aws

遷移指南

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: 遷移指南

Table of Contents

什麼是遷移指南？ 1
網路架構 2
Amazon MWAA 元件 2
連線能力 3
關鍵考量 4
身分驗證 4
執行角色 4
遷移至新的 Amazon MWAA 環境 6
先決條件 6
步驟一：建立新環境 6
步驟二：遷移工作流程資源 13
步驟三：匯出中繼資料 14
步驟四：匯入中繼資料 16
後續步驟 18
將工作負載從 遷移 AWS Data Pipeline 至 Amazon MWAA 19
選擇 Amazon MWAA 19
架構和概念映射 20
實作範例 21
定價比較 22
相關資源 22
文件歷史記錄 23
..... xxiv

什麼是 Amazon MWAA 遷移指南？

Amazon Managed Workflows for Apache Airflow 是 Apache Airflow 的受管協同運作服務，可讓您大規模操作雲端中的資料管道。Amazon MWAA 會管理 Apache Airflow 的佈建和持續維護，因此您不再需要擔心修補、擴展或保護執行個體。

Amazon MWAA 會自動擴展執行任務的運算資源，以隨需提供一致的效能。Amazon MWAA 預設會保護您的資料。您的工作負載會使用 Amazon Virtual Private Cloud 在您自己的隔離安全雲端環境中執行。這可確保使用 自動加密資料 AWS Key Management Service。

使用本指南將自我管理的 Apache Airflow 工作流程遷移至 Amazon MWAA，或將現有的 Amazon MWAA 環境升級至新的 Apache Airflow 版本。遷移教學說明如何建立或複製新的 Amazon MWAA 環境、遷移工作流程資源，以及將工作流程中繼資料和日誌傳輸至新的環境。

在您嘗試遷移教學課程之前，建議您檢閱下列主題。

- 網路架構
- 關鍵考量

探索 Amazon MWAA 網路架構

下一節說明組成 Amazon MWAA 環境的主要元件，以及每個環境整合的一組 AWS 服務，用於管理其資源、確保資料安全，並提供工作流程的監控和可見性。

主題

- Amazon MWAA 元件
- 連線能力

Amazon MWAA 元件

Amazon MWAA 環境包含下列四個主要元件：

1. 排程器 — 剖析和監控所有 DAGs，並在符合 DAG 相依性時佇列任務以執行。Amazon MWAA 會將排程器部署為具有至少 2 個排程器的 AWS Fargate 叢集。視您的工作負載而定，您可以將排程器計數增加到五個。如需 Amazon MWAA 環境類別的詳細資訊，請參閱 Amazon MWAA 環境類別。
2. 工作者 — 執行排程任務的一或多個 Fargate 任務。您環境的工作者數量取決於您指定的最小和最大數量之間的範圍。當佇列和執行中的任務數量超過現有工作者可以處理的數量時，Amazon MWAA 會開始自動調整工作者規模。執行 和排入佇列的任務總和為零超過兩分鐘時，Amazon MWAA 會將工作者數量縮減到最低。如需 Amazon MWAA 如何處理自動調整規模工作者的詳細資訊，請參閱 Amazon MWAA 自動調整規模。
3. Web 伺服器 — 執行 Apache Airflow Web UI。您可以使用私有或公有網路存取來設定 Web 伺服器。在這兩種情況下，對 Apache Airflow 使用者的存取是由您在 AWS Identity and Access Management (IAM) 中定義的存取控制政策所控制。如需為您的環境設定 IAM 存取政策的詳細資訊，請參閱存取 Amazon MWAA 環境。
4. 資料庫 — 儲存有關 Apache Airflow 環境和工作流程的中繼資料，包括 DAG 執行歷史記錄。資料庫是由 管理的單一租用戶 Aurora PostgreSQL 資料庫 AWS，可透過私有安全的 Amazon VPC 端點存取排程器和工作者 Fargate 容器。

每個 Amazon MWAA 環境也會與一組 AWS 服務互動，以處理各種任務，包括儲存和存取 DAGs 和任務相依性、保護靜態資料，以及記錄和監控您的環境。下圖示範 Amazon MWAA 環境的不同元件。

> **Note**
>
> 服務 Amazon VPC 不是共用的 VPC。Amazon MWAA 會為您建立的每個環境建立 AWS 擁有的 VPC。

- Amazon S3 — Amazon MWAA 會將 DAGs、需求和外掛程式檔案等所有工作流程資源存放在 Amazon S3 儲存貯體中。如需有關建立儲存貯體做為環境建立的一部分，以及上傳 Amazon MWAA 資源的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的為 Amazon MWAA 建立 Amazon S3 儲存貯體。
- Amazon SQS — Amazon MWAA 使用 Amazon SQS 將工作流程任務排入 Celery 執行器的佇列。
- Amazon ECR — Amazon ECR 託管所有 Apache Airflow 映像。Amazon MWAA 僅支援 AWS受管 Apache Airflow 映像。
- AWS KMS — Amazon MWAA 會使用 AWS KMS 來確保您的靜態資料安全。根據預設，Amazon MWAA 使用 AWS受管 AWS KMS 金鑰，但您可以將環境設定為使用自己的客戶受管 AWS KMS 金鑰。如需使用自有客戶受管 AWS KMS 金鑰的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的資料加密的客戶受管金鑰。
- CloudWatch — Amazon MWAA 與 CloudWatch 整合，並將 Apache Airflow 日誌和環境指標交付至 CloudWatch，讓您監控 Amazon MWAA 資源並疑難排解問題。

連線能力

您的 Amazon MWAA 環境需要存取其整合的所有 AWS 服務。Amazon MWAA 執行角色會控制如何授予 Amazon MWAA 存取權，以代表您連線到其他 AWS 服務。對於網路連線，您可以提供公有網際網路存取 Amazon VPC 或建立 Amazon VPC 端點。如需為您的環境設定 Amazon VPC 端點 (AWS PrivateLink) 的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的管理對 Amazon MWAA 上 VPC 端點的存取。

Amazon MWAA 會在排程器和工作者上安裝需求。如果您的需求來自公有 PyPi 儲存庫，您的環境需要連線至網際網路，才能下載所需的程式庫。對於私有環境，您可以使用私有 PyPi 儲存庫，或將檔案庫綁定 `.whl` 為環境的自訂外掛程式。

當您在私有模式中設定 Apache Airflow 時，Apache Airflow UI 只能透過 Amazon VPC 端點存取您的 Amazon VPC。

如需聯網的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的聯網。

遷移至新 MWAA 環境的重要考量事項

在您計劃將 Apache Airflow 工作負載遷移至 Amazon MWAA 時，進一步了解關鍵考量事項，例如身分驗證和 Amazon MWAA 執行角色。

主題

- 身分驗證
- 執行角色

身分驗證

Amazon MWAA 使用 AWS Identity and Access Management (IAM) 控制對 Apache Airflow UI 的存取。您必須建立和管理授予 Apache Airflow 使用者存取 Web 伺服器和管理 DAGs IAM 政策。您可以使用不同帳戶的 IAM 來管理 Apache Airflow 預設角色的身分驗證和授權。

您可以建立自訂 Airflow 角色並將其映射至 IAM 主體，進一步管理和限制 Apache Airflow 使用者只能存取工作流程 DAGs 的一部分。如需詳細資訊和step-by-step教學課程，請參閱教學課程：限制 Amazon MWAA 使用者對 DAGs 子集的存取。

您也可以設定聯合身分來存取 Amazon MWAA。如需詳細資訊，請參閱下列內容。

- 具有公有存取權的 Amazon MWAA 環境 - 在運算部落格中使用 Okta 做為 Amazon MWAA 的身分提供者。 AWS
- 具有私有存取的 Amazon MWAA 環境 — 使用聯合身分存取私有 Amazon MWAA 環境。

執行角色

Amazon MWAA 使用 執行角色，將許可授予您的環境以存取其他 AWS 服務。您可以將相關許可新增至角色，為工作流程提供 AWS 服務的存取權。如果您在第一次建立環境時選擇預設選項來建立新的執行角色，Amazon MWAA 會將所需的最低許可附加至角色，但 Amazon MWAA 會自動新增所有日誌群組的 CloudWatch Logs 除外。

建立執行角色後，Amazon MWAA 就無法代表您管理其許可政策。若要更新執行角色，您必須編輯政策，以視需要新增和移除許可。例如，您可以將 Amazon MWAA 環境與 整合 AWS Secrets Manager為後端，以安全地存放要在 Apache Airflow 工作流程中使用的秘密和連線字串。若要這麼做，請將下列許可政策連接至您環境的執行角色。

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": "*"
        }
    ]
}
```

與其他 AWS 服務整合遵循類似的模式：您可以將相關許可政策新增至 Amazon MWAA 執行角色，授予 Amazon MWAA 存取服務的許可。如需管理 Amazon MWAA 執行角色的詳細資訊，以及查看其他範例，請參閱《Amazon MWAA 使用者指南》中的 Amazon MWAA 執行角色。

遷移至新的 Amazon MWAA 環境

探索下列步驟，將現有的 Apache Airflow 工作負載遷移至新的 Amazon MWAA 環境。您可以使用這些步驟從舊版 Amazon MWAA 遷移至新版本版本，或將自我管理的 Apache Airflow 部署遷移至 Amazon MWAA。本教學假設您正在從現有的 Apache Airflow v1.10.12 遷移至執行 Apache Airflow v2.5.1 的新 Amazon MWAA，但您可以使用相同的程序從 遷移或遷移至不同的 Apache Airflow 版本。

主題

- 先決條件
- 步驟一：建立執行最新支援 Apache Airflow 版本的新 Amazon MWAA 環境
- 步驟二：遷移工作流程資源
- 步驟三：從現有環境匯出中繼資料
- 步驟四：將中繼資料匯入新環境
- 後續步驟

先決條件

若要能夠完成步驟並遷移您的環境，您需要下列項目：

- Apache Airflow 部署。這可以是自我管理或現有的 Amazon MWAA 環境。
- 您的本機作業系統已安裝 Docker。
- 第 AWS Command Line Interface 2 版已安裝。

步驟一：建立執行最新支援 Apache Airflow 版本的新 Amazon MWAA 環境

您可以使用《Amazon MWAA 使用者指南》中的 Amazon MWAA 入門中的詳細步驟或使用 CloudFormation 範本來建立環境。 如果您要從現有的 Amazon MWAA 環境遷移，並使用 CloudFormation 範本來建立舊環境，您可以變更 `AirflowVersion` 屬性以指定新版本。

```
MwaaEnvironment:
  Type: AWS::MWAA::Environment
  DependsOn: MwaaExecutionPolicy
  Properties:
    Name: !Sub "${AWS::StackName}-MwaaEnvironment"
```

```
    SourceBucketArn: !GetAtt EnvironmentBucket.Arn
    ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
    AirflowVersion: 2.5.1
    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
```

或者，如果從現有的 Amazon MWAA 環境遷移，您可以複製下列 Python 指令碼，該指令碼使用適用於 Python AWS (Boto3) 的 SDK 來複製您的環境。您也可以下載指令碼。

Python 指令碼

```
# This Python file uses the following encoding: utf-8
'''
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: MIT-0

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 __future__ import print_function
import argparse
import json
import socket
import time
import re
import sys
from datetime import timedelta
from datetime import datetime
import boto3
from botocore.exceptions import ClientError, ProfileNotFound
from boto3.session import Session
ENV_NAME = ""
REGION = ""

def verify_boto3(boto3_current_version):
    '''
    check if boto3 version is valid, must be 1.17.80 and up
    return true if all dependenceis are valid, false otherwise
    '''
    valid_starting_version = '1.17.80'
    if boto3_current_version == valid_starting_version:
        return True
    ver1 = boto3_current_version.split('.')
    ver2 = valid_starting_version.split('.')
    for i in range(max(len(ver1), len(ver2))):
        num1 = int(ver1[i]) if i < len(ver1) else 0
        num2 = int(ver2[i]) if i < len(ver2) else 0
        if num1 > num2:
            return True
        elif num1 < num2:
            return False
    return False
```

```
def get_account_id(env_info):
    '''
    Given the environment metadata, fetch the account id from the
    environment ARN
    '''
    return env_info['Arn'].split(":")[4]


def validate_envname(env_name):
    '''
    verify environment name doesn't have path to files or unexpected input
    '''
    if re.match(r"^[a-zA-Z][0-9a-zA-Z-_]*$", env_name):
        return env_name
    raise argparse.ArgumentTypeError("%s is an invalid environment name value" %
 env_name)


def validation_region(input_region):
    '''
    verify environment name doesn't have path to files or unexpected input
    REGION: example is us-east-1
    '''
    session = Session()
    mwaa_regions = session.get_available_regions('mwaa')
    if input_region in mwaa_regions:
        return input_region
    raise argparse.ArgumentTypeError("%s is an invalid REGION value" % input_region)


def validation_profile(profile_name):
    '''
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r"^[a-zA-Z0-9]*$", profile_name):
        return profile_name
    raise argparse.ArgumentTypeError("%s is an invalid profile name value" %
 profile_name)

def validation_version(version_name):
    '''
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r"[1-2].\d.\d", version_name):
```

```
        return version_name
    raise argparse.ArgumentTypeError("%s is an invalid version name value" %
 version_name)

def validation_execution_role(execution_role_arn):
    '''
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r'(?i)\b((?:[a-z][\w-]+:(?:/{1,3}|[a-z0-9%])|www\d{0,3}[.]|[a-z0-9.
\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|
(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:\'".,<>?«»“”‘’]))', execution_role_arn):
        return execution_role_arn
    raise argparse.ArgumentTypeError("%s is an invalid execution role ARN" %
 execution_role_arn)

def create_new_env(env):
    '''
    method to duplicate env
    '''
    mwaa = boto3.client('mwaa', region_name=REGION)

    print('Source Environment')
    print(env)
    if (env['AirflowVersion']=="1.10.12") and (VERSION=="2.2.2"):
        if env['AirflowConfigurationOptions']
['secrets.backend']=='airflow.contrib.secrets.aws_secrets_manager.SecretsManagerBackend':
            print('swapping',env['AirflowConfigurationOptions']['secrets.backend'])
            env['AirflowConfigurationOptions']
['secrets.backend']='airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
    env['LoggingConfiguration']['DagProcessingLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['SchedulerLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['TaskLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['WebserverLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['WorkerLogs'].pop('CloudWatchLogGroupArn')
    env['AirflowVersion']=VERSION
    env['ExecutionRoleArn']=EXECUTION_ROLE_ARN
    env['Name']=ENV_NAME_NEW
    env.pop('Arn')
    env.pop('CreatedAt')
    env.pop('LastUpdate')
    env.pop('ServiceRoleArn')
    env.pop('Status')
    env.pop('WebserverUrl')
    if not env['Tags']:
```

```
        env.pop('Tags')
    print('Destination Environment')
    print(env)

    return mwaa.create_environment(**env)

def get_mwaa_env(input_env_name):

    # https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/
mwaa.html#MWAA.Client.get_environment
    mwaa = boto3.client('mwaa', region_name=REGION)
    environment = mwaa.get_environment(
        Name=input_env_name
    )['Environment']

    return environment

def print_err_msg(c_err):
    '''short method to handle printing an error message if there is one'''
    print('Error Message: {}'.format(c_err.response['Error']['Message']))
    print('Request ID: {}'.format(c_err.response['ResponseMetadata']['RequestId']))
    print('Http code: {}'.format(c_err.response['ResponseMetadata']['HTTPStatusCode']))

#
# Main
#
# Usage:
# python3 clone_environment.py --envname MySourceEnv --envnamenew MyDestEnv --region
 us-west-2 --execution_role AmazonMWAA-MyDestEnv-ExecutionRole --version 2.2.2
#
# based on https://github.com/awslabs/aws-support-tools/blob/master/MWAA/verify_env/
verify_env.py
#

if __name__ == '__main__':
    if sys.version_info[0] < 3:
        print("python2 detected, please use python3. Will try to run anyway")
    if not verify_boto3(boto3.__version__):
        print("boto3 version ", boto3.__version__, "is not valid for this script. Need
 1.17.80 or higher")
        print("please run pip install boto3 --upgrade --user")
        sys.exit(1)
    parser = argparse.ArgumentParser()
```

```
   parser.add_argument('--envname', type=validate_envname, required=True, help="name
of the source MWAA environment")
   parser.add_argument('--region', type=validation_region,
default=boto3.session.Session().region_name,
                       required=False, help="region, Ex: us-east-1")
   parser.add_argument('--profile', type=validation_profile, default=None,
                       required=False, help="AWS CLI profile, Ex: dev")
   parser.add_argument('--version', type=validation_version, default="2.2.2",
                       required=False, help="Airflow destination version, Ex: 2.2.2")
   parser.add_argument('--execution_role', type=validation_execution_role,
default=None,
                       required=True, help="New environment execution role ARN, Ex:
arn:aws:iam::112233445566:role/service-role/AmazonMWAA-MyEnvironment-ExecutionRole")
   parser.add_argument('--envnamenew', type=validate_envname, required=True,
help="name of the destination MWAA environment")

   args, _ = parser.parse_known_args()
   ENV_NAME = args.envname
   REGION = args.region
   PROFILE = args.profile
   VERSION = args.version
   EXECUTION_ROLE_ARN = args.execution_role
   ENV_NAME_NEW = args.envnamenew

   try:
       print("PROFILE",PROFILE)
       if PROFILE:
           boto3.setup_default_session(profile_name=PROFILE)
       env = get_mwaa_env(ENV_NAME)
       response = create_new_env(env)
       print(response)
   except ClientError as client_error:
       if client_error.response['Error']['Code'] == 'LimitExceededException':
           print_err_msg(client_error)
           print('please retry the script')
       elif client_error.response['Error']['Code'] in ['AccessDeniedException',
'NotAuthorized']:
           print_err_msg(client_error)
           print('please verify permissions used have permissions documented in
readme')
       elif client_error.response['Error']['Code'] == 'InternalFailure':
           print_err_msg(client_error)
           print('please retry the script')
       else:
```

```
            print_err_msg(client_error)
    except ProfileNotFound as profile_not_found:
        print('profile', PROFILE, 'does not exist; check the profile name')
    except IndexError as error:
        print("Error:", error)
```

步驟二：遷移工作流程資源

Apache Airflow v2 是主要版本版本。如果您要從 Apache Airflow v1 遷移，您必須準備工作流程資源，並驗證您對 DAGs、需求和外掛程式所做的變更。若要這麼做，建議您使用 Docker 和 Amazon MWAA 本機執行器，在本機作業系統上設定 Apache Airflow 的橋接版本。Amazon MWAA 本機執行器提供命令列界面 (CLI) 公用程式，可在本機複寫 Amazon MWAA 環境。

每當您變更 Apache Airflow 版本時，請務必在 中參考正確的 `--constraint` URL`requirements.txt`。

遷移您的工作流程資源

1. 建立 aws-mwaa-local-runner 儲存庫的分支，並複製 Amazon MWAA 本機執行器的副本。
2. 檢查 aws-mwaa-local-runner 儲存庫的`v1.10.15`分支。Apache Airflow 發行了 v1.10.15 作為橋接器版本，以協助遷移至 Apache Airflow v2，雖然 Amazon MWAA 不支援 v1.10.15，但您可以使用 Amazon MWAA 本機執行器來測試您的資源。
3. 使用 Amazon MWAA 本機執行器 CLI 工具來建置 Docker 映像，並在本機執行 Apache Airflow。如需詳細資訊，請參閱 GitHub 儲存庫中的本機執行器 README。
4. 使用在本機執行的 Apache Airflow，請遵循 Apache Airflow 文件網站中從 1.10 升級到 2 中所述的步驟。

    a. 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。

    b. 如果您已將自訂運算子和感應器與現有 Apache Airflow v1.10.12 環境的外掛程式綁定在一起，請將它們移至 DAG 資料夾。如需 Apache Airflow v2+ 模組管理最佳實務的詳細資訊，請參閱 Apache Airflow 文件網站上的模組管理。
5. 對工作流程資源進行必要的變更後，請檢查 aws-mwaa-local-runner 儲存庫的`v2.5.1`分支，並在本機測試更新的工作流程 DAGs、需求和自訂外掛程式。如果您要遷移至不同的 Apache Airflow 版本，您可以改為為您的版本使用適當的本機執行器分支。

6. 成功測試工作流程資源後，請將 DAGs、 `requirements.txt`和 外掛程式複製到您使用新 Amazon MWAA 環境設定的 Amazon S3 儲存貯體。

步驟三：從現有環境匯出中繼資料

當您將更新的 DAG 檔案複製到您環境的 Amazon S3 儲存貯體，且排程器剖析它們時`dag`，Apache Airflow 中繼資料表，例如 `dag_tag`、 和 `dag_code`會自動填入。許可相關資料表也會根據您的 IAM 執行角色許可自動填入。您不需要遷移它們。

您可以視需要遷移與 DAG 歷史記錄、、`sla_miss`、 `variable` `slot_pool`和 、 `xcom` `job`和 `log`資料表相關的資料。任務執行個體日誌存放在`airflow-`*`{environment_name}`*日誌群組下的 CloudWatch Logs 中。如果您想要查看較舊執行的任務執行個體日誌，則必須將這些日誌複製到新的環境日誌群組。我們建議您只移動幾天的日誌，以減少相關聯的成本。

如果您要從現有的 Amazon MWAA 環境遷移，則無法直接存取中繼資料資料庫。您必須執行 DAG，將中繼資料從現有的 Amazon MWAA 環境匯出至您選擇的 Amazon S3 儲存貯體。如果您要從自我管理環境遷移，也可以使用下列步驟匯出 Apache Airflow 中繼資料。

匯出資料之後，您可以在新環境中執行 DAG 以匯入資料。在匯出和匯入過程中，所有其他 DAGs都會暫停。

從現有環境匯出中繼資料

1. 使用 建立 Amazon S3 儲存貯體 AWS CLI 來存放匯出的資料。將 `UUID`和 取代`region`為您的資訊。

```
aws s3api create-bucket \
--bucket mwaa-migration-{UUID}\
--region {region}
```

> **Note**
>
> 如果您要遷移敏感資料，例如存放在變數中的連線，建議您為 Amazon S3 儲存貯體啟用預設加密。

2.

> **Note**
>
> 不適用於從自我管理環境遷移。

修改現有環境的執行角色，並新增下列政策，將寫入存取權授予您在步驟 1 中建立的儲存貯體。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*"
            ],
            "Resource": [
                "arn:aws:s3:::mwaa-migration-{UUID}/*"
            ]
        }
    ]
}
```

3. 複製 amazon-mwaa-examples 儲存庫，然後導覽至遷移案例的metadata-migration子目錄。

```
git clone https://github.com/aws-samples/amazon-mwaa-examples.git
cd amazon-mwaa-examples/usecases/metadata-migration/existing-version-new-version/
```

4. 在 中export_data.py，將 的字串值取代S3_BUCKET為您建立來存放匯出中繼資料的 Amazon S3 儲存貯體。

```
S3_BUCKET = 'mwaa-migration-{UUID}'
```

5. 在 metadata-migration目錄中尋找 requirements.txt 檔案。如果您已有現有環境的需求檔案，請將 中指定的其他需求新增至 requirements.txt 檔案。如果您沒有現有的需求檔案，您可以直接使用 metadata-migration目錄中提供的需求檔案。
6. export_data.py 複製到與現有環境相關聯的 Amazon S3 儲存貯體的 DAG 目錄。如果從自我管理環境遷移，export_data.py請複製到您的/dags資料夾。
7. 將已更新的 複製到與現有環境相關聯的 requirements.txt Amazon S3 儲存貯體，然後編輯環境以指定新requirements.txt版本。
8. 環境更新後，請存取 Apache Airflow UI、取消暫停 db_export DAG，並觸發工作流程以執行。

9. 確認中繼資料已匯出至 Amazon S3 儲存貯data/migration/*existing-version*_to_*new-version*/export/體中的 mwaa-migration-*{UUID}* ，且每個資料表都在自己的專用檔案中。

步驟四：將中繼資料匯入新環境

將中繼資料匯入您的新環境

1. 在 中import_data.py，將下列項目的字串值取代為您的資訊。

   - 對於從現有 Amazon MWAA 環境遷移：

     ```
     S3_BUCKET = 'mwaa-migration-{UUID}'
            OLD_ENV_NAME='{old_environment_name}'
            NEW_ENV_NAME='{new_environment_name}'
            TI_LOG_MAX_DAYS = {number_of_days}
     ```

     MAX_DAYS 控制工作流程複製到新環境的日誌檔案天數。

   - 對於從自我管理環境遷移：

     ```
     S3_BUCKET = 'mwaa-migration-{UUID}'
            NEW_ENV_NAME='{new_environment_name}'
     ```

2. （選用）只會import_data.py複製失敗的任務日誌。如果您想要複製所有任務日誌，請修改getDagTasks函數，然後移除 ti.state = 'failed'，如下列程式碼片段所示。

   ```
   def getDagTasks():
        session = settings.Session()
        dagTasks = session.execute(f"select distinct ti.dag_id, ti.task_id,
    date(r.execution_date) as ed \
        from task_instance ti, dag_run r where r.execution_date > current_date -
    {TI_LOG_MAX_DAYS} and \
        ti.dag_id=r.dag_id and ti.run_id = r.run_id order by ti.dag_id,
    date(r.execution_date);").fetchall()
        return dagTasks
   ```

3. 修改新環境的執行角色，並新增下列政策。許可政策允許 Amazon MWAA 從您匯出 Apache Airflow 中繼資料的 Amazon S3 儲存貯體讀取，以及從現有的日誌群組複製任務執行個體日誌。以您的資訊取代所有預留位置。

> **Note**
>
> 如果您要從自我管理環境遷移，您必須從政策中移除 CloudWatch Logs 相關許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
             "arn:aws:logs:us-east-1:111122223333:log-
group:airflow-{old_environment_name}*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::mwaa-migration-{UUID}",
                "arn:aws:s3:::mwaa-migration-{UUID}/*"
            ]
        }
    ]
}
```

4. `import_data.py` 複製到與新環境相關聯的 Amazon S3 儲存貯體的 DAG 目錄，然後存取 Apache Airflow UI 以取消暫停 `db_import` DAG 並觸發工作流程。新的 DAG 會在幾分鐘內出現在 Apache Airflow UI 中。
5. DAG 執行完成後，透過存取每個個別 DAG 來驗證您的 DAG 執行歷史記錄是否已複製。

後續步驟

- 如需可用 Amazon MWAA 環境類別和功能的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的 Amazon MWAA 環境類別。
- 如需 Amazon MWAA 如何處理自動擴展工作者的詳細資訊，請參閱《Amazon MWAA 使用者指南》中的 Amazon MWAA 自動擴展。
- 如需 Amazon MWAA REST API 的詳細資訊，請參閱 Amazon MWAA REST API。

將工作負載從 遷移 AWS Data Pipeline 至 Amazon MWAA

AWS 已在 2012 年推出 AWS Data Pipeline 此服務。當時，客戶想要一種服務，讓他們使用各種運算選項在不同資料來源之間移動資料。由於資料傳輸需求會隨著時間而變更，因此請針對這些需求提供解決方案。您現在可以選擇最符合您業務需求的解決方案。您可以將工作負載遷移至下列任一 AWS 服務：

- 使用 Amazon Managed Workflows for Apache Airflow (Amazon MWAA) 來管理 Apache Airflow 的工作流程協調。
- 使用 Step Functions 在多個 之間協調工作流程 AWS 服務。
- 使用 AWS Glue 執行和協調 Apache Spark 應用程式。

您選擇的選項取決於您目前的工作負載 AWS Data Pipeline。本主題說明如何從 遷移 AWS Data Pipeline 至 Amazon MWAA。

主題

- 選擇 Amazon MWAA
- 架構和概念映射
- 實作範例
- 定價比較
- 相關資源

選擇 Amazon MWAA

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) 是 Apache Airflow 的受管協同運作服務，可讓您大規模在雲端中設定和操作end-to-end資料管道。Apache Airflow 是一種開放原始碼工具，用於以程式設計方式撰寫、排程和監控稱為工作流程的程序和任務序列。使用 Amazon MWAA，您可以使用 Apache Airflow 和 Python 程式設計語言來建立工作流程，而不必為了可擴展性、可用性和安全性管理基礎基礎設施。Amazon MWAA 會自動擴展其工作流程容量以符合您的需求，並與 AWS 安全服務整合，協助您快速且安全地存取資料。

以下重點介紹從 遷移 AWS Data Pipeline 到 Amazon MWAA 的一些優點：

- 增強的可擴展性和效能 – Amazon MWAA 為定義和執行工作流程提供了靈活且可擴展的架構。這可讓使用者輕鬆處理大型和複雜的工作流程，並利用動態任務排程、資料驅動型工作流程和平行處理等功能。
- 改善監控和記錄 – Amazon MWAA 與 Amazon CloudWatch 整合，以增強對工作流程的監控和記錄。Amazon MWAA 會自動將系統指標和日誌傳送至 CloudWatch。這表示您可以即時追蹤工作流程的進度和效能，並識別出現的任何問題。
- 更好的 AWS 服務與第三方軟體整合 – Amazon MWAA 與各種其他服務整合 AWS ，例如 Amazon S3 AWS Glue和 Amazon Redshift，以及 DBT、Snowflake 和 Databricks 等第三方軟體。這可讓您在不同的環境和服務中處理和傳輸資料。
- 開放原始碼資料管道工具 – Amazon MWAA 會利用您熟悉的相同開放原始碼 Apache Airflow 產品。Apache Airflow 是專為處理資料管道管理各方面而設計的工具，包括擷取、處理、傳輸、完整性測試、品質檢查，以及確保資料歷程。
- 現代且靈活的架構 – Amazon MWAA 利用容器化和雲端原生、無伺服器技術。這意味著要提高靈活性和可攜性，以及更輕鬆地部署和管理工作流程環境。

架構和概念映射

AWS Data Pipeline 和 Amazon MWAA 具有不同的架構和元件，這可能會影響遷移程序和定義和執行工作流程的方式。本節概述兩種服務的架構和元件，並強調一些主要差異。

AWS Data Pipeline 和 Amazon MWAA 都是全受管服務。當您將工作負載遷移至 Amazon MWAA 時，您可能需要學習新概念，以使用 Apache Airflow 建立現有工作流程的模型。不過，您不需要管理基礎設施、修補工作者和管理作業系統更新。

下表將 中的關鍵概念 AWS Data Pipeline 與 Amazon MWAA 中的關鍵概念建立關聯。使用此資訊做為設計遷移計劃的起點。

概念	AWS Data Pipeline	Amazon MWAA
管道定義	AWS Data Pipeline 使用 JSON 型組態檔案來定義工作流程。	Amazon MWAA 使用 Python 導向無環圖 (DAGs) 來定義工作流程。
管道執行環境	工作流程會在 Amazon EC2 執行個體上執行。 會代表您	Amazon MWAA 使用 Amazon ECS 容器化環境來執行任務。

概念	AWS Data Pipeline	Amazon MWAA
	AWS Data Pipeline 佈建和管理這些執行個體。	
管道元件	活動正在處理在工作流程中執行的任務。	Operators (任務) 是工作流程的基本處理單位。
	先決條件包含條件式陳述式，必須先為 true，活動才能執行。	感應器 (任務） 代表條件式陳述式，可在執行之前等待資源或任務完成。
	中的資源 AWS Data Pipeline 是指執行管道活動指定工作的 AWS 運算資源。Amazon EC2 和 Amazon EMR 是兩個可用的資源。	在 DAG 中使用任務，您可以定義各種運算資源，包括 Amazon ECS、Amazon EMR 和 Amazon EKS。Amazon MWAA 會在 Amazon ECS 上執行的工作者上執行 Python 操作。
管道執行	AWS Data Pipeline 支援使用一般速率型和 Cron 型模式來排程執行。	Amazon MWAA 支援使用 Cron 表達式和預設集以及自訂時間表進行排程。
	執行個體是指管道的每個執行。	DAG 執行是指 Apache Airflow 工作流程的每個執行。
	嘗試是指重試失敗的操作。	Amazon MWAA 支援您在 DAG 層級或任務層級定義的重試。

實作範例

在許多情況下，遷移至 Amazon MWAA AWS Data Pipeline 後，您將可以重複使用目前與 協調的資源。下列清單包含使用 Amazon MWAA 進行最常見 AWS Data Pipeline 使用案例的範例實作。

- 執行 Amazon EMR 任務 (AWS workshop)
- 為 Apache Hive 和 Hadoop 建立自訂外掛程式 (Amazon MWAA 使用者指南)

- 將資料從 S3 複製到 Redshift (AWS workshop)
- 在遠端 Amazon ECS 執行個體上執行 shell 指令碼 (Amazon MWAA 使用者指南)
- 協調混合式（內部部署）工作流程（部落格文章）

如需其他教學課程和範例，請參閱下列內容：

- Amazon MWAA 教學課程
- Amazon MWAA 程式碼範例

定價比較

的定價 AWS Data Pipeline 取決於管道數量，以及您在每個管道的使用量。您每天執行超過一次（高頻率）的活動，每個活動每月花費 1 美元。您每天執行一次或更少（低頻率）的活動，每個活動每月花費 0.60 美元。非作用中管道的定價為每個管道 1 美元。如需詳細資訊，請參閱 AWS Data Pipeline 定價頁面。

Amazon MWAA 的定價取決於您的受管 Apache Airflow 環境存在的持續時間，以及提供更多工作者或排程器容量所需的任何其他自動擴展。您每小時支付 Amazon MWAA 環境用量（以一秒解析度計費），費用會因環境大小而異。Amazon MWAA 會根據您的環境組態自動調整工作者數量。 會分別 AWS 計算其他工作者的成本。如需使用各種 Amazon MWAA 環境大小的每小時成本詳細資訊，請參閱 Amazon MWAA 定價頁面。

相關資源

如需使用 Amazon MWAA 的詳細資訊和最佳實務，請參閱下列資源：

- Amazon MWAA API 參考
- 在 Amazon MWAA 上監控儀表板和警示
- Amazon MWAA 上 Apache Airflow 的效能調校

Amazon MWAA 文件歷史記錄

下表說明從 2022 年 3 月開始 Amazon MWAA 遷移指南的重要新增項目。

變更	描述	日期
將工作負載從 遷移 AWS Data Pipeline 至 Amazon MWAA 的新主題	新增將現有工作負載從 遷移 AWS Data Pipeline 至 Amazon MWAA 的新資訊和指導。使用此資訊來協助您設計遷移計劃。 • 將工作負載從 遷移 AWS Data Pipeline 至 Amazon MWAA	2023 年 4 月 14 日
Amazon MWAA 遷移指南啟動	Amazon MWAA 現在提供遷移至新 Amazon MWAA 環境的詳細指引。Amazon MWAA 遷移指南中所述的步驟適用於從現有 Amazon MWAA 環境或從自我管理的 Apache Airflow 部署進行管理。 • 關於 Amazon MWAA 遷移指南	2022 年 3 月 7 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。