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

Explore la arquitectura MWAA de red de Amazon

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

Temas

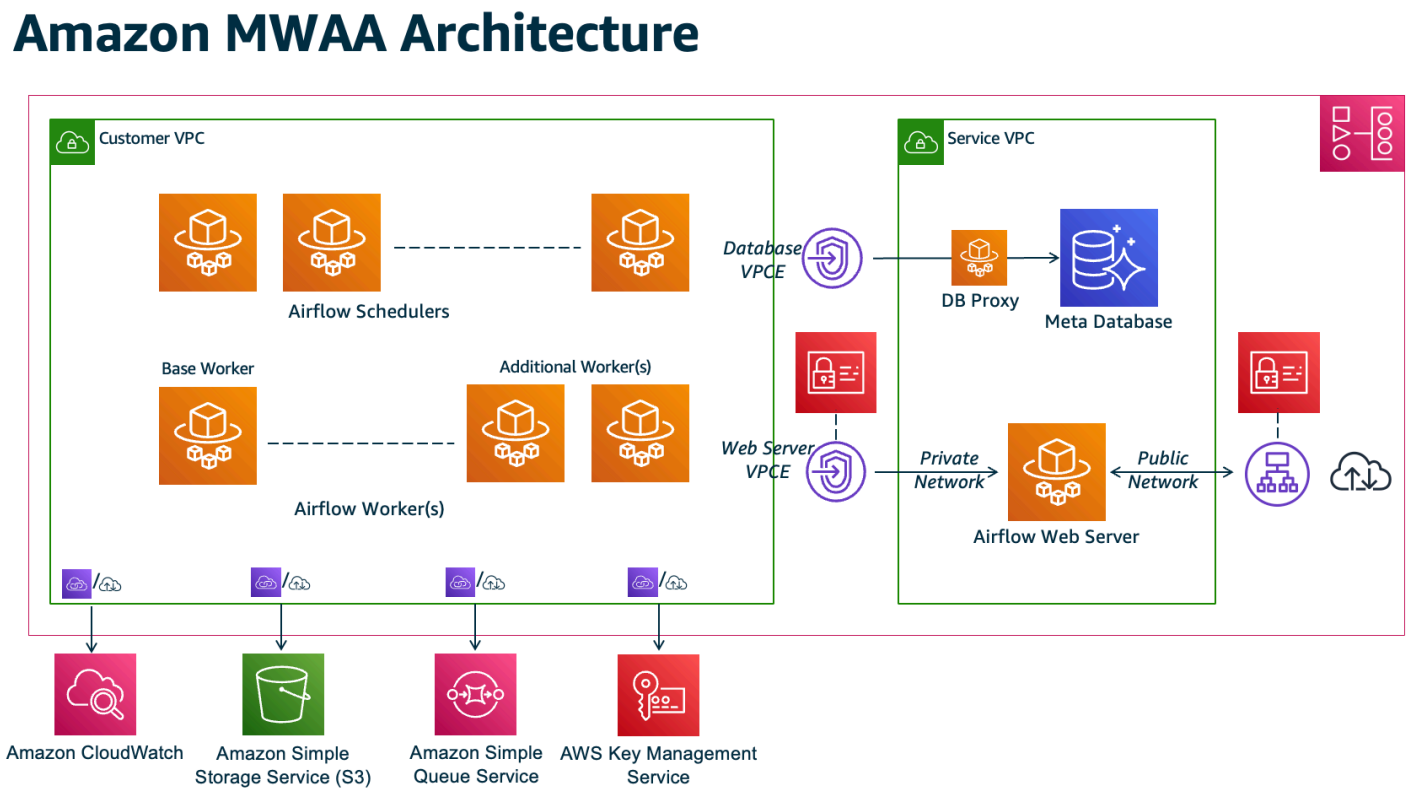
MWAAComponentes de Amazon

Los MWAA entornos de Amazon constan de los cuatro componentes principales siguientes:

1. Planificador: analiza y monitorea todas sus DAGs tareas y las pone en cola para su ejecución cuando se cumplen DAG las dependencias. Amazon MWAA implementa el programador como un AWS Fargate clúster 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 MWAA entorno de Amazon, consulte Amazon MWAA Environment Class.
2. Trabajadores: una o varias tareas de Fargate que ejecutan las tareas programadas. La cantidad de procesos de trabajo del entorno viene determinada por un rango entre el número mínimo y el máximo que especifique. Amazon MWAA comienza a autoescalar los trabajadores cuando el número de tareas en cola y en ejecución es mayor de lo que sus trabajadores actuales pueden gestionar. Cuando las tareas en ejecución y en cola suman cero durante más de dos minutos, MWAA Amazon reduce el número de trabajadores al mínimo. Para obtener más información sobre cómo MWAA gestiona Amazon a los trabajadores que se autoescalan, consulta Amazon MWAA Automatic Scaling.
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 defina en AWS Identity and Access Management ()IAM. Para obtener más información sobre la configuración de las políticas de IAM acceso para su entorno, consulte Acceder a un MWAA entorno de Amazon.
4. Base de datos: almacena metadatos sobre el entorno de Apache Airflow y sus flujos de trabajo, incluido el historial de DAG ejecución. La base de datos es una SQL base de datos Aurora Postgre de un solo inquilino administrada por AWSScheduler y los contenedores Fargate de

Workers y a la que pueden acceder a ellos a través de un punto final de Amazon con seguridad privada. VPC

Cada MWAA entorno de Amazon también interactúa con un conjunto de AWS servicios para gestionar una variedad de tareas, como el almacenamiento y el acceso DAGs y las dependencias de las tareas, la protección de los datos en reposo y el registro y la supervisión de su entorno. El siguiente diagrama muestra los distintos componentes de un MWAA entorno de Amazon.



> **Note**
>
> El servicio Amazon no VPC es compartidoVPC. Amazon MWAA crea un AWS entorno VPC propio para cada entorno que crees.

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

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

Conectividad

Su MWAA entorno de Amazon necesita acceder a todos los AWS servicios con los que se integra. La función de MWAA ejecución de Amazon controla la forma en que se concede el acceso MWAA a Amazon para conectarse a otros AWS servicios en tu nombre. Para la conectividad de red, puedes proporcionar acceso público a Internet a tu Amazon VPC o crear VPC puntos de conexión de Amazon. Para obtener más información sobre la configuración de los VPC puntos de enlace de Amazon (AWS PrivateLink) para su entorno, consulte Administrar el acceso a los VPC puntos de enlace MWAA en Amazon en la Guía MWAAdel usuario de Amazon.

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

Cuando configuras Apache Airflow en modo privado, tu Amazon solo puede acceder a la interfaz de usuario de Apache Airflow a través de los puntos de conexión de VPC AmazonVPC.

Para obtener más información sobre las redes, consulte Redes en la Guía del MWAA usuario de Amazon.

Consideraciones clave para migrar a un nuevo entorno MWAA

Obtenga más información sobre las consideraciones clave, como la autenticación y la función de MWAA ejecución de Amazon, al planificar la migración de sus cargas de trabajo de Apache Airflow a 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. Debe crear y administrar IAM políticas que otorguen a sus usuarios de Apache Airflow permiso para acceder al servidor web y administrarlo. DAGs Puede gestionar la autenticación y la autorización de las funciones predeterminadas de Apache Airflow IAM en diferentes cuentas.

Puede administrar y restringir aún más el acceso de los usuarios de Apache Airflow a un subconjunto de su flujo de trabajo DAGs creando roles de Airflow personalizados y asignándolos a sus directores. IAM Para obtener más información y un step-by-step tutorial, consulte Tutorial: Restringir el acceso de un MWAA usuario de Amazon a un subconjunto de DAGs.

También puedes configurar identidades federadas para acceder a AmazonMWAA. Para obtener más información, consulte los siguientes temas.

- MWAAEntorno de Amazon con acceso público: uso de Okta como proveedor de identidad con Amazon MWAA on the AWS Compute Blog.
- MWAAEntorno de Amazon con acceso privado: acceso a un MWAA entorno de Amazon privado 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 al rol. Si elige la opción predeterminada para crear un nuevo rol de ejecución cuando crea el entorno por primera vez, Amazon le MWAA asigna los permisos mínimos necesarios, excepto en el caso de CloudWatch Logs, para los que Amazon MWAA agrega todos los grupos de registros automáticamente.

Una vez creada la función de ejecución, Amazon MWAA no podrá gestionar sus políticas de permisos en tu 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 MWAA entorno de Amazon AWS Secrets Manager como backend para almacenar de forma segura los secretos y las cadenas de conexión para 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 MWAA ejecución de Amazon y concedes permiso MWAA a Amazon para acceder al servicio. Para obtener más información sobre la gestión de la función de MWAA

ejecución de Amazon y ver ejemplos adicionales, consulta la función de MWAA ejecución de Amazon en la Guía del MWAA usuario de Amazon.

Migre a un nuevo MWAA entorno de Amazon

Explore los siguientes pasos para migrar su carga de trabajo actual de Apache Airflow a un nuevo MWAA entorno de Amazon. Puedes seguir estos pasos para migrar de una versión anterior de Amazon MWAA a una versión nueva o migrar tu implementación autogestionada de Apache Airflow a Amazon. MWAA En este tutorial se asume que está migrando de un Apache Airflow v1.10.12 existente a un nuevo Amazon que MWAA ejecute Apache Airflow v2.5.1, pero puede utilizar los mismos procedimientos para migrar desde o hacia diferentes versiones 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 MWAA entorno Amazon autogestionado o existente.
- Docker debe estar instalado en el sistema operativo local.
- AWS Command Line Interface versión 2 instalada.

Paso uno: Cree un nuevo MWAA entorno de Amazon que ejecute la última versión compatible de Apache Airflow

Puedes crear un entorno siguiendo los pasos detallados en Cómo empezar a usar Amazon MWAA en la Guía del MWAA usuario de Amazon o mediante una AWS CloudFormation plantilla. Si vas

a migrar desde un MWAA entorno de Amazon existente y has utilizado una AWS CloudFormation plantilla para crear tu entorno anterior, puedes cambiar la `AirflowVersion` propiedad 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 MWAA entorno de Amazon existente, puede copiar el siguiente script de Python que utiliza Python (Boto3)AWS SDK 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 DAGs requisitos y complementos. Para ello, te recomendamos configurar una versión puente de Apache Airflow en tu sistema operativo local mediante Docker y el ejecutor MWAAlocal de Amazon. El ejecutor MWAA local de Amazon proporciona una utilidad de interfaz de línea de comandos (CLI) que replica un MWAA entorno de Amazon de forma local.

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

Migración de los recursos de su flujo de trabajo

1. Crea una bifurcación del aws-mwaa-local-runnerrepositorio y clona una copia del ejecutor MWAA local de Amazon.
2. Consulta la `v1.10.15` rama 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 Amazon local runner para probar sus recursos. MWAA

3. Utilice la CLI herramienta Amazon MWAA local Runner para crear la imagen de Docker y ejecutar Apache Airflow de forma local. Para obtener más información, consulta el ejecutor local READMEen el GitHub repositorio.
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 actualizarlo`requirements.txt`, sigue las prácticas recomendadas que se recomiendan en Gestión de las dependencias de Python, en la Guía del MWAA usuario de Amazon.
   b. Si ha agrupado sus operadores y sensores personalizados con sus complementos para 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 `v2.5.1` rama del repositorio aws-mwaa-local -runner y pruebe localmente el flujo de trabajo actualizadoDAGs, los requisitos y los complementos personalizados. 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 sus recursos de flujo de trabajo, copie sus DAGs complementos y los suyos en el bucket de Amazon S3 que configuró con su nuevo MWAA entorno de Amazon. `requirements.txt`

Paso tres: exportar los metadatos de su entorno actual

Las tablas de metadatos de Apache Airflow`dag`, como`dag_tag`, y, `dag_code` se rellenan automáticamente cuando copia los DAG archivos 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. IAM No es necesario migrarlos.

Puede migrar los datos relacionados con el DAG historial`variable`,`slot_pool`, `ysla_miss`, si es necesario, `xcomjob`, y `log` las tablas. El registro de instancias de tareas se almacena en los CloudWatch registros del grupo de `airflow-`*`{environment_name}`* 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 MWAA entorno Amazon existente, no hay acceso directo a la base de datos de metadatos. Debe ejecutar DAG a para exportar los metadatos de su MWAA entorno de Amazon existente a un 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.

Una vez exportados los datos, puede ejecutar un DAG en su nuevo entorno para importarlos. Durante el proceso de exportación e importación, todos los demás DAGs se detienen.

Exportación de los metadatos de su entorno actual

1. Cree un bucket de Amazon S3 con el AWS CLI para almacenar 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 el rol de ejecución del entorno existente y añada la siguiente política para conceder acceso de escritura al bucket 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 amazon-mwaa-examplesrepositorio y navegue hasta el `metadata-migration` subdirectorio 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. `export_data.py`Cópielo en el DAG directorio del bucket de Amazon S3 asociado a su 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 y active el `db_export` DAG flujo de trabajo para que se ejecute.
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

Importación de los metadatos a su nuevo entorno

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

   - Para migrar desde un MWAA entorno Amazon 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 el rol de ejecución de su nuevo entorno y añada la siguiente política. La política de permisos permite MWAA a Amazon leer desde el depósito de Amazon S3 al que exportó 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 los CloudWatch registros.

```
{
    "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. `import_data.py`Cópielo en el DAG directorio del bucket de Amazon S3 asociado a su nuevo entorno y, a continuación, acceda a la interfaz de usuario de Apache Airflow para reanudar la pausa `db_import` DAG y activar el flujo de trabajo. Lo nuevo DAG aparecerá en la interfaz de usuario de Apache Airflow en unos minutos.
5. Una vez finalizada la DAG ejecución, compruebe que el historial de DAG ejecución se haya copiado accediendo a cada DAG uno de ellos.

Siguientes pasos

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

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.

Migre las cargas de trabajo desde AWS Data Pipeline Amazon MWAA

AWS lanzó el AWS Data Pipeline servicio 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 cambiaban con el tiempo, también lo han hecho las soluciones para esas necesidades. Ahora tiene la opción de elegir la solución que mejor se adapte a sus requisitos empresariales. Puede migrar sus cargas de trabajo a cualquiera de los siguientes AWS servicios:

- Utilice Amazon Managed Workflows for Apache Airflow (AmazonMWAA) para gestionar la organización del flujo de trabajo de Apache Airflow.
- Usar Step Functions para orquestar flujos de trabajo entre varios Servicios de AWS.
- Se utiliza 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 migrar de AWS Data Pipeline a AmazonMWAA.

Temas

Elegir Amazon MWAA

Amazon Managed Workflows for Apache Airflow (AmazonMWAA) es un servicio de organización gestionado para Apache Airflow que le permite configurar y operar canalizaciones de end-to-end datos 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 AmazonMWAA, puede utilizar Apache Airflow y el lenguaje de programación Python para crear flujos de trabajo sin tener que gestionar la infraestructura

subyacente para garantizar la escalabilidad, la disponibilidad y la seguridad. Amazon escala MWAA automáticamente la capacidad de su flujo de trabajo para satisfacer tus necesidades y está integrado con los servicios de AWS seguridad para ayudarte a proporcionarte un acceso rápido y seguro a tus datos.

A continuación, se destacan algunos de los beneficios de migrar de AWS Data Pipeline a AmazonMWAA:

- Escalabilidad y rendimiento mejorados: Amazon MWAA proporciona 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 sus flujos de trabajo. Amazon envía MWAA 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 AWS servicios y software de terceros: Amazon MWAA se integra con una variedad de otros AWS servicios, como Amazon S3 y Amazon Redshift AWS Glue, así como con software de terceros, como Snowflake y DBTDatabricks. 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 aprovecha la contenedorización y las tecnologías sin servidor nativas de 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.

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

La siguiente tabla asocia los conceptos clave AWS Data Pipeline con los de AmazonMWAA. 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 JSON basado en el que se define el flujo de trabajo.	Amazon MWAA utiliza gráficos acíclicos dirigidos (DAGs) basados en Python que definen el flujo de trabajo.
Entorno de ejecución de canalizaciones	Los flujos de trabajo se ejecutan en EC2 instancias de Amazon. AWS Data Pipeline aprovisiona y gestiona estas instancias en tu nombre.	Amazon MWAA utiliza los entornos ECS contenerizados de Amazon para ejecutar las 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 al recurso AWS informático que realiza el trabajo que especifica una actividad de canalización.	Al utilizar las tareas de unDAG, puede definir una variedad de recursos informáticos, incluidos Amazon ECSEMR, Amazon

Concepto	AWS Data Pipeline	Amazon MWAA
	Amazon EC2 y Amazon EMR son dos recursos disponibles.	y AmazonEKS. Amazon MWAA ejecuta operaciones de Python en trabajadores que se ejecutan en AmazonECS.
Ejecución de canalizaciones	AWS Data Pipeline admite la programación de ejecucion es con patrones regulares basados en tasas y cronos.	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 DAGejecución se refiere 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 DAG nivel o a nivel de tarea.

Implementaciones de ejemplo

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

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

Para ver ejemplos, consulte los siguientes tutoriales:

- MWAATutoriales de Amazon
- Ejemplos de MWAA código de Amazon

Comparación de precios

AWS Data Pipeline El precio 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 que dure su entorno gestionado de Apache Airflow y en cualquier escalado automático adicional que se requiera para proporcionar más trabajadores o capacidad de planificador. Usted paga por el uso de su MWAA entorno de Amazon por hora (se factura con una resolución de un segundo), con tarifas que varían según el 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 varios tamaños de MWAA entorno de Amazon, consulta la página de MWAAprecios de Amazon.

Recursos relacionados

Para obtener más información y prácticas recomendadas para usar AmazonMWAA, consulta los siguientes recursos:

- La MWAA API referencia de Amazon
- Monitorización de cuadros de mando y alarmas en Amazon MWAA
- Ajuste del rendimiento 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 informaci ón y orientación sobre la migración de las cargas de trabajo existentes de AWS Data Pipeline a Amazon MWAA. Utilice esta informaci ón como ayuda para diseñar un plan de migración. • Migre las cargas de trabajo desde AWS Data Pipeline 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 autogesti onada 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.