aws

Guía de migración

Amazon Managed Workflows para Apache Airflow

Amazon Managed Workflows para Apache Airflow: Guía de migración

Table of Contents

¿Qué es la guía de migración de Amazon MWAA?

Amazon Managed Workflows para Apache Airflow es un servicio de orquestación administrada para Apache Airflow que le permite operar canalizaciones de datos integrales en la nube a escala. Amazon MWAA gestiona el aprovisionamiento y el mantenimiento continuo de Apache Airflow para que ya no tenga que preocuparse por aplicar parches, escalar o proteger las instancias.

Amazon MWAA escala automáticamente los recursos informáticos que ejecutan las tareas para ofrecer un rendimiento uniforme bajo demanda. Amazon MWAA protege sus datos de forma predeterminada. Sus cargas de trabajo se ejecutan en su propio entorno de nube seguro y aislado mediante Amazon Virtual Private Cloud. Esto garantiza que los datos se cifren automáticamente mediante AWS Key Management Service.

Utilice esta guía para migrar sus flujos de trabajo autogestionados de Apache Airflow a Amazon MWAA o para actualizar un entorno de Amazon MWAA existente a una nueva versión de Apache Airflow. El tutorial de migración describe cómo puede crear o clonar un nuevo entorno de Amazon MWAA, migrar los recursos de su flujo de trabajo y transferir los metadatos y registros del flujo de trabajo a su nuevo entorno.

Antes de iniciar el tutorial de migración, le recomendamos que consulte los siguientes temas.

- Arquitectura de redes
- Consideraciones clave

Arquitectura de red de Amazon MWAA

En la siguiente sección, se describen los componentes principales que forman un entorno de Amazon MWAA y el conjunto de servicios de AWS con los que se integra cada entorno para administrar sus recursos, mantener sus datos seguros y proporcionar supervisión y visibilidad a sus flujos de trabajo.

Temas

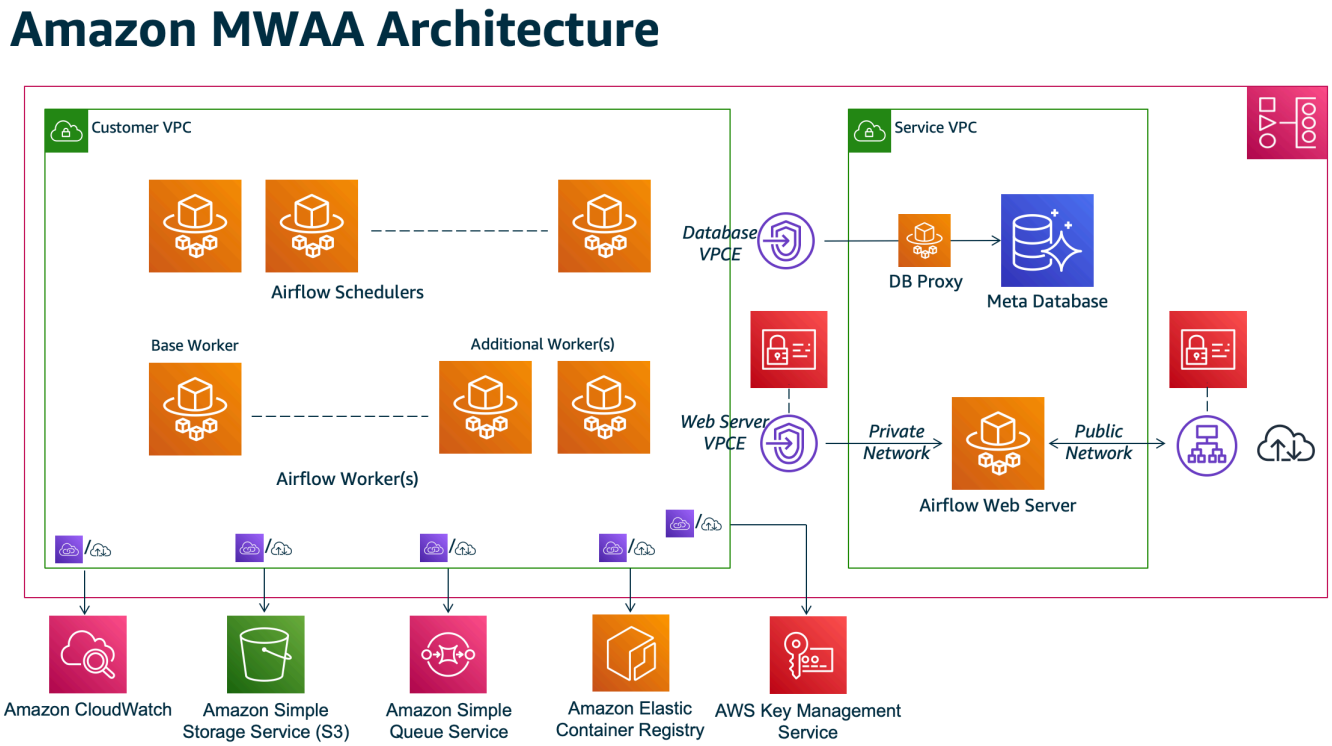
Componentes de Amazon MWAA

Los entornos de Amazon MWAA están formados por los siguientes cuatro componentes principales:

1. Programador: analiza y monitorea todos sus DAG y pone en cola las tareas para que se ejecuten cuando se cumplan las dependencias de un DAG. Amazon MWAA implementa el programador como un clúster de AWS Fargate con un mínimo de 2 programadores. Puede aumentar el número de programadores hasta cinco, en función de su carga de trabajo. Para obtener más información sobre las clases de entorno de Amazon MWAA, consulte Clases de entornos de Amazon MWAA.
2. Trabajadores: una o varias tareas de Fargate que ejecutan las tareas programadas. La cantidad de trabajadores del entorno viene determinada por un rango entre el número mínimo y el máximo que especifique. Amazon MWAA comienza a realizar un escalado automático de los trabajadores cuando el número de tareas en cola y en ejecución es superior al que pueden gestionar sus trabajadores actuales. Cuando las tareas en ejecución y en cola suman cero durante más de dos minutos, Amazon MWAA reduce el número de trabajadores al mínimo. Para obtener más información sobre cómo gestiona Amazon MWAA el escalado automático de trabajadores, consulte Escalado automático de Amazon MWAA.
3. Servidor web: ejecuta la interfaz de usuario web de Apache Airflow. Puede configurar el servidor web con acceso a la red pública o privada. En ambos casos, el acceso a los usuarios de Apache Airflow se controla mediante la política de control de acceso que usted defina en AWS Identity and Access Management (IAM). Para obtener más información sobre la configuración de las políticas de acceso de IAM para su entorno, consulte Acceso a un entorno de Amazon MWAA.
4. Base de datos: almacena metadatos sobre el entorno de Apache Airflow y sus flujos de trabajo, incluido el historial de ejecución de DAG. La base de datos es una base de datos basada en

Aurora PostgreSQL de un único inquilino administrada por AWS, y a la que pueden acceder los contenedores de Fargate de los programadores y los trabajadores a través de un punto de conexión de Amazon VPC protegido de forma privada.

Cada entorno de Amazon MWAA interactúa, además, con un conjunto de servicios de AWS con el fin de gestionar diversas tareas, como almacenar los DAG y las dependencias de las tareas y acceder a ellos, proteger los datos en reposo, y registrar y supervisar el entorno. En el siguiente diagrama se muestran los diferentes componentes de un entorno de Amazon MWAA.



> **Note**
>
> El servicio Amazon VPC no es una VPC compartida. Amazon MWAA crea una VPC propiedad de AWS para cada entorno que cree.

- Amazon S3: Amazon MWAA almacena todos los recursos de su flujo de trabajo, como los DAG, los requisitos y los archivos de complementos, en un bucket de Amazon S3. Para obtener más información sobre la creación del bucket como parte de la creación del entorno y sobre la carga de sus recursos de Amazon MWAA, consulte Creación de un bucket de Amazon S3 para Amazon MWAA en la Guía del usuario de Amazon MWAA.

- Amazon SQS: Amazon MWAA utiliza Amazon SQS para poner en cola las tareas del flujo de trabajo con un ejecutor de Celery.
- Amazon ECR: todas las imágenes de Apache Airflow se alojan en Amazon ECR. Amazon MWAA solo admite imágenes administradas por AWS de Apache Airflow.
- AWS KMS: Amazon MWAA utiliza AWS KMS para garantizar que sus datos estén seguros en reposo. De forma predeterminada, Amazon MWAA usa claves AWS KMS administradas por AWS, pero usted puede configurar su entorno para que use su propia clave AWS KMS administrada por el cliente. Para obtener más información sobre el uso de su propia clave AWS KMS administrada por el cliente, consulte Claves administradas por el cliente para el cifrado de datos en la Guía del usuario de Amazon MWAA.
- CloudWatch: Amazon MWAA se integra con CloudWatch y proporciona registros de Apache Airflow y métricas del entorno a CloudWatch, lo que le permite supervisar sus recursos de Amazon MWAA y solucionar problemas.

Conectividad

Su entorno de Amazon MWAA necesita acceder a todos los servicios de AWScon los que se integra. El rol de ejecución de Amazon MWAA controla la forma en que se concede el acceso a Amazon MWAA para conectarse a otros servicios de AWS en su nombre. Para la conectividad de red, puede proporcionar acceso público a Internet a su Amazon VPC o crear puntos de conexión de Amazon VPC. Para obtener más información sobre la configuración de los puntos de conexión de Amazon VPC (AWS PrivateLink) para su entorno, consulte Administración del acceso a los puntos de conexión de VPC en Amazon MWAA en la Guía del usuario de Amazon MWAA.

Amazon MWAA instala los requisitos en el programador y en el trabajador. Si sus requisitos provienen de un repositorio público PyPi, el entorno necesita conectividad a Internet para descargar las bibliotecas necesarias. Para entornos privados, puede usar un repositorio PyPi privado o agrupar las bibliotecas en archivos `.whl` como complementos personalizados para su entorno.

Al configurar Apache Airflow en modo privado, su Amazon VPC solo puede acceder a la interfaz de usuario de Apache Airflow a través de los puntos de conexión de Amazon VPC.

Para obtener información sobre ACL de red, consulte ACL de red en la Guía del usuario de Amazon VPC.

Consideraciones clave

Consulte los siguientes temas antes de realizar la migración a un nuevo entorno de Amazon MWAA.

Temas

- Autenticación
- Rol de ejecución

Autenticación

Amazon MWAA usa AWS Identity and Access Management (IAM) para controlar el acceso a la interfaz de usuario de Apache Airflow. Usted debe crear y administrar políticas de IAM que concedan permiso a los usuarios de Apache Airflow para acceder al servidor web y administrar los DAG. Puede gestionar la autenticación y la autorización de los roles predeterminados de Apache Airflow mediante la IAM en diferentes cuentas.

Para gestionar y restringir todavía más el acceso de los usuarios de Apache Airflow únicamente a un subconjunto de DAG de su flujo de trabajo, puede crear roles de Airflow personalizados y asignarlos a las entidades principales de IAM. Para obtener más información y un step-by-step tutorial, consulte Tutorial: Restringir el acceso de un usuario de Amazon MWAA a un subconjunto de DAG.

También puede configurar identidades federadas para acceder a Amazon MWAA. Para obtener más información, consulte los siguientes temas.

- Entorno de Amazon MWAA con acceso público: uso de Okta como proveedor de identidades con Amazon MWAA blog de computación de AWS .
- Entorno de Amazon MWAA con acceso privado: acceso a un entorno privado de Amazon MWAA mediante identidades federadas.

Rol de ejecución

Amazon MWAA utiliza una función de ejecución que concede permisos a su entorno para acceder a otros AWS servicios. Puede proporcionar a su flujo de trabajo acceso a AWS los servicios añadiendo los permisos pertinentes a la función. Si elige la opción predeterminada para crear una nueva función de ejecución cuando crea el entorno por primera vez, Amazon MWAA asigna los permisos mínimos

necesarios a la función, excepto en el caso de Logs, para los que Amazon MWAA añade todos los grupos de CloudWatch registros automáticamente.

Una vez creado el rol de ejecución, Amazon MWAA no podrá administrar sus políticas de permisos en su nombre. Para actualizar el rol de ejecución, debe editar la política que se va a añadir y eliminar permisos según sea necesario. Por ejemplo, puede integrar su entorno de Amazon MWAA con AWS Secrets Manager como backend para almacenar de forma segura los secretos y las cadenas de conexión con el fin de utilizarlos en sus flujos de trabajo de Apache Airflow. Para ello, adjunte la siguiente política de permisos al rol de ejecución de su entorno.

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

La integración con otros AWS servicios sigue un patrón similar: añades la política de permisos correspondiente a tu función de ejecución de Amazon MWAA y concedes permiso a Amazon MWAA para acceder al servicio. Para obtener más información sobre la gestión del rol de ejecución de Amazon MWAA y ver ejemplos adicionales, consulte Rol de ejecución de Amazon MWAA en la Guía del usuario de Amazon MWAA.

Migración a un nuevo entorno de Amazon MWAA

En el siguiente tema se describen los pasos para migrar la carga de trabajo de Apache Airflow existente a un nuevo entorno de Amazon MWAA. Puede seguir los siguientes pasos para migrar de una versión anterior de Amazon MWAA a una nueva versión, o bien migrar su implementación autogestionada de Apache Airflow a Amazon MWAA. En este tutorial se da por sentado que está migrando desde una versión 1.10.12 de Apache Airflow existente a una nueva Amazon MWAA con Apache Airflow v2.5.1, pero puede utilizar los mismos procedimientos para migrar desde o hacia versiones diferentes de Apache Airflow.

Temas

Requisitos previos

Para poder completar los pasos y migrar su entorno, necesitará lo siguiente:

- Una implementación de Apache Airflow. Puede ser un entorno Amazon MWAA existente o autogestionado.
- Docker debe estar instalado en el sistema operativo local.
- AWS Command Line Interface Versión 2 instalada.

Primer paso: cree un nuevo entorno Amazon MWAA que ejecute la última versión compatible de Apache Airflow

Puede crear un entorno siguiendo los pasos detallados de Introducción a Amazon MWAA en la Guía del usuario de Amazon MWAA o utilizando una plantilla. AWS CloudFormation Si va a migrar desde un entorno Amazon MWAA existente y ha utilizado una plantilla AWS CloudFormation para crear el entorno anterior, puede cambiar la propiedad `AirflowVersion` para especificar la nueva versión.

```
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
```

Como alternativa, si migra desde un entorno Amazon MWAA existente, puede copiar el siguiente script de Python que utiliza el AWSSDK para Python (Boto3) para clonar su entorno. También puede descargar el script.

Script de Python

```
# 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)
```

Paso dos: migre los recursos de su flujo de trabajo

Apache Airflow v2 es una versión principal. Si va a migrar desde Apache Airflow v1, debe preparar los recursos de su flujo de trabajo y verificar los cambios que realiza en sus DAG, requisitos y complementos. Para ello, le recomendamos configurar una versión puente de Apache Airflow en su sistema operativo local mediante Docker y el ejecutor local Amazon MWAA. El ejecutor local de Amazon MWAA proporciona una utilidad de interfaz de línea de comandos (CLI) que replica un entorno de Amazon MWAA de forma local.

Siempre que cambie las versiones de Apache Airflow, asegúrese de hacer referencia `--constraint` a la URL correcta en su `requirements.txt`.

Para migrar los recursos de su flujo de trabajo

1. Cree una bifurcación del repositorio aws-mwaa-local-runner y clone una copia del ejecutor local de Amazon MWAA.

2. Consulte la sucursal `v1.10.15`del repositorio aws-mwaa-local-runner. Apache Airflow publicó la versión 1.10.15 como versión puente para facilitar la migración a Apache Airflow v2 y, aunque Amazon MWAA no es compatible con la versión 1.10.15, puede utilizar el ejecutor local de Amazon MWAA para probar sus recursos.
3. Utilice la herramienta CLI del ejecutor local Amazon MWAA para crear la imagen de Docker y ejecutar Apache Airflow de forma local. Para obtener más información, consulte README en el repositorio GitHub.
4. Si Apache Airflow se ejecuta de forma local, siga los pasos descritos en la sección Actualización de la versión 1.10 a la versión 2 en el sitio web de documentación de Apache Airflow.

    a. Para actualizar los `requirements.txt`, siga las prácticas recomendadas que se indican en Administrar las dependencias de Python, en la Guía del usuario de Amazon MWAA.

    b. Si ha agrupado sus operadores y sensores personalizados con los complementos de su entorno Apache Airflow v1.10.12 existente, muévalos a su carpeta DAG. Para obtener más información sobre las prácticas recomendadas de administración de módulos para Apache Airflow v2+, consulte la administración de módulos en el sitio web de documentación de Apache Airflow.
5. Una vez que haya realizado los cambios necesarios en los recursos de su flujo de trabajo, consulte la sucursal `v2.5.1` del repositorio aws-mwaa-local-runner y pruebe los DAG del flujo de trabajo actualizados, los requisitos y los complementos personalizados de forma local. Si va a migrar a una versión diferente de Apache Airflow, puede utilizar la rama de ejecución local adecuada para su versión.
6. Una vez que haya probado correctamente los recursos de flujo de trabajo, copie los DAG, `requirements.txt` y los complementos en el bucket de Amazon S3 que configuró con su nuevo entorno Amazon MWAA.

Paso tres: exportar los metadatos de su entorno actual

Las tablas de metadatos de Apache Airflow como `dag`, `dag_tag` y `dag_code` se rellenan automáticamente cuando copia los archivos DAG actualizados en el bucket de Amazon S3 de su entorno y el programador los analiza. Las tablas relacionadas con los permisos también se rellenan automáticamente en función del permiso de la función de ejecución de IAM. No es necesario migrarlos.

Puede migrar los datos relacionados con el historial del DAG, `variable`, `slot_pool`, `sla_miss`, y si es necesario, `xcom`, `job`, y las tablas `log`. El registro de instancias de tareas se almacena en

CloudWatch Logs, en `airflow-{environment_name}` el grupo de registros. Si quiere ver los registros de las instancias de tareas de las ejecuciones anteriores, debe copiarlos en el nuevo grupo de registros del entorno. Le recomendamos que mueva solo los registros correspondientes a unos pocos días para reducir los costes asociados.

Si está migrando desde un entorno Amazon MWAA existente, no hay acceso directo a la base de datos de metadatos. Debe ejecutar un DAG para exportar los metadatos del entorno de Amazon MWAA existente al bucket de Amazon S3 de su elección. Los siguientes pasos también se pueden utilizar para exportar los metadatos de Apache Airflow si va a migrar desde un entorno autogestionado.

Después de exportar los datos, puede ejecutar un DAG en el nuevo entorno para importar los datos. Durante el proceso de exportación e importación, todos los demás DAG se pausan.

Para exportar los metadatos de su entorno actual

1. Cree un bucket de Amazon S3 con el AWS CLI para guardar los datos exportados. Reemplace `UUID` y `region` con su propia información.

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

> **Note**
>
> Si va a migrar datos confidenciales, como las conexiones que almacena en variables, le recomendamos que habilite el cifrado predeterminado para el bucket de Amazon S3.

2.

> **Note**
>
> No se aplica a la migración desde un entorno autogestionado.

Modifique la función de ejecución del entorno existente y añada la siguiente política para conceder acceso de escritura al depósito que creó en el primer paso.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

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

3. Clone el repositorio amazon-mwaa-examples y navegue hasta el subdirectorio `metadata-migration` correspondiente a su escenario de migración.

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

4. En `export_data.py`, sustituya el valor de cadena para `S3_BUCKET` con el bucket de Amazon S3 que creó para almacenar los metadatos exportados.

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

5. Ubique el archivo `requirements.txt` en el directorio `metadata-migration`. Si ya tiene un archivo de requisitos para su entorno actual, añada los requisitos adicionales especificados en el archivo `requirements.txt`. Si no tiene un archivo de requisitos existente, simplemente puede usar el que se proporciona en el directorio `metadata-migration`.
6. Copie `export_data.py` en el directorio DAG el bucket de Amazon S3 asociado con el entorno actual. Si va a migrar desde un entorno autogestionado, copie `export_data.py` a su carpeta `/dags`.
7. Copie la actualización de `requirements.txt` en el bucket de Amazon S3 asociado a su entorno actual y, a continuación, edite el entorno para especificar la nueva versión `requirements.txt`.
8. Una vez actualizado el entorno, acceda a la interfaz de usuario de Apache Airflow, anule la pausa del DAG `db_export` y active la ejecución del flujo de trabajo.
9. Compruebe que los metadatos se exportan al `data/migration/existing-version_to_new-version/export/` en el bucket de Amazon S3 `mwaa-migration-{UUID}`, con cada tabla en su propio archivo dedicado.

Paso cuatro: importar los metadatos a su nuevo entorno

Importar los metadatos a su nuevo entorno

1. En `import_data.py`, sustituya los valores de cadena de los siguientes valores por su información.

   - Para la migración desde un entorno Amazon MWAA existente:

   ```
   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` controla el número de días de archivos de registro que el flujo de trabajo copia al nuevo entorno.

   - Para migrar desde un entorno autoadministrado:

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

2. (Opcional) `import_data.py` copia solo los registros de tareas fallidas. Si desea copiar todos los registros de tareas, modifique la función `getDagTasks` y elimine `ti.state = 'failed'` como se muestra en el siguiente fragmento de código.

   ```
   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. Modifique la función de ejecución de su nuevo entorno y añada la siguiente política. La política de permisos permite a Amazon MWAA leer del bucket de Amazon S3 al que exportaron los metadatos de Apache Airflow y copiar los registros de instancias de tareas de los grupos de registros existentes. Sustituya todos los marcadores de posición por su información.

> **Note**
>
> Si va a migrar desde un entorno autogestionado, debe eliminar de la política los permisos relacionados con CloudWatch Logs.

```
{
    "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. Copie `import_data.py` en el directorio DAG del bucket de Amazon S3 asociado a su nuevo entorno y, a continuación, acceda a la interfaz de usuario de Apache Airflow para detener el DAG `db_import` y active el flujo de trabajo. El nuevo DAG aparecerá en la interfaz de usuario de Apache Airflow en unos minutos.
5. Una vez finalizada la ejecución del DAG, compruebe que el historial de ejecuciones del DAG esté copiado accediendo a cada DAG individual.

Pasos siguientes

- Para obtener más información sobre las clases y capacidades del entorno Amazon MWAA disponibles, consulte la clase de entorno Amazon MWAA en la Guía del usuario de Amazon MWAA.
- Para obtener más información sobre cómo Amazon MWAA gestiona el escalado automático de los trabajadores, consulte el escalado automático de Amazon MWAA en la Guía del usuario de Amazon MWAA.
- Para obtener más información acerca de la API de REST de Amazon MWAA, consulte la API de REST de Amazon MWAA.

Recursos relacionados

- Modelos de Apache Airflow (documentación de Apache Airflow): obtenga más información sobre los modelos de bases de datos de metadatos de Apache Airflow.

Migración de cargas de trabajo de AWS Data Pipeline a Amazon MWAA

AWS lanzó el servicio AWS Data Pipeline en 2012. En ese momento, los clientes necesitaban un servicio que les permitiera usar distintas opciones informáticas para mover datos entre diferentes orígenes de datos. A medida que las necesidades de transferencia de datos han cambiado con el paso del tiempo, también lo han hecho las soluciones para esas necesidades. Ahora tiene la opción de elegir la solución que mejor se adapte a los requisitos de su empresa. Puede migrar sus cargas de trabajo a cualquiera de los siguientes servicios de AWS:

- Utilice Amazon Managed Workflows para Apache Airflow (Amazon MWAA) para gestionar la organización de flujos de trabajo de Apache Airflow.
- Use Step Functions para organizar flujos de trabajo entre varios Servicios de AWS.
- Utilice AWS Glue para ejecutar y organizar las aplicaciones de Apache Spark.

La opción que elija depende de su carga de trabajo actual en AWS Data Pipeline. En este tema se explica cómo llevar a cabo migraciones de AWS Data Pipeline a Amazon MWAA.

Temas

Elección de Amazon MWAA

Amazon Managed Workflows para Apache Airflow (Amazon MWAA) es un servicio de orquestación administrado para Apache Airflow que facilita la configuración y el funcionamiento de canalizaciones de datos integrales en la nube a escala. Apache Airflow es una herramienta de código abierto que se utiliza para crear, programar y supervisar secuencias de procesos y tareas denominadas flujos de trabajo mediante programación. Con Amazon MWAA, puede usar el lenguaje de programación Airflow y Python para crear flujos de trabajo sin tener que administrar la infraestructura subyacente para garantizar la escalabilidad, la disponibilidad y la seguridad. Amazon MWAA escala

automáticamente su capacidad de ejecución del flujo de trabajo para adaptarla a sus necesidades y está integrado con los servicios de seguridad AWS para proporcionarle un acceso rápido y seguro a sus datos.

A continuación, se destacan algunos de los beneficios de llevar a cabo migraciones de AWS Data Pipeline a Amazon MWAA:

- Escalabilidad y rendimiento mejorados: Amazon MWAA ofrece un marco flexible y escalable para definir y ejecutar flujos de trabajo. Esto permite a los usuarios gestionar flujos de trabajo grandes y complejos con facilidad y sacar partido a características como la programación dinámica de tareas, los flujos de trabajo basados en datos y el paralelismo.
- Supervisión y registro mejorados: Amazon MWAA se integra con Amazon CloudWatch para mejorar la supervisión y el registro de los flujos de trabajo. Amazon MWAA envía automáticamente las métricas y los registros del sistema a CloudWatch. Esto significa que puede realizar un seguimiento del progreso y el rendimiento de los flujos de trabajo en tiempo real e identificar cualquier problema que surja.
- Mejores integraciones con servicios de AWS y software de terceros: Amazon MWAA se integra con otros servicios de AWS, como Amazon S3 y Amazon Redshift, AWS Glue, así como con software de terceros, como DBT, Snowflake y Databricks. Esto permite procesar y transferir datos entre distintos entornos y servicios.
- Herramienta de canalización de datos de código abierto: Amazon MWAA utiliza el mismo producto Apache Airflow de código abierto con el que está familiarizado. Apache Airflow es una herramienta diseñada especialmente para gestionar todos los aspectos de la gestión de la canalización de datos, incluida la incorporación, el procesamiento, la transferencia, las pruebas de integridad y los controles de calidad, además de garantizar el linaje de los datos.
- Arquitectura moderna y flexible: Amazon MWAA utiliza la organización en contenedores y las tecnologías sin servidor nativas en la nube. Esto se traduce en una mayor flexibilidad y portabilidad, así como en una implementación y administración más sencillas de los entornos de flujo de trabajo.

Mapeo conceptual y de arquitectura

AWS Data Pipeline y Amazon MWAA tienen arquitecturas y componentes diferentes, lo que puede afectar al proceso de migración y a la forma en que se definen y ejecutan los flujos de trabajo. En esta sección se expone una descripción general de la arquitectura y los componentes de ambos servicios y se destacan algunas de las diferencias principales.

Tanto AWS Data Pipeline como Amazon MWAA como son servicios totalmente administrados. Cuando migre sus cargas de trabajo a Amazon MWAA, es posible que tenga que aprender nuevos conceptos para modelar sus flujos de trabajo existentes con Apache Airflow. Sin embargo, no tendrá que administrar la infraestructura, parchear a los trabajadores ni administrar las actualizaciones del sistema operativo.

En la siguiente tabla, se asocian los conceptos clave de AWS Data Pipeline con los de Amazon MWAA. Utilice esta información como punto de partida para diseñar un plan de migración.

Concepto	AWS Data Pipeline	Amazon MWAA
Definición de la canalización	AWS Data Pipeline utiliza un archivo de configuración basado en JSON que define el flujo de trabajo.	Amazon MWAA utiliza gráficos acíclicos dirigidos (DAG) basados en Python que definen el flujo de trabajo.
Entorno de ejecución de canalizaciones	Los flujos de trabajo se ejecuten en instancias de Amazon EC2. AWS Data Pipeline proporciona y administra estas instancias en su nombre.	Amazon MWAA utiliza entornos organizados en contenedores de Amazon ECS para ejecutar tareas.
Componentes de canalización	Las actividades son tareas de procesamiento que se ejecutan como parte del flujo de trabajo.	Los operadores (tareas) son las unidades de procesami ento fundamentales de un flujo de trabajo.
	Las precondiciones contienen instrucciones condicionales que deben cumplirse antes de que una actividad pueda ejecutarse.	Los sensores (tareas) son instrucciones condicionales que pueden esperar a que se complete un recurso o una tarea antes de ejecutarse.
	Un recurso en AWS Data Pipeline se refiere a es el recurso informático AWS que realiza el trabajo que	Con las tareas de un DAG, puede definir distintos recursos informáticos, incluidos Amazon ECS,

Concepto	AWS Data Pipeline	Amazon MWAA
	una actividad de canalizac ión especifica. Amazon EC2 y Amazon EMR son dos recursos disponibles.	Amazon EMR y Amazon EKS. Amazon MWAA ejecuta operaciones de Python en trabajadores que se ejecutan en Amazon ECS.
Ejecución de canalizaciones	AWS Data Pipeline admite la programación de ejecucion es con patrones regulares basados en tasas y expresion es cron.	Amazon MWAA admite la programación con expresion es cron y ajustes preestabl ecidos, así como con horarios personalizados.
	Una instancia hacer referenci a a cada ejecución de la canalización.	Una ejecución de DAG hace referencia a cada ejecución de un flujo de trabajo de Apache Airflow.
	Un intento se refiere a un reintento de una operación que ha dado error.	Amazon MWAA admite los reintentos que usted defina a nivel de DAG o a nivel de tarea.

Despliegue de ejemplo

En muchos casos, podrá reutilizar los recursos que esté orquestando con AWS Data Pipeline tras migrar a Amazon MWAA. La siguiente lista contiene ejemplos de implementaciones que utilizan Amazon MWAA para los casos de uso de AWS Data Pipeline más comunes.

- Ejecución de un trabajo de Amazon EMR (taller de AWS)
- Creación de un complemento personalizado para Apache Hive y Hadoop (Guía del usuario de Amazon MWAA)
- Cómo copiar datos de S3 a Redshift (taller de AWS)
- Ejecución de un script de shell en una instancia remota de Amazon ECS (Guía del usuario de Amazon MWAA)

- Orquestación de flujos de trabajo híbridos (locales) (entrada del blog)

Para ver ejemplos, consulte los siguientes tutoriales:

- Tutoriales de Amazon MWAA
- Ejemplos de código de Amazon MWAA

Comparación de precios

El precio de AWS Data Pipeline se basa en la cantidad de canalizaciones, así como en el uso de cada canalización. Las actividades que lleve a cabo más de una vez al día (frecuencia alta) cuestan 1 USD al mes por actividad. Las actividades que lleve a cabo una vez al día (frecuencia baja) cuestan 0.60 USD al mes por actividad. Las canalizaciones inactivas tienen un precio de 1 USD por canalización. Para obtener más información, consulte la página de precios de AWS Data Pipeline.

Los precios de Amazon MWAA se basan en el tiempo durante el que exista su entorno administrado de Apache Airflow y en cualquier escalado automático adicional necesario para proporcionar más trabajadores o aumentar la capacidad de programadores. El uso del entorno de Amazon MWAA se paga por hora (se factura con una resolución de un segundo) y las tarifas varían en función del tamaño del entorno. Amazon MWAA escala automáticamente el número de trabajadores en función de la configuración del entorno. AWS calcula el coste de los trabajadores adicionales por separado. Para obtener más información sobre el coste por hora de utilizar entornos de Amazon MWAA de distintos tamaños, consulte la página de precios de Amazon MWAA.

Recursos relacionados

Para obtener más información y descubrir prácticas recomendadas para usar Amazon MWAA, consulte los siguientes recursos:

- La referencia de la API de Amazon MSK
- Monitorización de paneles y alarmas en Amazon MWAA
- Ajuste del desempeño de Apache Airflow en Amazon MWAA

Historial de documentos de Amazon MWAA

En la siguiente tabla se describen los cambios importantes de la guía de migración a Amazon MWAA, a partir de marzo de 2022.

Cambio	Descripción	Fecha
Nuevo tema sobre la migración de cargas de trabajo de AWS Data Pipeline a Amazon MWAA	Se añadió nueva información y orientación sobre la migración de las cargas de trabajo existentes de AWS Data Pipeline a Amazon MWAA. Utilice esta información como ayuda para diseñar un plan de migración. • Migración de cargas de trabajo de AWS Data Pipeline a Amazon MWAA	14 de abril de 2023
Lanzamiento de la guía de migración a Amazon MWAA	Amazon MWAA ahora ofrece una guía detallada sobre la migración a un nuevo entorno de Amazon MWAA. Los pasos descritos en la guía de migración a Amazon MWAA se aplican a la migración desde un entorno de Amazon MWAA existente o desde una implementación autogestionada de Apache Airflow. • Acerca de la guía de migración a Amazon MWAA	7 de marzo de 2022

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.