 Apache Airflow Web 伺服器存取字符。 • 已根據使用者意見回饋修正 Snowflake 連線主題，以針對位於 的倉儲參數使用單引號對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 7 月 29 日

移除或移動主題	新增下列變更： • 已重組現有頁面，以在 中包含所有監控和指標文件頁面Amazon Managed Workflows for Apache Airflow 的監控和指標。 • CloudWatch 中的 Apache Airflow 環境指標 移至監控和指標導覽功能表。	2021 年 7 月 23 日
新指南	新增下列變更： • 已建立 安裝在 Amazon MWAA 環境上的 Apache Airflow 提供者套件。 • 已建立 Amazon MWAA 的監控概觀。 • 已建立 在 中存取稽核日誌 AWS CloudTrail。 • 已建立 在 Amazon CloudWatch 中存取 Airflow 日誌。	2021 年 7 月 23 日

修正項目	新增下列變更： • 修正以使用者意見回饋為基礎的 Python 程式碼範例，以正確順序產生 Airflow 連線字串，並在 中新增連接埠參數使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。 • 根據 中的使用者意見回饋，新增在本機安裝 unzip 套件的步驟使用 Oracle 建立自訂外掛程式。	2021 年 7 月 23 日
新的主題和使用案例	新增下列變更： • 在 AWS 新增 DMS Operators 的主題Amazon MWAA 常見問答集。 • 已將遠端日誌錯誤的疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • `variables set` 移至 中不支援的 Airflow CLI 命令Apache Airflow CLI 命令參考。	2021 年 7 月 16 日

新的主題和使用案例	新增下列變更： • 新增循序步驟，根據 的使用者意見回饋來建立 requirements.txt 檔案安裝 Python 相依性。 • 新增循序步驟，根據 的使用者意見回饋來建立 plugins.zip 檔案安裝自訂外掛程式。 • 在 Amazon Managed Workflows for Apache Airflow API 參考指南的 API 參考指南中，在整個使用者指南中新增了交叉參考連結。 • 已新增為何外掛程式未列在 Airflow 2.0 Admin > 外掛程式選單中的主題Amazon MWAA 常見問答集。	2021 年 7 月 9 日
新指南	新增下列變更： • 已建立 刪除 Amazon S3 上的檔案。	2021 年 7 月 9 日
新的主題和使用案例	新增下列變更： • 在 新增支援值的清單使用客戶受管金鑰進行加密。 • 根據 中的使用者意見回饋，更新並釐清私有儲存庫 URL 的範例在 requirements.txt 中管理 Python 相依性。	2021 年 7 月 2 日

新的範本程式碼	新增下列變更： • 新增 Apache Airflow v1.10.12 範例程式碼，以在 中使用私有金鑰 AWS Secrets Manager 進行 SSH 連線使用 建立 SSH 連線 `SSHOperator` 。	2021 年 7 月 2 日
新的主題和使用案例	新增下列變更： • 已將 StartedTaskInstances 和 FinishedTaskInstances 指標新增至 CloudWatch 中的 Apache Airflow 環境指標。	2021 年 6 月 25 日
新的範本程式碼	新增下列變更： • 在 新增 Apache Airflow 2.0.2 版的範例程式碼搭配使用 Amazon MWAA 與 Amazon EKS。	2021 年 6 月 25 日
新指南	新增下列變更： • 已建立 Amazon MWAA 上 Apache Airflow 的效能調校。	2021 年 6 月 25 日

新的主題和使用案例	新增下列變更：	2021 年 6 月 18 日

- 在 將 `connections add`和 `connections delete`新增至支援的 Apache Airflow v2.0.2 CLI 命令Apache Airflow CLI 命令參考。
- 新增 中可用的最新版本 CloudFormation 是位於 的 Apache Airflow 2.0.2 版Amazon Managed Workflows for Apache Airflow 的快速入門教學課程 。
- 新增將臨時資料儲存在 Apache Airflow 工作者上的問題至 Amazon MWAA 常見問答集。
- 新增 'Executor 將任務執行個體 %s 已完成' 錯誤的主題報告至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。
- 已新增「伺服器意外關閉連線」日誌至 的主題對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。
- 新增對堡壘主機在 SSH 通道上執行 CLI 命令的範例。 建立 Apache Airflow CLI 字符
- 已將隨機產生的使用者名稱主題新增至 對 Amazon

Managed Workflows for Apache Airflow 進行故障診斷。

- 在 CLI 中執行 DAG 時，新增 503 錯誤的主題至對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。
- 新增 Apache Airflow v2.0.2 中自訂外掛程式的主題，該主題需要 的 Airflow 組態選項，`core.lazy_load_plugins : False`才能在每個 Airflow 程序開始時載入外掛程式，以覆寫版本的預設設定為在 Amazon MWAA 上使用 Apache Airflow 組態選項。
- 在 新增 Apache Airflow v2.0.2 外掛程式範例程式碼的 Airflow 組態選項步驟使用 Apache Hive 和 Hadoop 建立自訂外掛程式。
- 新增 Apache Airflow v2.0.2 外掛程式範例程式碼的 Airflow 組態選項步驟。
- 在 新增 Apache Airflow v2.0.2 外掛程式範例程式碼的 Airflow 組態選項步驟為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式。
- 在 新增 Apache Airflow v2.0.2 外掛程式範例程式碼

	的 Airflow 組態選項步驟使用 Oracle 建立自訂外掛程式。	
新的範本程式碼	新增下列變更： • 在 新增 Apache Airflow Snowflake 連線的範例程式碼在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow Snowflake 連線。	2021 年 6 月 18 日

新的主題和使用案例	新增下列變更：	2021 年 6 月 2 日

- 已將伺服器端加密指引新增至 為 Amazon MWAA 建立 Amazon S3 儲存貯體。
- 已將 Apache Airflow v2.0.2 的秘密後端新增至 使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。
- 新增 Apache Airflow 工作者配額增加對 請求的問題Amazon MWAA 常見問答集。
- 新增了哪些指標用於決定是否將 Apache Airflow 工作者擴展到 的問題Amazon MWAA 常見問答集。
- 新增在 CloudWatch 中建立自訂指標至 的問題Amazon MWAA 常見問答集。
- 新增在 中為具有私有路由的 VPC 啟用 Amazon S3 VPC 介面端點私有 IP 地址的步驟在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點。
- 新增在 中使用本機連接埠轉送設定 SSH 通道的選項教學課程：使用 Linux 堡壘主機設定私有網路存取。

新的範本程式碼	新增下列變更： • 新增 DAG 的範例程式碼，查詢 Amazon Aurora PostgreSQL 中繼資料資料庫，並將自訂指標發佈至 Amazon CloudWatch使用 DAG 在 CloudWatch 中寫入自訂指標。	2021 年 6 月 2 日
新指南	新增下列變更： • 已建立指南，說明如何在 的 Apache Airflow UI 中交替使用連線範本連線類型概觀。	2021 年 6 月 2 日
修正項目	新增下列變更： • 在選項三：建立沒有網際網路存取 的 VPC 網路中，將 Apache Airflow VPC 端點新增至 CloudFormation 範本建立 VPC 網路。	2021 年 6 月 2 日

Apache Airflow 2.0.2 版啟動	Apache Airflow v2.0.2 的一般可用性啟動。 • 已建立 Amazon Managed Workflows for Apache Airflow 的 Apache Airflow 版本。 • 已建立 CloudWatch 中的 Apache Airflow 環境指標。 • 已將 Apache Airflow v2.0.2 的版本特定連結新增至 在 Amazon MWAA 上使用 Apache Airflow 組態選項。 • 已將 Apache Airflow 2.0.2 版的特定指南新增至 安裝 Python 相依性。 • 已將 Apache Airflow 2.0.2 版的特定版本指引新增至 在 requirements.txt 中管理 Python 相依性。 • 已將 Apache Airflow 2.0.2 版範例外掛程式新增至 安裝自訂外掛程式。 • 已將 Apache Airflow 2.0.2 版範本程式碼新增至 Amazon MWAA 環境上的 Aurora PostgreSQL 資料庫清除。 • 已將 Apache Airflow 2.0.2 版範本程式碼新增至 在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線。 • 已將 Apache Airflow 2.0.2 版範本程式碼新增至 為 Apache Airflow PythonVir	2021 年 5 月 26 日

	tualenvOperator 建立自訂外掛程式。 • 已將 Apache Airflow v2.0.2 命令新增至 Apache Airflow CLI 命令參考。 • 已將 Apache Airflow 2.0.2 版指令碼新增至 建立 Apache Airflow CLI 字符。 • 已新增 Amazon MWAA 預設使用最新 Apache Airflow 版本的備註建立 Amazon MWAA 環境。	
新的主題和使用案例	新增下列變更： • 新增了對 停滯或未執行的 Airflow 任務進行故障診斷的指引對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 5 月 14 日
修正項目	新增下列變更： • 我們已更新範例外掛程式程式碼，以在 中使用最新的 Java 版本使用 Apache Hive 和 Hadoop 建立自訂外掛程式。先前，它是 `os.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"` 。	2021 年 5 月 12 日

移除或移動主題	新增下列變更： • 依類別對 Amazon Managed Workflows for Apache Airflow 進行故障診斷將 中的主題移至新頁面。	2021 年 5 月 10 日
新的主題和使用案例	新增下列變更： • 已將 Amazon S3 儲存貯體概觀新增至 在 Amazon MWAA 上使用 DAG。	2021 年 5 月 10 日
移除或移動主題	新增下列變更： • 存取 Apache Airflow 移至主層級導覽，並新增 建立 Apache Airflow Web 伺服器存取字符、 建立 Apache Airflow CLI 字符和 的頁面Apache Airflow CLI 命令參考。	2021 年 5 月 7 日

新的主題和使用案例	新增下列變更： • 針對 中所有支援和不支援的 Airflow CLI 命令，新增 Apache Airflow 參考指南的版本特定連結Apache Airflow CLI 命令參考。 • 新增 Apache Airflow 參考指南中所有組態選項的版本特定連結在 Amazon MWAA 上使用 Apache Airflow 組態選項。 • 已將 Amazon MWAA CLI 公用程式新增至 在 requirements.txt 中管理 Python 相依性。	2021 年 5 月 7 日

新主題和使用案例	新增下列變更： • 新增如何在 中建構 plugins.zip 的平面和巢狀範例安裝自訂外掛程式。 • 已將 Amazon MWAA CLI 公用程式新增至 新增或更新 DAGs、 安裝自訂外掛程式和 安裝 Python 相依性頁面。 • 根據 、 和 安裝 Python 相依性 頁面中的使用者意見回饋，將內容重組為概觀、上傳至 Amazon S3安裝自訂外掛程式，以及在 Amazon MWAA 區段上安裝 。 • 新增範例使用案例，以建立必要的 VPC 端點，並將其連接到 中沒有網際網路存取權的現有 Amazon VPC關於 Amazon MWAA 上的聯網。	2021 年 4 月 30 日
新的範本程式碼	新增下列變更： • 新增在 Secrets Manager 中使用秘密金鑰的範例程式碼，用於 中的 Apache Airflow 變數針對 AWS Secrets Manager Apache Airflow 變數在 中使用私密金鑰。	2021 年 4 月 30 日
新指南	新增下列變更： • 已建立 在具有私有路由的 Amazon VPC 中建立所需的 VPC 服務端點。	2021 年 4 月 30 日

修正項目	新增下列變更： • 我們已在 `webserver.default_ui_timezone` 中將 更新`core.default_ui_timezone` 為在 Amazon MWAA 上使用 Apache Airflow 組態選項。	2021 年 4 月 30 日
新的主題和使用案例	新增下列變更： • 已將 SSH 通道的 Windows (PuTTY) 步驟新增至 教學課程：使用 Linux 堡壘主機設定私有網路存取。 • 已新增 的主題`apache-airflow-providers-amazon` ，僅與 Apache Airflow 2.0 相容至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 4 月 23 日
新的範本程式碼	新增下列變更： • 新增範例程式碼，在 Secrets Manager 中使用秘密金鑰，以在 中使用 Apache Airflow 連線在 中使用私密金鑰 AWS Secrets Manager 進行 Apache Airflow 連線。	2021 年 4 月 23 日

新指南	新增下列變更：	2021 年 4 月 23 日

- 已建立 關於 Amazon MWAA 上的聯網。
- 已建立 Amazon MWAA 上 VPC 的安全性。
- 已建立 在 Amazon MWAA 上管理對服務特定 Amazon VPC 端點的存取。

新的主題和使用案例	新增下列變更：	2021 年 4 月 16 日

- 新增 CloudFormation 範本，以在 中建立沒有網際網路存取權的 Amazon VPC 網路建立 VPC 網路。
- 新增了在 AWS Client VPN 中建立 的新教學課程教學課程：使用 設定私有網路存取 AWS Client VPN。
- 根據使用者意見回饋和 中的簡化文件，將網路存取頁面的名稱變更為 Apache Airflow 存取模式Apache Airflow 存取模式。
- 根據 中的使用者意見回饋，簡化文件僅包含 Amazon VPC 入門資訊和範本建立 VPC 網路。
- 已將 BigQuery 運算子解決方法新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。
- 已將 Apache Airflow v1.10.12 限制條件檔案最佳實務新增至 安裝 Python 相依性。

新的範例程式碼	新增下列變更： • 新增範例程式碼，以在 中使用 Oracle 建立自訂外掛程式使用 Oracle 建立自訂外掛程式。 • 新增範例程式碼以建立可產生執行時間環境變數的自訂外掛程式。 •	2021 年 4 月 16 日
新的主題和使用案例	新增下列變更： • 已將 VPC 安全群組上自我參考規則要求的主題新增至 Amazon MWAA 常見問答集。 • 已將自訂外掛程式目錄和大小限制新增至 安裝自訂外掛程式。 • 已將需求目錄和大小限制新增至 安裝 Python 相依性。 • 釐清 中 `foo.user`和 的 Apache Airflow `foo.pass` 組態選項在 requirements.txt 中管理 Python 相依性。 • 已將組態選項概觀新增至在 Amazon MWAA 上使用 Apache Airflow 組態選項。	2021 年 4 月 9 日

新的範本程式碼	新增下列變更： • 新增範例程式碼，以在 PythonVirtualenvOperator 中使用 建立自訂外掛程式為 Apache Airflow PythonVirtualenvOperator 建立自訂外掛程式。 • 新增範例程式碼，以在 中使用 Apache Hive 和 Hadoop 建立自訂外掛程式使用 Apache Hive 和 Hadoop 建立自訂外掛程式。	2021 年 4 月 9 日
修正項目	新增下列變更： • 我們已更新 requirements.txt 的格式，並在 中新增與 Apache Airflow v1.10.12 相容的範例安裝 Python 相依性。	2021 年 3 月 31 日
新的主題和使用案例	新增下列變更： • 新增了將 requirements.txt 或 plugins.zip 移除到 的解決方法Amazon MWAA 常見問答集。 • 已將環境中 SSH 的 bash 因應措施新增至 Amazon MWAA 常見問答集。 • 已將 CloudTrail ResourceAlreadyExistsException 錯誤的主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 3 月 26 日

新的主題和使用案例	新增下列變更：	2021 年 3 月 19 日

- 新增用於 AWS 的服務清單Amazon MWAA 執行角色。
- 新增用於 AWS 的服務清單Amazon MWAA 的服務連結角色。
- 已將 Amazon MWAA 的 Python 3.7 版本問題新增至 Amazon MWAA 常見問答集。
- 已將 的問題`PythonVirtualenvOperator` 新增至 Amazon MWAA 常見問答集。
- 新增故障診斷指令碼，做為 中與 VPC 和環境組態相關的所有主題的後續步驟對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。
- 釐清文件，Linux 堡壘必須與環境位於相同的區域教學課程：使用 Linux 堡壘主機設定私有網路存取。

新指南	新增下列變更： • 在 建立適用於 AWS Secrets Manager 的 Apache Airflow 連線指南使用 AWS Secrets Manager 秘密設定 Apache Airflow 連線。 • 使用 CloudFormation 範本建立快速入門教學課程，以在 建立 Amazon VPC 基礎設施、Amazon S3 儲存貯體和 Amazon MWAA 環境Amazon Managed Workflows for Apache Airflow 的快速入門教學課程。	2021 年 3 月 19 日
新的主題和使用案例	新增下列變更： • 新增建立 Amazon S3 儲存貯體疑難排解主題 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 新增建立 JSON 政策並將其連接至 的步驟Amazon MWAA 執行角色。	2021 年 3 月 12 日
新的範本程式碼	新增下列變更： • 新增範例程式碼，以在觸發 DAG 至 時新增組態存取 Apache Airflow。	2021 年 3 月 12 日

新指南	新增下列變更： • 在 建立最佳實務指南在 requirements.txt 中管理 Python 相依性。	2021 年 3 月 12 日
新的主題和使用案例	新增下列變更： • 已將 Google/GCP/BigQuery 疑難排解主題新增至對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 Cython 疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 MySQL 疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 5xx Web 伺服器錯誤疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 3 月 5 日

現在支援	新增下列變更： • 先前，backend_kwargs 不支援 ，AWS Secrets Manager 您需要解決方法來覆寫 Secrets Manager 函數呼叫。現在支援 backend_kwargs 。請參閱 中的 AWS Secrets Manager 疑難排解主題對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 3 月 4 日
修正項目	新增下列變更： • 我們已更新每個環境類別的大小，以反映 中的實際 GB設定 Amazon MWAA 環境類別。	2021 年 3 月 4 日
新的主題和使用案例	新增下列變更： • 使用 VPC 端點政策將私有網路存取新增至 Apache Airflow 存取模式。 • 已將建立環境故障診斷主題的其他檢查新增至對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 新增將 的日誌存取requirements.txt 至的步驟安裝 Python 相依性。	2021 年 2 月 26 日

新的主題和使用案例	新增下列變更： • 已將 Apache Hive 使用案例新增至 安裝 Python 相依性。 • 釐清文件，Apache Airflow 套件所需的相依性需要包含在 的 `requirements.txt` 檔案中安裝 Python 相依性。 • 已將更新 requirements.txt 疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。	2021 年 2 月 25 日
新的教學課程	新增下列變更： • 已將私有網路教學課程新增至 教學課程：使用 Linux 堡壘主機設定私有網路存取。	2021 年 2 月 22 日
新的主題和使用案例	新增下列變更： • 已將私有和公有網路組態新增至 Apache Airflow 存取模式。 • 已將開發群組使用案例和使用者案例新增至 Amazon MWAA 執行角色。	2021 年 2 月 22 日

新的範本程式碼	新增下列變更： • 已將 Web 登入字符和 CLI 字符的範例 Python 指令碼新增至 存取 Apache Airflow。 • 新增在另一個環境中觸發 DAG 的範例程式碼至 Amazon Managed Workflows for Apache Airflow 的程式碼範例。 • 新增使用 Lambda 函數觸發 DAG 的範例程式碼至 使用 Lambda 函數叫用 DAGs。	2021 年 2 月 22 日
新的命令和程序	新增下列變更： • 已將逐步程序新增至 的所有指令碼存取 Apache Airflow。	2021 年 2 月 22 日
新的範本程式碼	新增下列變更： • 在 更新 Web 登入字符的 curl 範例存取 Apache Airflow。 • 新增將 Amazon RDS Microsoft SQL Server 連線至 的範例程式碼搭配使用 Amazon MWAA 與 Amazon RDS for Microsoft SQL Server。	2021 年 2 月 17 日

新的命令和程序	新增下列變更： • 已將 AWS CLI 命令新增至在 Amazon MWAA 上使用 DAG頁面。 • Apache Airflow 不支援 CLI 命令中的序列化 DAGs。由於 CLI 在基於安全考量而沒有外掛程式或需求的 Web 伺服器上執行，因此任何具有 plugins.zip 或 requirements.txt 的 MWAA 環境都不支援這些命令。將 Apache Airflow `list_dags` 和`backfill`命令移至 不支援的命令存取 Apache Airflow。	2021 年 2 月 17 日
GitHub 啟動	使用者指南文件現在是 GitHub 上的開放原始碼。在任何頁面的 GitHub 上選擇編輯此頁面。	2021 年 2 月 17 日

新的主題和使用案例	新增下列變更： • 已將 Step Functions v. Amazon MWAA 使用案例的問題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 CLI 存取政策新增至 存取 Amazon MWAA 環境。 • 釐清可在 指定任何支援 Apache Airflow 組態選項的文件在 Amazon MWAA 上使用 Apache Airflow 組態選項。 • 釐清文件：如果一個可用區域中的 Fargate 容器失敗，MWAA 會切換到位於 的不同可用區域中的另一個容器建立 VPC 網路。	2021 年 2 月 12 日
新的主題和使用案例	新增下列變更： • 新增了 設定 Amazon MWAA 環境類別。	2021 年 2 月 5 日
移除或移動主題	新增下列變更： • 已移除從 開始對 Amazon S3 儲存貯`airflow-`體名稱的需求開始使用 Amazon Managed Workflows for Apache Airflow。 • 已將 存取 Amazon MWAA 環境和 Amazon MWAA 執行角色移至 管理對 Amazon MWAA 環境的存取。	2021 年 2 月 4 日

Amazon MWAA CloudFormation	更新參數以在 Amazon MWAA CloudFormation 建立環境。 • 移除 SubnetList。 • 移除 TagList。 • 新增 NetworkConfiguration。 • 新增 TagMap。 • 新增建立環境請求範例。	2021 年 2 月 4 日
新的主題和使用案例	新增下列變更： • 已將電子郵件組態範例新增至 在 Amazon MWAA 上使用 Apache Airflow 組態選項。 • 已將 PostgresHook 疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將 AWS Secrets Manager 疑難排解主題新增至 對 Amazon Managed Workflows for Apache Airflow 進行故障診斷。 • 已將高效能使用案例新增至 設定 Amazon MWAA 工作者自動擴展。	2021 年 1 月 29 日
Amazon MWAA 啟動	Amazon Managed Workflows for Apache Airflow 的一般可用性啟動。 • 使用者指南文件 • CloudFormation 文件	2020 年 11 月 24 日

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。