aws

迁移指南

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: 迁移指南

Table of Contents

什么是《Amazon MWAA 迁移指南》？

Amazon MWAA 是 Apache Airflow 的托管式编排服务，让您能够更轻松地在云中大规模操作数据管道。Amazon MWAA 管理 Apache Airflow 的配置和持续维护，因此您无需再担心修补、扩展或保护实例。

Amazon MWAA 会自动扩缩执行任务的计算资源，以按需提供稳定的性能。Amazon MWAA 默认保护您的数据。工作负载使用 Amazon Virtual Private Cloud 在您自己的隔离且安全的云环境中运行。这样可以确保使用 AWS Key Management Service 自动加密数据。

使用本指南将自行管理的 Apache Airflow 工作流程迁移到 Amazon MWAA，或者将现有的 Amazon MWAA 环境升级到新的 Apache Airflow 版本。迁移教程介绍了如何创建或克隆新的 Amazon MWAA 环境、迁移工作流程资源以及如何将工作流程元数据和日志传输到新环境。

在尝试迁移教程之前，我们建议您先阅读以下主题。

- 网络架构
- 重要注意事项：

Amazon MWAA 网络架构

下一节介绍构成 Amazon MWAA 环境的主要组件，以及每个环境为管理其资源、保护数据安全以及为工作流程提供监控和可见性而集成的一组 AWS 服务。

主题

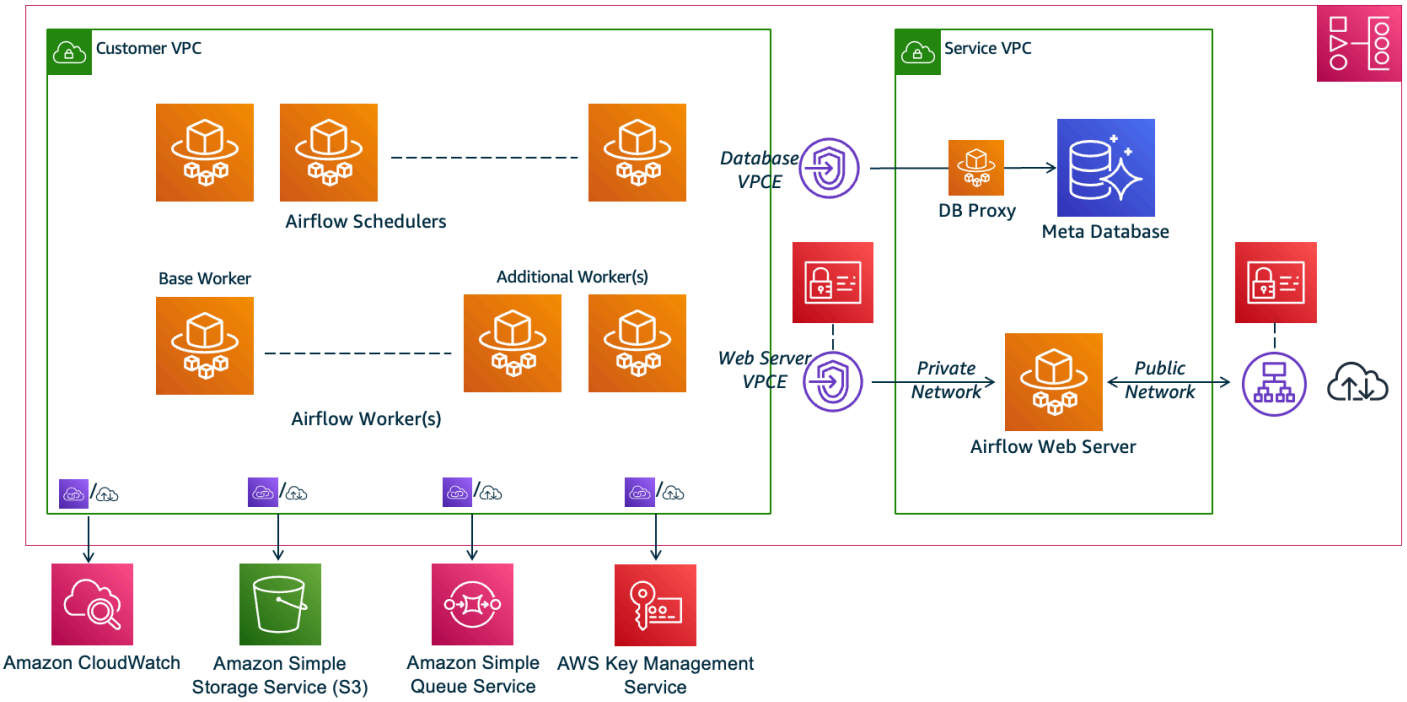
Amazon MWAA 组件

Amazon MWAA 环境由以下四个主要组件组成：

1. 计划程序— 解析和监控所有 DAG，并在满足 DAG 的依赖项时对任务进行排队，以便执行。Amazon MWAA 将计划程序部署为至少有 2 个计划程序的 AWS Fargate 集群。根据工作负载，您可以将计划程序数量增加到五个。有关 Amazon MWAA 环境类的更多信息，请参阅 Amazon MWAA 环境类。
2. 工作线程— 运行计划任务的一个或多个 Fargate 任务。环境中的工作线程数量由您指定的最小和最大数量之间的范围决定。当排队和正在运行的任务数量超过现有工作线程的处理能力时，Amazon MWAA 会开始自动扩缩工作线程。当正在运行且排队的任务总和为零超过两分钟时，Amazon MWAA 会将工作线程数量缩减到最低。有关 Amazon MWAA 如何处理自动扩缩工作线程的更多信息，请参阅 Amazon MWAA 自动扩缩。
3. Web 服务器— 运行 Apache Airflow Web UI。您可以将 Web 服务器配置为私有或公有网络访问权限。在这两种情况下，Apache Airflow 用户的访问权限都由您在 AWS Identity and Access Management（IAM）中定义的访问控制策略控制。有关为环境配置 IAM 访问策略的更多信息，请参阅访问 Amazon MWAA 环境。
4. 数据库— 存储有关 Apache Airflow 环境和工作流程的元数据，包括 DAG 运行历史记录。该数据库是由 AWS 托管的单租户 Aurora PostgreSQL 数据库，可通过私有保护的 Amazon VPC 端点从计划程序和工作线程的 Fargate 容器访问。

每个 Amazon MWAA 环境还与一组 AWS 服务交互以处理各种任务，包括存储和访问 DAG 和任务依赖项、保护静态数据以及记录和监控环境。下图演示了 Amazon MWAA 环境的不同组件。

Amazon MWAA Architecture



> **Note**
>
> 服务 Amazon VPC 不是共享 VPC。Amazon MWAA 会为您创建的每个环境创建一个 AWS 自有的 VPC。

- Amazon S3— Amazon MWAA 将所有工作流程资源（例如 DAG、要求和插件文件）存储在 Amazon S3 存储桶中。有关创建存储桶作为创建环境的一部分以及上传 Amazon MWAA 资源的更多信息，请参阅《Amazon MWAA 用户指南》中的为 Amazon MWAA 创建 Amazon S3 存储桶。
- Amazon SQS — Amazon MWAA 使用 Amazon SQS 将工作流程任务与 Celery 执行程序一起排队。
- Amazon ECR — Amazon ECR 托管所有 Apache Airflow 镜像。Amazon MWAA 仅支持 AWS 托管 Apache Airflow 镜像。
- AWS KMS— Amazon MWAA 使用 AWS KMS 来确保静态数据安全。默认情况下，Amazon MWAA 使用AWS 托管的 AWS KMS 密钥，但您可以将环境配置为使用您自己的由客户托管的 AWS KMS 密钥。有关使用您自己的由客户托管的 AWS KMS 密钥的更多信息，请参阅《Amazon MWAA 用户指南》中的用于数据加密的由客户托管的密钥。
- CloudWatch— Amazon MWAA 与 CloudWatch 集成，并向 CloudWatch 提供 Apache Airflow 日志和环境指标，允许您监控 Amazon MWAA 资源并排查问题。

连接

Amazon MWAA 环境需要访问与其集成的所有 AWS 服务。Amazon MWAA 执行角色控制如何授予 Amazon MWAA 代表您连接其他 AWS 服务的访问权限。要实现网络连接，您可以为 Amazon VPC 提供公共互联网访问权限，也可以创建 Amazon VPC 端点。有关为环境配置 Amazon VPC 端点（AWS PrivateLink）的更多信息，请参阅《Amazon MWAA 用户指南》中的在 Amazon MWAA 上管理对 VPC 端点的访问。

Amazon MWAA 在计划程序和工作线程上安装要求。如果您的要求来自公共 PyPI 存储库，则环境需要连接到互联网才能下载所需的库。对于私有环境，您可以使用私有 PyPI 存储库，也可以将这些库捆绑成.whl 文件作为环境的自定义插件。

当您在私有模式下配置 Apache Airflow 时，Amazon VPC 只能通过Amazon VPC 端点访问 Apache Airflow UI。

有关联网的更多信息，请参阅《Amazon VPC 用户指南》中的联网。

重要注意事项：

在迁移到新的 Amazon MWAA 环境之前，请阅读以下主题。

主题

身份验证

亚马逊 MWAA 使用 AWS Identity and Access Management (IAM) 来控制对 Apache Airflow 用户界面的访问。您必须创建和管理 IAM 策略，这些策略授予 Apache Airflow 用户访问 Web 服务器和管理 DAG 的权限。您可以跨不同账户使用 IAM 管理 Apache Airflow 默认角色的身份验证和授权。

通过创建自定义 Airflow 角色并将其映射到 IAM 主体，您可以进一步管理和限制 Apache Airflow 用户仅访问部分工作流程 DAG。有关更多信息和 step-by-step 教程，请参阅教程：限制 Amazon MWAA 用户访问一部分 DAG。

您也可以配置联合身份以访问 Amazon MWAA。有关更多信息，请参阅下列内容。

- 具有公共访问权限的 Amazon MWAA 环境 — 使用 Okta 作为身份提供商，在 AWS Compute Blog 上使用 Amazon MWAA。
- 具有私有访问权限的 Amazon MWAA 环境 — 使用联合身份访问私有 Amazon MWAA 环境。

执行角色

Amazon MWAA 使用执行角色向您的环境授予访问其他 AWS 服务的权限。您可以通过向角色添加相关权限来为工作流程提供 AWS 服务访问权限。如果您在首次创建环境时选择默认选项来创建新的执行角色，则 Amazon MWAA 会为该角色附加所需的最低权限，但 Amazon MWAA 会自动为其添加所有 CloudWatch 日志组的日志除外。

一旦创建了执行角色，Amazon MWAA 就无法代表您管理其权限策略。要更新执行角色，必须根据需要编辑策略以添加和删除权限。例如，您可以将 Amazon MWAA 环境 与 AWS Secrets Manager集成作为后端，以安全地存储密钥和连接字符串，以便在 Apache Airflow 工作流程中使用。为此，请将以下权限策略附加到环境的执行角色。

```
{
```

```
    "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": "*"
        }
    ]
}
```

与其他 AWS 服务的集成遵循类似的模式：您将相关的权限策略添加到您的 Amazon MWAA 执行角色中，授予亚马逊 MWAA 访问该服务的权限。有关管理 Amazon MWAA 执行角色的更多信息以及要查看更多示例，请访问《Amazon MWAA 用户指南》中的 Amazon MWAA 执行角色。

迁移到新的 Amazon MWAA 环境

以下主题描述了将现有 Apache Airflow 工作负载迁移到新的 Amazon MWAA 环境的步骤。您可以使用以下步骤从旧版本的 Amazon MWAA 迁移到新发布版本，或者将自行管理的 Apache Airflow 部署迁移到 Amazon MWAA。本教程假设您正在从现有的 Apache Airflow v1.10.12 迁移到运行 Apache Airflow v2.5.1 的新的 Amazon MWAA，但您可以使用相同的过程从不同的 Apache Airflow 版本迁移或迁移到不同的 Apache Airflow 版本。

主题

先决条件

为了能够完成这些步骤并迁移环境，您需要具备以下条件：

- Apache Airflow 部署。这可以是自行管理的，也可以是现有的 Amazon MWAA 环境。
- 在本地系统上安装的 Docker。
- 已安装的 AWS Command Line Interface 版本 2。

步骤 1：创建一个运行最新支持的 Apache Airflow 版本的 Amazon MWAA 环境

您可以使用《Amazon MWAA 用户指南》中的 Amazon MWAA 入门中的详细步骤或使用 AWS CloudFormation 模板来创建环境。如果您要从现有的 Amazon MWAA 环境迁移并已使用 AWS CloudFormation 模板创建旧环境，则可以更改 `AirflowVersion` 属性以指定新版本。

```
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，如果从现有 Amazon MWAA 环境迁移，则可以复制以下 Python 脚本，该脚本使用 Python 的 AWS SDK（Boto3）来克隆环境。您也可以下载脚本。

Python 脚本

```
# 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)
```

步骤 2：迁移工作流程资源

Apache Airflow v2 是主要版本。如果要从 Apache Airflow v1 迁移，则必须准备工作流程资源并验证对 DAG、要求和插件所做的更改。为此，我们建议使用 Docker 和 Amazon MWAA 本地运行器在本地操作系统上配置 Apache Airflow 的桥版本。Amazon MWAA 本地运行器提供了命令行界面（CLI）实用工具，可在本地复制 Amazon MWAA 环境。

每当您更改 Apache Airflow 版本时，请确保在 `requirements.txt` 中引用正确的 `--constraint` URL。

要迁移工作流程资源，请执行以下操作

1. 创建 aws-mwaa-local-runner 存储库的分支，然后克隆 Amazon MWAA 本地运行器的副本。
2. 查看 aws-mwaa-local-runner 存储库的 `v1.10.15` 分支。Apache Airflow 发布 v1.10.15 作为桥版本，以帮助迁移到 Apache Airflow v2，尽管 Amazon MWAA 不支持 v1.10.15，但您可以使用 Amazon MWAA 本地运行器来测试资源。
3. 使用 Amazon MWAA 本地运行器 CLI 工具来构建 Docker 映像并在本地运行 Apache Airflow。有关更多信息，请参阅 GitHub 存储库中的 README。
4. 使用本地运行的 Apache Airflow，按照 Apache Airflow 文档网站从 1.10 升级到 2 中描述的步骤进行操作。

    a. 要更新 `requirements.txt`，请按照《Amazon MWAA 用户指南》中管理 Python 依赖项中推荐的最佳实践进行操作。

    b. 如果您已将自定义运算符和传感器与现有 Apache Airflow v1.10.12 环境的插件捆绑在一起，请将其移动至 DAG 文件夹。有关 Apache Airflow v2+ 模块管理最佳实践的更多信息，请参阅 Apache Airflow 文档网站中的模块管理。

5. 对工作流程资源进行必要的更改后，请查看 aws-mwaa-local-runner 存储库的 `v2.5.1` 分支，然后在本地测试更新的工作流程 DAG、要求和自定义插件。如果您要迁移到其他 Apache Airflow 版本，则可以改为使用适合您版本的本地运行器分支。
6. 成功测试工作流程资源后，将 DAG、`requirements.txt` 和插件复制到您使用新 Amazon MWAA 环境配置的 Amazon S3 存储桶中。

步骤 3：从现有环境中导出元数据

当您将更新的 DAG 文件复制到环境的 Amazon S3 存储桶并且计划程序对其进行解析时，Apache Airflow 元数据表格（例如`dag`、`dag_tag` 和 `dag_code`）会自动填充。与权限相关的表格也会根据 IAM 执行角色权限自动填充。您不需要迁移它们。

如果需要，您可以迁移与 DAG 历史记录、`variable`、`slot_pool`、`sla_miss` 以及（如果需要）`xcom`、`job` 和 `log` 表格相关的数据。任务实例日志存储在 `airflow-`*`{environment_name}`* 日志组下的 CloudWatch Logs 中。如果要查看较早运行的任务实例日志，则必须将这些日志复制到新的环境日志组中。我们建议您只移动几天的日志，以降低相关成本。

如果您要从现有的 Amazon MWAA 环境迁移，则无法直接访问元数据数据库。您必须运行 DAG 才能将元数据从现有 Amazon MWAA 环境导出到您选择的 Amazon S3 存储桶。如果您要从自行管理的环境迁移，也可以使用以下步骤导出 Apache Airflow 元数据。

导出数据后，您可以在新的环境中运行 DAG 来导入数据。在导出和导入过程中，所有其他 DAG 都将暂停。

要从现有环境中导出元数据，请执行以下操作

1. 使用 AWS CLI 创建 Amazon S3 存储桶，用于存储导出的数据。用您的信息替换 `UUID` 和 `region`。

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

> **Note**
>
> 如果您要迁移敏感数据，例如存储在变量中的连接，我们建议您为 Amazon S3 存储桶启用默认加密。

2.

> **Note**
>
> 不适用于从自行管理的环境中迁移。

修改现有环境的执行角色并添加以下策略以授予对您在步骤 1 中创建的存储桶的写入权限。

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

3. 克隆 amazon-mwaa-examples 存储库，然后导航到适合迁移场景的 metadata-migration 子目录。

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

4. 在 export_data.py 中，将字符串值替换为 S3_BUCKET，即您创建的用于存储导出的元数据的 Amazon S3 存储桶。

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

5. 将 requirements.txt 文件放入 metadata-migration 目录中。如果您已有现有环境的要求文件，请将 requirements.txt 中指定的额外要求添加到您的文件中。如果您没有现有的要求文件，则只需使用 metadata-migration 目录中提供的要求文件即可。
6. 将 export_data.py 复制到与现有环境相关联的 Amazon S3 存储桶的 DAG 目录中。如果从自行管理的环境中迁移，请将 export_data.py 复制到 /dags 文件夹。
7. 将更新内容 requirements.txt 复制到与现有环境相关联的 Amazon S3 存储桶，然后编辑环境以指定新的 requirements.txt 版本。

8. 环境更新后，访问 Apache Airflow UI，取消暂停 db_export DAG，然后触发工作流程运行。
9. 确认元数据已导出到 mwaa-migration-*{UUID}* Amazon S3 存储桶中的 data/migration/*existing-version*_to_*new-version*/export/ 中，且每个表格都位于自己的专用文件中。

步骤 4：将元数据导入新环境

要将元数据导入新环境，请执行以下操作

1. 在 import_data.py 中，将以下内容的字符串值替换为您的信息。
   - 要从现有 Amazon MWAA 环境迁移，请执行以下操作：

     ```
     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 控制工作流程将指定天数的日志文件复制到新环境。
   - 要从自行管理环境迁移，请执行以下操作：

     ```
     S3_BUCKET = 'mwaa-migration-{UUID}'
     NEW_ENV_NAME='{new_environment_name}'
     ```
2. （可选）import_data.py 仅复制失败的任务日志。如果要复制所有任务日志，请修改 getDagTasks 函数并删除 ti.state = 'failed'，如以下代码片段所示。

   ```
   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. 修改新环境的执行角色并添加以下策略。权限策略允许 Amazon MWAA 从您导出 Apache Airflow 元数据的 Amazon S3 存储桶中读取数据，并允许从现有日志组复制任务实例日志。用您的信息替换所有占位符。

> **Note**
>
> 如果您要从自行管理的环境迁移，则必须从策略中移除 CloudWatch Logs 相关权限。

```
{
    "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. 将 `import_data.py` 复制到与新环境关联的 Amazon S3 存储桶的 DAG 目录中，然后访问 Apache Airflow UI 取消暂停 `db_import` DAG 并触发工作流程。几分钟后，新的 DAG 将出现在 Apache Airflow UI 中。
5. DAG 运行完成后，访问每个 DAG 来验证 DAG 运行历史记录是否已复制。

后续步骤

- 有关可用 Amazon MWAA 环境类和功能的更多信息，请参阅《Amazon MWAA 用户指南》中的 Amazon MWAA 环境类。
- 有关 Amazon MWAA 如何处理自动扩缩工作线程的更多信息，请参阅《Amazon MWAA 用户指南》中的Amazon MWAA 自动扩缩。
- 有关 Amazon MWAA REST API 的更多信息，请参阅 Amazon MWAA REST API。

相关的 资源

- Apache Airflow 模型（Apache Airflow 文档）— 了解有关 Apache Airflow 元数据数据库模型的更多信息。

将工作负载从 AWS Data Pipeline 迁移到 Amazon MWAA

AWS 于 2012 年推出 AWS Data Pipeline 服务。当时，客户想要一种服务，以允许他们使用各种计算选项在不同数据来源之间移动数据。随着时间的推移，数据传输需求发生了改变，因此满足这些需求的解决方案也随之改变。现在，您可以选择最符合您的业务要求的解决方案。您可以将工作负载迁移到以下任何 AWS 服务：

- 使用 Amazon Managed Workflows for Apache Airflow（Amazon MWAA）来管理 Apache Airflow 的工作流程编排。
- 使用 Step Functions 来编排多个 AWS 服务 之间的工作流程。
- 使用 AWS Glue 来运行和编排 Apache Spark 应用程序。

您选择的选项因 AWS Data Pipeline 上的当前工作负载而定。本主题介绍如何从 AWS Data Pipeline 迁移到 Amazon MWAA。

主题

选择 Amazon MWAA

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) 是一项适用于 Apache Airflow 的托管式编排服务，让您能够更轻松地在云中大规模设置和操作端到端的数据管道。Apache Airflow 是一种开源工具，用于以编程方式编写、调度和监视统称为工作流程的各种流程和任务序列。借助 Amazon MWAA，您可以使用 Apache Airflow 和 Python 编程语言来创建工作流程，而无需管理底层基础设施以实现可扩展性、可用性和安全性。Amazon MWAA 会自动扩展其工作流程容量以满足您的需求，并与 AWS 安全服务集成，帮助您快速、安全地访问数据。

以下内容重点介绍了从 AWS Data Pipeline 迁移到 Amazon MWAA 的一些好处：

- 增强了可扩展性和性能 — Amazon MWAA 为定义和执行工作流程提供了灵活且可扩展的框架。这使用户可以轻松处理大型和复杂的工作流程，并利用动态任务调度、数据驱动的工作流程和并行性等功能。
- 改进了监控和日志记录 — Amazon MWAA 与 Amazon CloudWatch 集成，以增强对工作流程的监控和记录。Amazon MWAA 会自动向 CloudWatch 发送系统指标和日志。这意味着您可以实时跟踪工作流程的进度和性能，并识别出现的任何问题。
- 更好地与 AWS 服务和第三方软件集成 — Amazon MWAA 可与各种其他 AWS 服务（例如 Amazon S3、AWS Glue 和 Amazon Redshift）以及第三方软件（例如 DBT、Snowflake 和 Databricks）集成。这使您可以跨不同的环境和服务处理和传输数据。
- 开源数据管道工具 — Amazon MWAA 利用的是您熟悉的开源 Apache Airflow 产品。Apache Airflow 是一款专门构建的工具，旨在处理数据管道管理的各个方面，包括摄取、处理、传输、完整性测试、质量检查和确保数据世系。
- 现代灵活的架构 — Amazon MWAA 利用容器化和云原生无服务器技术。这意味着可以提高灵活性和便携性，并更轻松地部署和管理工作流程环境。

架构和概念映射

AWS Data Pipeline 和 Amazon MWAA 拥有不同的架构和组件，这可能会影响迁移过程以及工作流程的定义和执行方式。本节概述了这两种服务的架构和组件，并重点介绍了其中的一些主要区别。

AWS Data Pipeline 和 Amazon MWAA 都是完全托管的服务。当您将工作负载迁移到 Amazon MWAA 时，您可能需要学习新的概念，以便使用 Apache Airflow 对现有工作流程进行建模。但是，您无需管理基础设施、修补工作线程和管理操作系统更新。

下表将 AWS Data Pipeline 中的关键概念与 Amazon MWAA 中的关键概念相关联。使用此信息作为起点来设计迁移计划。

概念	AWS Data Pipeline	Amazon MWAA
管道定义	AWS Data Pipeline 使用基于 JSON 的配置文件来定义工作流程。	Amazon MWAA 使用基于 Python 的有向无环图（DAG）来定义工作流程。
管道执行环境	工作流程运行在 Amazon EC2 实例上。AWS Data Pipeline 代表您配置和管理这些实例。	Amazon MWAA 使用 Amazon ECS 容器化环境来运行任务。

概念	AWS Data Pipeline	Amazon MWAA
管道组件	活动是作为工作流程一部分运行的处理任务。	运算符（任务）是工作流程的基本处理单元。
	先决条件包含必须为 true 的条件语句，然后活动才能运行。	传感器（任务）表示条件语句，可以等待资源或任务完成后再运行。
	AWS Data Pipeline 中的资源是指执行管道活动所指定工作的 AWS 计算资源。Amazon EC2 和 Amazon EMR 是两个可用资源。	使用 DAG 中的任务，您可以定义各种计算资源，包括 Amazon ECS、Amazon EMR 和 Amazon EKS。Amazon MWAA 对在 Amazon ECS 上运行的工作线程执行 Python 操作。
管道执行	AWS Data Pipeline 支持使用常规的基于速率和基于 cron 的模式来调度运行。	Amazon MWAA 支持使用 cron 表达式和预设以及自定义时间表进行调度。
	实例是指管道的每次运行。	DAG 运行是指 Apache Airflow 工作流程的每次运行。
	尝试是指重试失败的操作。	Amazon MWAA 支持您在 DAG 级别或任务级别定义的重试。

示例实施

在许多情况下，迁移到 Amazon MWAA 后，您将能够重复使用当前采用 AWS Data Pipeline 编排的资源。以下列表包含针对最常见AWS Data Pipeline 用例使用 Amazon MWAA 的示例实施。

- 运行 Amazon EMR 任务（AWS 研讨会）
- 为 Apache Hive 和 Hadoop 创建自定义插件（《Amazon MWAA 用户指南》）
- 将数据从 S3 复制到 Redshift（AWS 研讨会）
- 在远程 Amazon ECS 实例上执行 shell 脚本（《Amazon MWAA 用户指南》）

- 编排混合（本地）工作流程（博客文章）

有关额外教程和示例，请参阅以下：

- Amazon MWAA 教程
- Amazon S3 代码示例

定价比较

AWS Data Pipeline 的定价取决于管道数以及每个管道的使用量。您每天运行一次以上（高频率）的活动每项活动每月花费 1 美元。您每天运行一次或以下（低频率）的活动每项活动每月花费 0.6 美元。非活跃管道的价格为每条管道 1 美元。有关更多信息，请参阅 AWS Data Pipeline 价格页面。

Amazon MWAA 的定价基于托管 Apache Airflow 环境存在的时间长短，以及提供更多工作线程或计划程序容量所需的任何额外自动扩缩。您按小时为 Amazon MWAA 环境使用量付费（按一秒钟的分辨率计费），费用因环境大小而异。Amazon MWAA 会根据环境配置自动扩缩工作线程数。AWS 分别计算额外工作线程的成本。有关使用各种 Amazon MWAA 环境大小的小时费用的更多信息，请参阅Amazon MW AA 定价页面。

相关的 资源

有关更多信息以及 使用 Amazon MWAA 的最佳实践，请参阅以下资源：

- Amazon MWAA API 参考
- 监控 Amazon MWAA 上的控制面板和警报
- Amazon MWAA 上的 Apache Airflow 的性能调整

Amazon MWAA 文档历史记录

下表介绍了自 2022 年 3 月以来对《Amazon MWAA 迁移指南》的重要新增。

变更	说明	日期
关于将工作负载从 AWS Data Pipeline 迁移到 Amazon MWAA 的新主题	已添加有关将现有工作负载从 AWS Data Pipeline 迁移到 Amazon MWAA 的新信息和指南。使用此信息来帮助您设计迁移计划。 • 将工作负载从 AWS Data Pipeline 迁移到 Amazon MWAA	2023 年 4 月 14 日
《Amazon MWAA 迁移指南》发布	Amazon MWAA 现在提供了有关迁移到新的 Amazon MWAA 环境的详细指导。《Amazon MWAA 迁移指南》中描述的步骤适用于从现有 Amazon MWAA 环境或从自行管理的 Apache Airflow 部署迁移。 • 关于《Amazon MWAA 迁移指南》	2022 年 3 月 7 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。