aws

移轉指南

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: 移轉指南

Table of Contents

什麼是亞馬遜 MWAA 遷移指南？

Amazon Managed Workflows in Apache Airflow 是一項適用於 Apache Airflow 的受管協同運作服務，可以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的問題。

Amazon MWAA 會自動擴展執行任務的運算資源，以根據需求提供一致的效能。亞馬遜 MWAA 預設會保護您的資料。您的工作負載會使用 Amazon 虛擬私有雲端在您自己獨立且安全的雲端環境中執行。如此可確保資料會使用自動加密AWS Key Management Service。

使用本指南將您的自我管理 Apache 氣流工作流程遷移到亞馬遜 MWAA，或將現有的亞馬遜 MWAA 環境升級到新的 Apache 氣流版本。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到新環境。

在您嘗試移轉自學課程之前，我們建議您先檢閱下列主題。

- 網路架構
- 關鍵考量

亞馬遜 MWAA 網路架構

以下部分說明組成 Amazon MWAA 環境的主要元件，以及每個環境整合的一組AWS服務，以管理其資源、保護資料安全，以及為工作流程提供監控和可見性。

主題

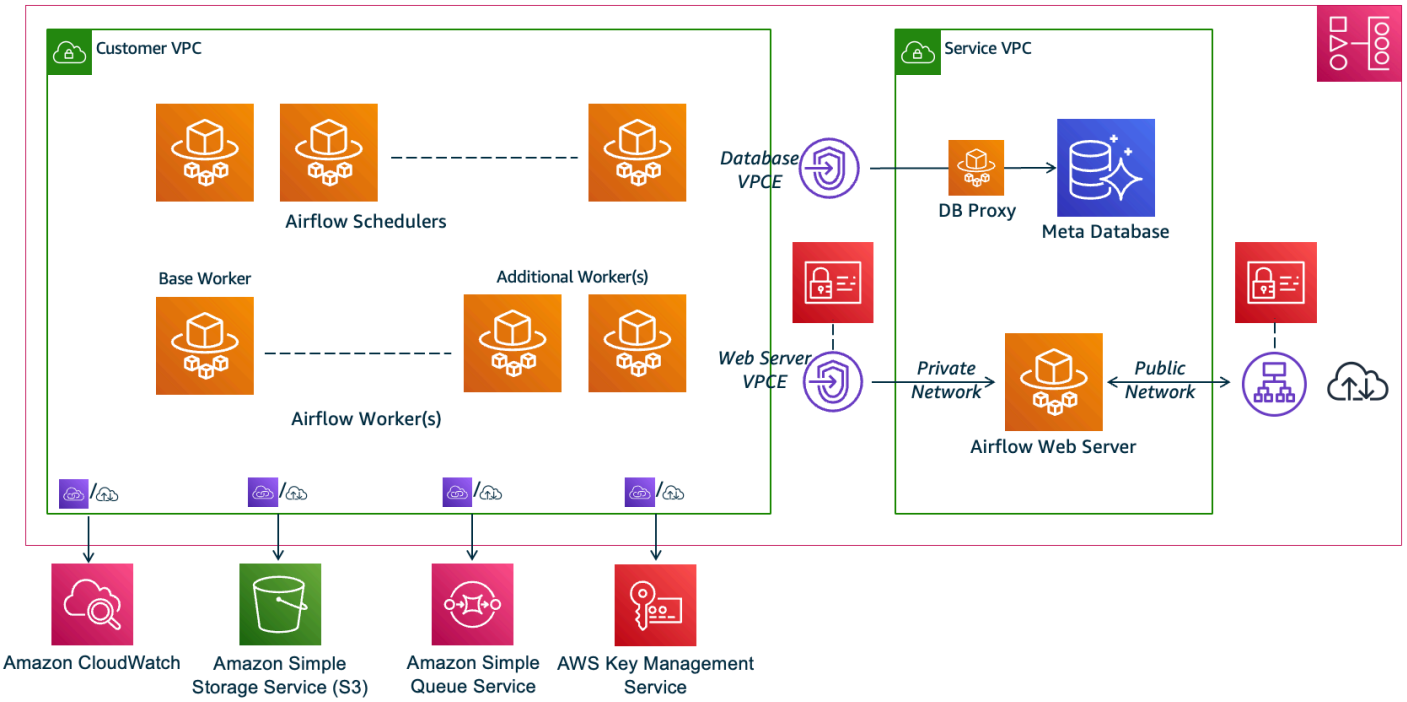
亞馬遜 MWAA 組件

Amazon MWAA 環境包含下列四個主要元件：

1. 排程器 — 剖析並監視所有 DAG，並將工作排入佇列，以便在符合 DAG 的相依性時執行。Amazon MWAA 會將排程器部署為至少有 2 個排程器的AWS Fargate叢集。根據您的工作負載，您最多可以將排程器計數增加到五個。如需有關亞馬遜 MWAA 環境類別的詳細資訊，請參閱亞馬遜 MWAA 環境類別。
2. 工作者 — 執行排程工作的一或多個 Fargate 工作。您環境的 Worker 數量是由您指定的最小數目與上限之間的範圍決定。當排入佇列和執行中的任務數量超過現有工作者所能處理的數量時，Amazon MWAA 會啟 auto-scaling 的工作程式。當執行中和排入佇列的任務總和為零超過兩分鐘時，Amazon MWAA 會將工作者數量調回最小值。如需 Amazon MWAA 如何處理 auto-scaling 工作者的詳細資訊，請參閱 Amazon M WAA 自動擴展。
3. 網頁伺服器 — 執行 Apache 氣流網頁使用者介面。您可以配置具有私人或公共網絡訪問的 Web 服務器。在這兩種情況下，對 Apache Airflow 使用者的存取權限都是由您在 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 Apache Airflow 環境和工作流程的相關中繼資料，包括 DAG 執行歷程記錄。此資料庫是單一租用戶 Aurora PostgreSQL 資料庫AWS，由排程器和工作者的 Fargate 容器所管理，可透過私有保護的 Amazon VPC 端點存取。

每個 Amazon MWAA 環境也會與一組服務互動，以處理各種任AWS務，包括存放和存取 DAG 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。

Amazon MWAA Architecture



> **Note**
>
> 該服務亞馬遜 VPC 不是共享的 VPC。Amazon MWAA 會為您建立的每個環境建立一個AWS擁有的 VPC。

- Amazon S3 — Amazon MWAA 會將您的所有工作流程資源 (例如 DAG、要求和外掛程式檔案) 存放在 Amazon S3 儲存貯體中。如需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 使用者指南中的為亞馬遜 MWAA 建立 Amazon S3 儲存貯體。
- Amazon SQS — 亞馬遜 MWAA 使用 Amazon SQS 將您的工作流程任務與 Celly 執行程序排隊。
- 亞馬遜 ECR — 亞馬遜 ECR 託管所有阿帕奇氣流圖像。亞馬遜 MWAA 僅支援AWS受管的 Apache 氣流影像。
- AWS KMS— Amazon MWAA 用AWS KMS於確保您的資料在靜態時保持安全。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AWS KMS管理金鑰。如需有關使用您自己的客戶管理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 Apache 氣流日誌 CloudWatch 和環境指標整合並提供給您 CloudWatch，讓您能夠監控 Amazon MWAA 資源並對問題進行疑難排解。

連線能力

您的 Amazon MWAA 環境需要存取與之整合的所有AWS服務。Amazon MWAA 執行角色可控制授與 Amazon MWAA 存取權的方式，以代表您連線到其他AWS服務。對於網路連線，您可以提供對 Amazon VPC 的公用網際網路存取權，也可以建立 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 Amazon MWAA 上 VPC 端點的存取權限。

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用PyPi存放庫，則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 存放庫，或將`.whl`檔案中的程式庫組合為您環境的自訂外掛程式。

當您在私有模式中設定 Apache 氣流時，只有透過 Amazon VPC 端點，您的 Amazon VPC 才能存取 Apache 氣流使用者介面。

如需有關聯網路的詳細資訊，請參閱 Amazon MWAA 使用者指南中的聯網。

關鍵考量

移轉到新的 Amazon MWAA 環境之前，請先檢閱下列主題。

主題

身分驗證

Amazon MWAA 使用 AWS Identity and Access Management (IAM) 來控制對 Apache 氣流使用者介面的存取。您必須建立和管理 IAM 政策，以授與 Apache Airflow 使用者存取網頁伺服器和管理 DAG 的權限。您可以使用 IAM 跨不同帳戶管理 Apache Airflow 預設角色的身份驗證和授權。

您可以建立自訂氣流角色並將其對應至 IAM 主體，進一步管理和限制 Apache Airflow 使用者僅存取工作流程 DAG 的子集。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 DAG 子集的存取權。

您也可以設定聯合身分識別以存取 Amazon MWAA。如需詳細資訊，請參閱下列內容：

- 具有公開存取權的 Amazon MWAA 環境 — 透過運算部落格上的 Amazon MWAA 使用 Okta 做為身分提供者。AWS
- 具有私有存取權的 Amazon MWAA 環境 — 使用聯合身分存取私有 Amazon MWAA 環境。

執行角色

Amazon MWAA 使用的執行角色授與您的環境許可以存取其他 AWS 服務。您可以將相關權限新增至角色，為工作流程提供 AWS 服務存取權。如果您選擇在第一次建立環境時建立新的執行角色的預設選項，Amazon MWAA 會將所需的最低許可附加至角色，但 Amazon MWAA 自動新增所有 CloudWatch 日誌群組的日誌除外。

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，您必須編輯原則，視需要新增和移除權限。例如，您可以將 Amazon MWAA 環境與 AWS Secrets Manager做為後端整合，以安全地存放密碼和連接字串，以便在 Apache Airflow 工作流程中使用。若要這麼做，請將下列權限原則附加至您環境的執行角色。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:012345678910:secret:*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        }
    ]
}
```

與其他 AWS 服務整合遵循類似的模式：您可以將相關的許可政策新增至 Amazon MWAA 執行角色，並授予 Amazon MWAA 存取服務的權限。如需管理 Amazon MWAA 執行角色的詳細資訊，以及若要查看其他範例，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

下列主題說明將現有 Apache 氣流工作負載遷移到新的 Amazon MWAA 環境的步驟。您可以使用下列步驟將舊版 Amazon MWAA 遷移到新版本，或將自我管理的 Apache 氣流部署遷移到 Amazon MWAA。本教學假設您要從現有的 Apache 氣流 v1.10.12 遷移到執行 Apache 氣流 v2.5.1 的新亞馬遜 MWAA，但您可以使用相同的程序從不同的 Apache 氣流版本遷移或遷移到不同的 Apache 氣流版本。

主題

先決條件

若要完成步驟並遷移您的環境，您需要以下事項：

- 阿帕奇氣流部署。這可以是自我管理或現有的 Amazon MWAA 環境。
- 為您的本地操作系統安裝的 Docker。
- AWS Command Line Interface版本 2 已安裝。

步驟一：建立執行最新支援 Apache 氣流版本的新亞馬遜 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，或使用範本來建立環境。AWS CloudFormation如果您要從現有的 Amazon MWAA 環境遷移，並使用AWS CloudFormation範本建立舊環境，則可以變更`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，您可以複製下列使用 Python AWSSDK for Python (Boto3) 指令碼來複製您的環境。您也可以下載指令碼。

蟒蛇腳本

```
# 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, please doublecheck the profile
name')
    except IndexError as error:
        print("Error:", error)
```

步驟二：移轉工作流程資源

阿帕奇氣流 V2 是一個主要版本發布。如果您要從 Apache Airflow v1 移轉，則必須準備工作流程資源，並驗證您對 DAG、需求和外掛程式所做的變更。若要這麼做，我們建議您使用 Docker 和 Amazon MWAA 本機執行器，在您的本機作業系統上設定 Apache 氣流的橋接版本。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。

每當您要變更 Apache 氣流版本時，請`--constraint`務必在`requirements.txt`.

欲遷移您的工作流程資源

1. 建立aws-mwaa-local-runner儲存庫的分支，然後複製 Amazon MWAA 本機執行器的副本。
2. 簽出aws-mwaa-local-runner存儲庫的`v1.10.15`分支。阿帕奇氣流發行 v1.10.15 作為橋接版本，以協助遷移到 Apache 氣流 v2，雖然亞馬遜 MWAA 不支持 1.10.15 版，但您可以使用亞馬遜 MWAA 本地運行器來測試您的資源。
3. 使用亞馬遜 MWAA 本地運行器 CLI 工具來構建碼頭映像並在本地運行 Apache 氣流。如需詳細資訊，請參閱GitHub存放庫中的本機執行者 README。
4. 使用在本機執行的 Apache 氣流，請依照 Apache 氣流說明文件網站中的從 1.10 升級到 2 中所述的步驟進行。

    a. 若要更新您的資訊`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 環境的外掛程式搭配在一起，請將它們移至您的 DAG 資料夾。如需 Apache 氣流 v2+ 模組管理最佳作法的詳細資訊，請參閱 Apache 氣流文件網站中的模組管理。

5. 對工作流程資源進行必要的變更後，請簽出aws-mwaa-local-runner存放庫的`v2.5.1`分支，並在本機測試更新的工作流程 DAG、需求和自訂外掛程式。如果您要移轉至不同的 Apache Airflow 版本，您可以改為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 DAG 和外掛程式複製到使用新 Amazon MWAA 環境設定的 Amazon S3 儲存貯體。`requirements.txt`

步驟三：從現有環境匯出中繼資料

Apache Airflow 中繼資料表 (例如`dag`、`dag_tag`) ，並在您將更新的 DAG 檔案複製到環境的 Amazon S3 儲存貯體並且排程器剖析它們時`dag_code`自動填入。權限相關資料表也會根據您的 IAM 執行角色權限自動填入。您不需要遷移它們。

您可以移轉與 DAG 歷史記錄`variable`、`slot_pool`、`sla_miss`、、以及資料`log`表相關的資料。`xcom` `job`任務實例日誌存儲在CloudWatch日誌組下的日`airflow-`*`{environment_name}`*誌中。如果您想要查看較舊執行的工作執行個體記錄，則必須將這些記錄複製到新的環境記錄群組。我們建議您只移動數天的記錄，以降低相關成本。

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 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的環境移轉，也可以使用下列步驟來匯出 Apache Airflow 中繼資料。

匯出資料後，您可以在新環境中執行 DAG 以匯入資料。在匯出和匯入程序期間，會暫停所有其他 DAG。

從現有環境匯出詮釋資料的步驟

1. 使用建立 Amazon S3 儲存貯體AWS CLI以存放匯出後的資料。將`UUID`和取代為您`region`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，建議您啟用 Amazon S3 儲存貯體的預設加密。

2.

> **Note**
>
> 不適用於從自我管理環境移轉。

修改現有環境的執行角色，並新增下列原則，將寫入存取權授與您在步驟 1 中建立的值區。

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

3. 複製存amazon-mwaa-examples放庫，然後瀏覽至移轉案例的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，將其字串值取代為您建立S3_BUCKET用於存放匯出的中繼資料的 Amazon S3 儲存貯體。

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

5. 在目錄中找到requirements.txt檔metadata-migration案。如果您已經有現有環境的需求檔案，請將中指定的其他需求新增requirements.txt至檔案。如果您沒有現有的需求文件，則可以簡單地使用metadata-migration目錄中提供的文件。
6. 將export_data.py與您現有環境關聯的 Amazon S3 儲存貯體的 DAG 目錄。如果是從自我管理的環境移轉，請複製export_data.py到您的/dags資料夾。
7. 將更新版本複製requirements.txt到與現有環境相關聯的 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，存取 Apache 氣流使用者介面、取消暫停 db_export DAG，然後觸發工作流程以執行。

9. 確認中繼資料已匯出至 mwaa-migration-*{UUID}* Amazon S3 儲存貯體data/migration/*existing-version*_to_*new-version*/export/中，每個表都位於其專用檔案中。

步驟四：將中繼資料匯入新環境

將詮釋資料匯入新環境的步驟

1. 在中import_data.py，以您的資訊取代下列項目的字串值。

   - 對於從現有亞馬遜 MWAA 環境進行遷移：

     ```
     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記錄檔天數。

   - 如果是從自我管理環境

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

2. (選擇性) 僅import_data.py複製失敗的工作記錄。如果要複製所有任務日誌，請修改該getDagTasks函數，然後刪除ti.state = 'failed'如下面的代碼片段所示。

   ```
   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，並新增以下政策。權限政策允許 Amazon MWAA 從您匯出 Apache Airflow 中繼資料的 Amazon S3 儲存貯體讀取，並從現有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的環境移轉，則必須從策略中移除與CloudWatch記錄相關的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:{region}:{account_number}: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. 複製import_data.py到與新環境相關聯之 Amazon S3 儲存貯體的 DAG 目錄，然後存取 Apache 氣流使用者介面以取消暫停 db_import DAG 並觸發工作流程。新的 DAG 會在幾分鐘後出現在 Apache 氣流使用者介面中。
5. DAG 執行完成之後，請存取每個個別的 DAG，確認已複製 DAG 執行歷程記錄。

後續步驟

- 如需有關可用 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 如何處理自動調度資源工作者的詳細資訊，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。

相關資源

- Apache 氣流模型 (Apache 氣流文件) — 深入瞭解 Apache 氣流中繼資料資料庫模型。

將工作負載從遷移AWS Data Pipeline到亞馬遜 MWAA

AWS於二零一二年推出該AWS Data Pipeline服務。當時，客戶希望能夠使用各種運算選項在不同資料來源之間移動資料的服務。隨著數據傳輸需求隨著時間的推移而改變，因此可以滿足這些需求。您現在可以選擇最符合您業務需求的解決方案。您可以將工作負載移轉至下列任何AWS服務：

- 使用 Amazon Managed Workflows to Apache Airflow (Amazon MWAA) 管理 Apache Airflow 的 Workflows。
- 使用 Step Functions 來協調多個之間的工作流程AWS 服務。
- 用AWS Glue於執行和協調 Apache 星火應用程式。

您選擇的選項會依目前工作負載上AWS Data Pipeline。本主題介紹如AWS Data Pipeline何從 Amazon MWAA 遷移。

主題

選擇亞馬遜 MWAA

Amazon Managed Workflows the Apache Airflow (Amazon MWAA) 是一項適用於 Apache Airflow 的受管協同運作服務，可以大規模設定和操作端對端資料管道。Apache Airflow 是一種開放原始碼工具，用於以程式設計方式撰寫、排程和監視稱為工作流程的程序和工作序列。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，而不必管理基礎設施以提高可擴展性、可用性和安全性。Amazon MWAA 會自動擴展工作流程容量以符合您的需求，並與AWS安全服務整合，協助您快速安全地存取資料。

以下內容強調從遷移AWS Data Pipeline到 Amazon MWAA 的一些好處：

- 增強的可擴展性和效能 — Amazon MWAA 為定義和執行工作流程提供靈活且可擴展的架構。這使用戶可以輕鬆處理大型和複雜的工作流程，並利用諸如動態任務計劃，數據驅動的工作流程和並行性等功能。

- 改善監控和記錄功能 — Amazon MWAA 與 Amazon 整合，CloudWatch以增強工作流程的監控和記錄功能。Amazon MWAA 會自動將系統指標和日誌傳送到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。
- 與AWS服務和第三方軟體更完善的整合 — Amazon MWAA 與各種其他AWS服務 (例如 Amazon S3 和 Amazon Redshift) 以及第三方軟體 (例如 DBT、雪花和資料庫) 整合。AWS Glue這可讓您在不同的環境和服務之間處理和傳輸資料。
- 開放原始碼資料管道工具 — Amazon MWAA 利用您熟悉的相同開放原始碼 Apache 氣流產品。Apache Airflow 是專為處理資料管線管理的各個層面而設計的工具，包括擷取、處理、傳輸、完整性測試、品質檢查和確保資料歷程。
- 現代靈活的架構 — Amazon MWAA 利用容器化和雲端原生無伺服器技術。這意味著更大的靈活性和可攜性，以及更容易部署和管理您的工作流程環境。

架構和概念映射

AWS Data Pipeline而 Amazon MWAA 具有不同的架構和元件，這些架構和元件可能會影響遷移程序以及工作流程的定義和執行方式。本節概述了這兩種服務的架構和組件，並重點介紹一些主要差異。

AWS Data Pipeline和亞馬遜 MWAA 都是全受管服務。將工作負載遷移到 Amazon MWAA 時，您可能需要學習新概念，以使用 Apache 氣流來建立現有工作流程的模型。不過，您不需要管理基礎結構、修補程式背景工作程式，以及管理作業系統更新。

下表將中的關鍵概念AWS Data Pipeline與 Amazon MWAA 中的關鍵概念產生關聯。使用此資訊做為設計移轉計劃的起點。

概念	AWS Data Pipeline	亞馬遜分公司
管道定義	AWS Data Pipeline使用定義工作流程的 JSON 配置檔案。	Amazon MWAA 使用以 Python 為基礎的有向無環圖 (DAG) 來定義工作流程。
管道執行環境	Workflows 執行於 Amazon EC2 執行個體之上。 AWS Data Pipeline代表您佈建和管理這些執行個體。	亞馬遜 MWAA 使用亞馬遜 ECS 容器化環境來執行任務。
Pipeline	活動是處理作為工作流程一部分執行的任務。	運算子 (工作) 是工作流程的基本處理單元。

概念	AWS Data Pipeline	亞馬遜分公司
	先決條件包含條件陳述式，這些條件陳述式必須為 true，才能執行活動。	感測器 (T asks) 代表條件陳述式，可在執行前等待資源或工作完成。
	中的資源AWS Data Pipeline是指執行管線活動指定之工作的AWS計算資源。Amazon EC2 和亞馬遜 EMR 有兩種可用資源。	使用 DAG 中的任務，您可以定義各種運算資源，包括亞馬遜 ECS、亞馬遜 EMR 和亞馬遜 EKS。亞馬遜 MWAA 在亞馬遜 ECS 上執行的工作者上執行 Python 操作。
管道執行	AWS Data Pipeline支援以一般比率為基礎的排程執行，以及以 Cron 為基礎的模式。	Amazon MWAA 支援使用 cron 運算式和預設集進行排程，以及自訂時間表。
	例證是指管線的每個執行。	D AG 執行是指 Apache 氣流工作流程的每次執行。
	嘗試是指重試失敗的作業。	Amazon MWAA 支援您在 DAG 層級或工作層級定義的重試次數。

實作範例

在許多情況下，遷移到 Amazon MWAAAWS Data Pipeline 後，您將能夠重複使用目前正在協調的資源。下列清單包含針對最常見使用AWS Data Pipeline案例使用 Amazon MWAA 的範例實作。

- 執行亞馬遜 EMR 工作 (AWS研討會)
- 為阿帕奇蜂巢和 Hadoop 創建一個自定義插件（亞馬遜 MWAA 用戶指南）
- 將資料從 S3 複製到 Redshift (AWS研討會)
- 在遠端亞馬遜 ECS 執行個體上執行殼層指令碼 (亞馬遜 MWAA 使用者指南)
- 協調混合式 (內部部署) 工作流程 (部落格文章)

如需其他自學課程和範例，請參閱下列：

- 亞馬遜 MWAA 教學課程
- 亞馬遜 MWAA 程式碼範例

價格比較

的定價取決AWS Data Pipeline於管線的數量，以及每個管道的使用量。您每天執行一次以上的活動 (高頻率)，每個活動每月費用為 $1。您每天執行一次或更少 (低頻率) 的活動，每個活動每月的費用為 $0.60。非作用中管道的價格為每個管線 1 美元。如需詳細資訊，請參閱 AWS Data Pipeline 定價頁面。

Amazon MWAA 的定價取決於您受管 Apache Airflow 環境存在的時間持續時間，以及提供更多員工或排程器容量所需的任何額外 auto 擴展。您可以按小時支付 Amazon MWAA 環境用量的費用 (以一秒解析度計費)，費用會根據環境的大小而有所不同。Amazon MWAA 會根據您的環境組態自動調整工作者的數量。 AWS分別計算額外工人的成本。如需使用各種 Amazon MWAA 環境大小的小時費用的詳細資訊，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：

- 亞馬遜 MWAA 應用程式介面參考
- 監控亞馬遜 MWAA 上的儀表板和警報
- 亞馬遜 MWAA 上阿帕奇氣流的性能調整

Amazon MWAA 文件歷史記錄

下表說明 Amazon MWAA 遷移指南自 2018 年 3 月起的重要增補。

變更	描述	日期
有關將工作負載從亞馬遜 MWAA 遷移AWS Data Pipeline到的新主題	已新增將現有工作負載從 Amazon MWAA 遷移AWS Data Pipeline到 Amazon MWAA 的新資訊和指引。使用此資訊可協助您設計移轉計劃。 • 將工作負載從遷移AWS Data Pipeline到亞馬遜 MWAA	2023 年 4 月 14 日
亞馬遜 MWAA 遷移指南發布	亞馬遜 MWAA 現在提供有關遷移到新的亞馬遜 MWAA 環境的詳細指導。Amazon MWAA 移轉指南中所述的步驟適用於從現有 Amazon MWAA 環境或自我管理的 Apache 氣流部署進行管理。 • 關於亞馬遜 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。