aws

Manuel de migration

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: Manuel de migration

Table of Contents

Qu'est-ce que le guide de migration Amazon MWAA ?

Amazon Managed Workflows for Apache Airflow est un service d'orchestration géré pour Apache Airflow qui vous permet d'exploiter des pipelines de données dans le cloud à grande échelle. Amazon MWAA gère le provisionnement et la maintenance continue d'Apache Airflow afin que vous n'ayez plus à vous soucier de l'application des correctifs, du dimensionnement ou de la sécurisation des instances.

Amazon MWAA adapte automatiquement les ressources de calcul qui exécutent les tâches afin de fournir des performances constantes à la demande. Amazon MWAA sécurise vos données par défaut. Vos charges de travail s'exécutent dans votre propre environnement cloud isolé et sécurisé à l'aide d'Amazon Virtual Private Cloud. Cela garantit que les données sont automatiquement cryptées à l'aide de AWS Key Management Service.

Utilisez ce guide pour migrer vos flux de travail Apache Airflow autogérés vers Amazon MWAA, ou pour mettre à niveau un environnement Amazon MWAA existant vers une nouvelle version d'Apache Airflow. Le didacticiel de migration décrit comment créer ou cloner un nouvel environnement Amazon MWAA, migrer les ressources de votre flux de travail et transférer les métadonnées et les journaux de votre flux de travail vers votre nouvel environnement.

Avant de suivre le didacticiel de migration, nous vous recommandons de consulter les rubriques suivantes.

- Architecture réseau
- Considérations clés

Découvrez l'architecture réseau Amazon MWAA

La section suivante décrit les principaux composants d'un environnement Amazon MWAA, ainsi que l'ensemble des AWS services auxquels chaque environnement s'intègre pour gérer ses ressources, garantir la sécurité de vos données et assurer la surveillance et la visibilité de vos flux de travail.

Rubriques

Composants Amazon MWAA

Les environnements Amazon MWAA se composent des quatre composants principaux suivants :

1. Planificateur : analyse et surveille toutes vos DAGs tâches et les met en file d'attente pour exécution lorsque les dépendances d'un DAG sont satisfaites. Amazon MWAA déploie le planificateur sous la forme d'un AWS Fargate cluster avec un minimum de 2 planificateurs. Vous pouvez augmenter le nombre de planificateurs jusqu'à cinq, en fonction de votre charge de travail. Pour plus d'informations sur les classes d'environnement Amazon MWAA, reportez-vous à la section Classe d'environnement Amazon MWAA.
2. Travailleurs : une ou plusieurs tâches Fargate qui exécutent vos tâches planifiées. Le nombre de travailleurs pour votre environnement est déterminé par une plage comprise entre le nombre minimum et maximum que vous spécifiez. Amazon MWAA lance l'auto-scaling des travailleurs lorsque le nombre de tâches en attente et en cours d'exécution est supérieur à ce que vos employés actuels peuvent gérer. Lorsque la somme des tâches en cours d'exécution et en file d'attente est nulle pendant plus de deux minutes, Amazon MWAA réduit le nombre de travailleurs à son minimum. Pour plus d'informations sur la façon dont Amazon MWAA gère le dimensionnement automatique des travailleurs, consultez Amazon MWAA automatic scaling.
3. Serveur Web : exécute l'interface utilisateur Web d'Apache Airflow. Vous pouvez configurer le serveur Web avec un accès réseau privé ou public. Dans les deux cas, l'accès à vos utilisateurs d'Apache Airflow est contrôlé par la politique de contrôle d'accès que vous définissez dans Gestion des identités et des accès AWS (IAM). Pour plus d'informations sur la configuration des politiques d'accès IAM pour votre environnement, consultez la section Accès à un environnement Amazon MWAA.

4. Base de données — Stocke les métadonnées relatives à l'environnement Apache Airflow et à vos flux de travail, y compris l'historique des exécutions du DAG. La base de données est une base de données Aurora PostgreSQL à locataire unique gérée et accessible AWS par le planificateur et les conteneurs Fargate de travail via un point de terminaison Amazon VPC sécurisé de manière privée.

Chaque environnement Amazon MWAA interagit également avec un ensemble de AWS services pour gérer diverses tâches, notamment le stockage, l'accès DAGs et les dépendances entre les tâches, la sécurisation de vos données au repos, ainsi que la journalisation et la surveillance de votre environnement. Le schéma suivant illustre les différents composants d'un environnement Amazon MWAA.

> **Note**
>
> Le service Amazon VPC n'est pas un VPC partagé. Amazon MWAA crée un VPC AWS propriétaire pour chaque environnement que vous créez.

- Amazon S3 — Amazon MWAA stocke toutes les ressources de votre flux de travail DAGs, telles que les exigences et les fichiers de plug-in dans un compartiment Amazon S3. Pour plus d'informations sur la création du compartiment dans le cadre de la création de l'environnement et sur le téléchargement de vos ressources Amazon MWAA, consultez la section Créer un compartiment Amazon S3 pour Amazon MWAA dans le guide de l'utilisateur Amazon MWAA.
- Amazon SQS — Amazon MWAA utilise Amazon SQS pour mettre en file d'attente vos tâches de flux de travail avec un exécuteur Celery.
- Amazon ECR — Amazon ECR héberge toutes les images Apache Airflow. Amazon MWAA prend uniquement en charge les AWS images Apache Airflow gérées.
- AWS KMS— Amazon MWAA les utilise AWS KMS pour garantir la sécurité de vos données au repos. Par défaut, Amazon MWAA utilise des AWS KMS clés AWS gérées, mais vous pouvez configurer votre environnement pour utiliser votre propre clé gérée par le client AWS KMS . Pour plus d'informations sur l'utilisation de votre propre AWS KMS clé gérée par le client, reportez-vous à la section Clés gérées par le client pour le chiffrement des données dans le guide de l'utilisateur Amazon MWAA.

- CloudWatch— Amazon MWAA s'intègre à Apache Airflow CloudWatch et fournit des journaux et des indicateurs CloudWatch environnementaux, ce qui vous permet de surveiller vos ressources Amazon MWAA et de résoudre les problèmes.

Connectivité

Votre environnement Amazon MWAA doit accéder à tous les AWS services auxquels il s'intègre. Le rôle d'exécution Amazon MWAA contrôle la manière dont l'accès est accordé à Amazon MWAA pour se connecter à d'autres AWS services en votre nom. Pour la connectivité réseau, vous pouvez fournir un accès Internet public à votre Amazon VPC ou créer des points de terminaison Amazon VPC. Pour plus d'informations sur la configuration des points de terminaison Amazon VPC (AWS PrivateLink) pour votre environnement, consultez la section Gestion de l'accès aux points de terminaison VPC sur Amazon MWAA dans le guide de l'utilisateur Amazon MWAA.

Amazon MWAA installe les exigences sur le planificateur et le programme de travail. Si vos besoins proviennent d'un PyPiréférentiel public, votre environnement doit être connecté à Internet pour télécharger les bibliothèques requises. Pour les environnements privés, vous pouvez soit utiliser un PyPi dépôt privé, soit regrouper les bibliothèques dans des `.whl`fichiers sous forme de plugins personnalisés pour votre environnement.

Lorsque vous configurez Apache Airflow en mode privé, l'interface utilisateur d'Apache Airflow n'est accessible à votre Amazon VPC que via les points de terminaison Amazon VPC.

Pour plus d'informations sur la mise en réseau, reportez-vous à la section Mise en réseau du guide de l'utilisateur Amazon MWAA.

Considérations clés relatives à la migration vers un nouvel environnement MWAA

Apprenez-en davantage sur les principales considérations, telles que l'authentification et le rôle d'exécution Amazon MWAA, lorsque vous planifiez de migrer vos charges de travail Apache Airflow vers Amazon MWAA.

Rubriques

Authentification

Amazon MWAA utilise Gestion des identités et des accès AWS (IAM) pour contrôler l'accès à l'interface utilisateur d'Apache Airflow. Vous devez créer et gérer des politiques IAM qui accordent à vos utilisateurs Apache Airflow l'autorisation d'accéder au serveur Web et de le gérer. DAGs Vous pouvez gérer à la fois l'authentification et l'autorisation pour les rôles par défaut d'Apache Airflow à l'aide d'IAM sur différents comptes.

Vous pouvez également gérer et restreindre l'accès des utilisateurs d'Apache Airflow à un sous-ensemble de votre flux de travail uniquement DAGs en créant des rôles Airflow personnalisés et en les mappant à vos principaux IAM. Pour plus d'informations et un step-by-step didacticiel, reportez-vous à Tutoriel : Restreindre l'accès d'un utilisateur Amazon MWAA à un sous-ensemble de. DAGs

Vous pouvez également configurer des identités fédérées pour accéder à Amazon MWAA. Pour plus d'informations, reportez-vous à ce qui suit.

- Environnement Amazon MWAA avec accès public — Utilisation d'Okta comme fournisseur d'identité avec Amazon MWAA sur le AWS blog Compute.
- Environnement Amazon MWAA avec accès privé : accès à un environnement Amazon MWAA privé à l'aide d'identités fédérées.

Rôle d'exécution

Amazon MWAA utilise un rôle d'exécution qui accorde des autorisations à votre environnement pour accéder à d'autres AWS services. Vous pouvez donner accès aux AWS services à votre flux de travail en ajoutant les autorisations appropriées au rôle. Si vous choisissez l'option par défaut pour créer un nouveau rôle d'exécution lorsque vous créez l'environnement pour la première fois, Amazon MWAA attache les autorisations minimales nécessaires au rôle, sauf dans le cas des CloudWatch journaux pour lesquels Amazon MWAA ajoute automatiquement tous les groupes de journaux.

Une fois le rôle d'exécution créé, Amazon MWAA ne peut pas gérer ses politiques d'autorisation en votre nom. Pour mettre à jour le rôle d'exécution, vous devez modifier la politique afin d'ajouter et de supprimer des autorisations selon les besoins. Par exemple, vous pouvez intégrer votre environnement Amazon MWAA en AWS Secrets Manager tant que backend pour stocker en toute sécurité les secrets et les chaînes de connexion à utiliser dans vos flux de travail Apache Airflow. Pour ce faire, associez la politique d'autorisation suivante au rôle d'exécution de votre environnement.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:111122223333:secret:*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        }
    ]
}
```

L'intégration à d'autres AWS services suit un schéma similaire : vous ajoutez la politique d'autorisation appropriée à votre rôle d'exécution Amazon MWAA, en autorisant Amazon MWAA à accéder au service. Pour plus d'informations sur la gestion du rôle d'exécution Amazon MWAA et pour voir des exemples supplémentaires, consultez le rôle d'exécution Amazon MWAA dans le guide de l'utilisateur Amazon MWAA.

Migrer vers un nouvel environnement Amazon MWAA

Découvrez les étapes suivantes pour migrer votre charge de travail Apache Airflow existante vers un nouvel environnement Amazon MWAA. Vous pouvez suivre ces étapes pour migrer d'une ancienne version d'Amazon MWAA vers une nouvelle version, ou migrer votre déploiement Apache Airflow autogéré vers Amazon MWAA. Ce didacticiel part du principe que vous migrez d'un Apache Airflow v1.10.12 existant vers un nouvel Amazon MWAA exécutant Apache Airflow v2.5.1, mais vous pouvez utiliser les mêmes procédures pour migrer depuis ou vers différentes versions d'Apache Airflow.

Rubriques

Conditions préalables

Pour effectuer les étapes et migrer votre environnement, vous aurez besoin des éléments suivants :

- Un déploiement d'Apache Airflow. Il peut s'agir d'un environnement Amazon MWAA autogéré ou existant.
- Docker est installé sur votre système d'exploitation local.
- AWS Command Line Interface version 2 installée.

Étape 1 : créer un nouvel environnement Amazon MWAA exécutant la dernière version prise en charge d'Apache Airflow

Vous pouvez créer un environnement en suivant les étapes détaillées décrites dans Getting started with Amazon MWAA dans le guide de l'utilisateur Amazon MWAA, ou en utilisant un CloudFormation modèle. Si vous migrez depuis un environnement Amazon MWAA existant et que vous avez

utilisé un CloudFormation modèle pour créer votre ancien environnement, vous pouvez modifier la `AirflowVersion` propriété pour spécifier la nouvelle version.

```
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
```

Si vous migrez depuis un environnement Amazon MWAA existant, vous pouvez également copier le script Python suivant qui utilise le AWS SDK pour Python (Boto3) pour cloner votre environnement. Vous pouvez également télécharger le script.

Script 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; check the profile name')
    except IndexError as error:
        print("Error:", error)
```

Deuxième étape : migrer les ressources de votre flux de travail

Apache Airflow v2 est une version majeure. Si vous migrez depuis Apache Airflow v1, vous devez préparer les ressources de votre flux de travail et vérifier les modifications que vous DAGs apportez à vos exigences et à vos plugins. Pour ce faire, nous vous recommandons de configurer une version bridge d'Apache Airflow sur votre système d'exploitation local à l'aide de Docker et de l'exécuteur local Amazon MWAA. L'exécuteur local Amazon MWAA fournit un utilitaire d'interface de ligne de commande (CLI) qui réplique un environnement Amazon MWAA localement.

Chaque fois que vous modifiez la version d'Apache Airflow, assurez-vous de référencer l'`--constraint`URL correcte dans votre. `requirements.txt`

Pour migrer les ressources de votre flux de travail

1. Créez un fork du aws-mwaa-local-runnerréférentiel et clonez une copie du runner local Amazon MWAA.
2. Consultez la `v1.10.15` branche du aws-mwaa-local-runner référentiel. Apache Airflow a publié la version v1.10.15 en tant que version intermédiaire pour faciliter la migration vers Apache Airflow v2. Bien qu'Amazon MWAA ne soit pas compatible avec la version v1.10.15, vous pouvez utiliser le moteur d'exécution local Amazon MWAA pour tester vos ressources.

3. Utilisez l'outil CLI Amazon MWAA local Runner pour créer l'image Docker et exécuter Apache Airflow localement. Pour plus d'informations, consultez le fichier README du lanceur local dans le GitHub référentiel.
4. Si Apache Airflow est exécuté localement, suivez les étapes décrites dans la section Mise à niveau de la version 1.10 vers la version 2 sur le site Web de documentation d'Apache Airflow.

    a. Pour mettre à jour votre `requirements.txt` compte, suivez les meilleures pratiques que nous recommandons dans la section Gestion des dépendances Python, dans le guide de l'utilisateur Amazon MWAA.

    b. Si vous avez intégré vos opérateurs et capteurs personnalisés à vos plug-ins pour votre environnement Apache Airflow v1.10.12 existant, déplacez-les dans votre dossier DAG. Pour plus d'informations sur les meilleures pratiques de gestion des modules pour Apache Airflow v2+, reportez-vous à la section Gestion des modules sur le site Web de documentation d'Apache Airflow.
5. Une fois que vous avez apporté les modifications nécessaires à vos ressources de flux de travail, consultez la `v2.5.1` branche du aws-mwaa-local-runner référentiel et testez localement votre flux de travail DAGs mis à jour, vos exigences et vos plugins personnalisés. Si vous migrez vers une autre version d'Apache Airflow, vous pouvez plutôt utiliser la branche d'exécution locale appropriée à votre version.
6. Une fois que vous avez testé avec succès les ressources de votre flux de travail DAGs, `requirements.txt` copiez-les, ainsi que les plugins, dans le compartiment Amazon S3 que vous avez configuré avec votre nouvel environnement Amazon MWAA.

Troisième étape : Exporter les métadonnées de votre environnement existant

Les tables de métadonnées Apache Airflow`dag`, telles que`dag_tag`, et `dag_code` sont automatiquement renseignées lorsque vous copiez les fichiers DAG mis à jour dans le compartiment Amazon S3 de votre environnement et que le planificateur les analyse. Les tables relatives aux autorisations sont également renseignées automatiquement en fonction de votre autorisation de rôle d'exécution IAM. Il n'est pas nécessaire de les faire migrer.

Vous pouvez migrer les données relatives à l'historique du DAG `variable` `slot_pool`s`la_miss`,,, et, si nécessaire `xcom`, `job` et aux `log` tables. Le journal des instances de tâches est stocké dans les CloudWatch journaux du groupe de `airflow-`*`{environment_name}`* journaux. Si vous souhaitez consulter les journaux des instances de tâches pour les anciennes exécutions, ces journaux doivent

être copiés dans le nouveau groupe de journaux d'environnement. Nous vous recommandons de ne déplacer que l'équivalent de quelques jours de journaux afin de réduire les coûts associés.

Si vous migrez depuis un environnement Amazon MWAA existant, il n'y a pas d'accès direct à la base de données de métadonnées. Vous devez exécuter un DAG pour exporter les métadonnées de votre environnement Amazon MWAA existant vers un compartiment Amazon S3 de votre choix. Les étapes suivantes peuvent également être utilisées pour exporter les métadonnées Apache Airflow si vous migrez depuis un environnement autogéré.

Une fois les données exportées, vous pouvez exécuter un DAG dans votre nouvel environnement pour les importer. Pendant le processus d'exportation et d'importation, tous les autres processus DAGs sont interrompus.

Pour exporter les métadonnées depuis votre environnement existant

1. Créez un compartiment Amazon S3 à l'aide du AWS CLI pour stocker les données exportées. Remplacez le `UUID` et `region` par vos informations.

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

   > **Note**
   >
   > Si vous migrez des données sensibles, telles que des connexions que vous stockez dans des variables, nous vous recommandons d'activer le chiffrement par défaut pour le compartiment Amazon S3.

2. > **Note**
   >
   > Ne s'applique pas à la migration depuis un environnement autogéré.

   Modifiez le rôle d'exécution de l'environnement existant et ajoutez la politique suivante pour accorder un accès en écriture au bucket que vous avez créé à la première étape.

JSON

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

3. Clonez le amazon-mwaa-examplesréférentiel et accédez au `metadata-migration` sous-répertoire correspondant à votre scénario de migration.

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

4. Dans`export_data.py`, remplacez la valeur de chaîne pour S3_BUCKET par le compartiment Amazon S3 que vous avez créé pour stocker les métadonnées exportées.

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

5. Localisez le `requirements.txt` fichier dans le `metadata-migration` répertoire. Si vous disposez déjà d'un fichier d'exigences pour votre environnement existant, ajoutez les exigences supplémentaires spécifiées dans `requirements.txt` votre fichier. Si vous n'avez pas de fichier d'exigences existant, vous pouvez simplement utiliser celui fourni dans le `metadata-migration` répertoire.

6. `export_data.py`Copiez dans le répertoire DAG du compartiment Amazon S3 associé à votre environnement existant. Si vous migrez depuis un environnement autogéré, copiez-le dans votre `export_data.py` `/dags` dossier.

7. Copiez votre mise `requirements.txt` à jour dans le compartiment Amazon S3 associé à votre environnement existant, puis modifiez l'environnement pour spécifier la nouvelle `requirements.txt` version.
8. Une fois l'environnement mis à jour, accédez à l'interface utilisateur d'Apache Airflow, annulez le `db_export` DAG et déclenchez l'exécution du flux de travail.
9. Vérifiez que les métadonnées sont `data/migration/`*`existing-version`*`_to_`*`new-version`*`/export/` exportées vers le compartiment `mwaa-migration-`*`{UUID}`* Amazon S3, chaque table figurant dans son propre fichier dédié.

Étape 4 : Importation des métadonnées dans votre nouvel environnement

Pour importer les métadonnées dans votre nouvel environnement

1. Dans`import_data.py`, remplacez les valeurs de chaîne suivantes par vos informations.

   - Pour la migration depuis un environnement Amazon MWAA existant :

     ```
     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`contrôle le nombre de jours de fichiers journaux que le flux de travail copie dans le nouvel environnement.

   - Pour la migration depuis un environnement autogéré :

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

2. (Facultatif) `import_data.py` copie uniquement les journaux des tâches ayant échoué. Si vous souhaitez copier tous les journaux des tâches, modifiez la `getDagTasks` fonction et supprimez-la `ti.state = 'failed'` comme indiqué dans l'extrait de code suivant.

   ```
   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. Modifiez le rôle d'exécution de votre nouvel environnement et ajoutez la politique suivante. La politique d'autorisation permet à Amazon MWAA de lire le contenu du compartiment Amazon S3 dans lequel vous avez exporté les métadonnées d'Apache Airflow, et de copier les journaux des instances de tâches à partir de groupes de journaux existants. Remplacez tous les espaces réservés par vos informations.

> **Note**
>
> Si vous migrez depuis un environnement autogéré, vous devez supprimer les autorisations liées aux CloudWatch journaux de la politique.

JSON

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

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

4. Copiez `import_data.py` dans le répertoire DAG du compartiment Amazon S3 associé à votre nouvel environnement, puis accédez à l'interface utilisateur d'Apache Airflow pour annuler le `db_import` DAG et déclencher le flux de travail. Le nouveau DAG apparaîtra dans l'interface utilisateur d'Apache Airflow dans quelques minutes.
5. Une fois l'exécution du DAG terminée, vérifiez que l'historique d'exécution de votre DAG est copié en accédant à chaque DAG individuel.

Étapes suivantes

- Pour plus d'informations sur les classes et fonctionnalités d'environnement Amazon MWAA disponibles, reportez-vous à la classe d'environnement Amazon MWAA dans le guide de l'utilisateur Amazon MWAA.
- Pour plus d'informations sur la façon dont Amazon MWAA gère le dimensionnement automatique des travailleurs, reportez-vous au dimensionnement automatique Amazon MWAA dans le guide de l'utilisateur Amazon MWAA.
- Pour plus d'informations sur l'API REST Amazon MWAA, consultez l'API REST Amazon MWAA.

Migrer les charges de travail depuis Amazon AWS Data Pipeline MWAA

AWS a lancé le AWS Data Pipeline service en 2012. À l'époque, les clients recherchaient un service leur permettant d'utiliser diverses options de calcul pour déplacer des données entre différentes sources de données. À mesure que les besoins en matière de transfert de données ont évolué au fil du temps, les solutions à ces besoins ont également évolué. Vous avez désormais la possibilité de choisir la solution qui répond le mieux aux besoins de votre entreprise. Vous pouvez migrer vos charges de travail vers l'un des AWS services suivants :

- Utilisez Amazon Managed Workflows for Apache Airflow (Amazon MWAA) pour gérer l'orchestration des flux de travail pour Apache Airflow.
- Utilisez Step Functions pour orchestrer des flux de travail entre plusieurs Services AWS.
- AWS Glue À utiliser pour exécuter et orchestrer les applications Apache Spark.

L'option que vous choisissez dépend de votre charge de travail actuelle AWS Data Pipeline. Cette rubrique explique comment effectuer une migration depuis AWS Data Pipeline Amazon MWAA.

Rubriques

Choisir Amazon MWAA

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) est un service d'orchestration géré pour Apache Airflow qui vous permet de configurer et d'exploiter des pipelines de end-to-end données dans le cloud à grande échelle. Apache Airflow est un outil open source utilisé pour créer, planifier et surveiller par programmation des séquences de processus et de tâches appelées flux de travail. Avec Amazon MWAA, vous pouvez utiliser Apache Airflow et le langage de programmation Python pour créer des flux de travail sans avoir à gérer l'infrastructure sous-jacente en termes d'évolutivité, de disponibilité et de sécurité. Amazon MWAA adapte automatiquement sa capacité

de flux de travail en fonction de vos besoins et est intégré aux services de AWS sécurité pour vous permettre d'accéder rapidement et en toute sécurité à vos données.

Ce qui suit met en évidence certains des avantages de la migration depuis AWS Data Pipeline Amazon MWAA :

- Évolutivité et performances améliorées : Amazon MWAA fournit un cadre flexible et évolutif pour définir et exécuter des flux de travail. Cela permet aux utilisateurs de gérer facilement des flux de travail volumineux et complexes et de tirer parti de fonctionnalités telles que la planification dynamique des tâches, les flux de travail basés sur les données et le parallélisme.
- Surveillance et journalisation améliorées : Amazon MWAA s'intègre CloudWatch à Amazon pour améliorer la surveillance et la journalisation de vos flux de travail. Amazon MWAA envoie automatiquement les statistiques et les journaux du système à CloudWatch. Cela signifie que vous pouvez suivre la progression et les performances de vos flux de travail en temps réel et identifier les problèmes éventuels.
- Meilleures intégrations avec les AWS services et les logiciels tiers : Amazon MWAA s'intègre à de nombreux autres AWS services, tels qu'Amazon S3 et Amazon Redshift AWS Glue, ainsi qu'à des logiciels tiers tels que DBT, Snowflake et Databricks. Cela vous permet de traiter et de transférer des données entre différents environnements et services.
- Outil de pipeline de données open source : Amazon MWAA utilise le même produit open source Apache Airflow que vous connaissez. Apache Airflow est un outil spécialement conçu pour gérer tous les aspects de la gestion du pipeline de données, notamment l'ingestion, le traitement, le transfert, les tests d'intégrité, les contrôles de qualité et la garantie du lignage des données.
- Architecture moderne et flexible : Amazon MWAA tire parti de la conteneurisation et des technologies sans serveur natives dans le cloud. Cela signifie une flexibilité et une portabilité accrues, ainsi qu'un déploiement et une gestion simplifiés de vos environnements de flux de travail.

Cartographie de l'architecture et des concepts

AWS Data Pipeline et Amazon MWAA ont des architectures et des composants différents, ce qui peut affecter le processus de migration et la façon dont les flux de travail sont définis et exécutés. Cette section présente l'architecture et les composants des deux services et met en évidence certaines des principales différences.

Amazon MWAA AWS Data Pipeline et Amazon sont tous deux des services entièrement gérés. Lorsque vous migrez vos charges de travail vers Amazon MWAA, vous devrez peut-être apprendre

de nouveaux concepts pour modéliser vos flux de travail existants à l'aide d'Apache Airflow. Toutefois, vous n'aurez pas à gérer l'infrastructure, les correctifs ni les mises à jour du système d'exploitation.

Le tableau suivant associe les concepts clés à ceux AWS Data Pipeline d'Amazon MWAA. Utilisez ces informations comme point de départ pour concevoir un plan de migration.

Concept	AWS Data Pipeline	Amazon MWAA
Définition du pipeline	AWS Data Pipeline utilise un fichier de configuration basé sur JSON qui définit le flux de travail.	Amazon MWAA utilise des graphes acycliques dirigés basés sur Python (DAGs) qui définissent le flux de travail.
Environnement d'exécution du pipeline	Les flux de travail s'exécute nt sur des instances Amazon EC2. AWS Data Pipeline approvisionne et gère ces instances en votre nom.	Amazon MWAA utilise les environnements conteneurisés Amazon ECS pour exécuter des tâches.
Composants du pipeline	Les activités sont des tâches de traitement exécutées dans le cadre du flux de travail.	Les opérateurs (tâches) sont les unités de traitemen t fondamentales d'un flux de travail.
	Les conditions préalables contiennent des instructions conditionnelles qui doivent être vraies pour qu'une activité puisse être exécutée.	Les capteurs (tâches) représentent des instructions conditionnelles qui peuvent attendre la fin d'une ressource ou d'une tâche avant de l'exécuter.
	Une ressource in AWS Data Pipeline fait référence à la ressource de AWS calcul qui exécute le travail spécifié par une activité de pipeline. Amazon EC2 et Amazon	À l'aide des tâches d'un DAG, vous pouvez définir diverses ressources de calcul, notamment Amazon ECS, Amazon EMR et Amazon EKS. Amazon MWAA exécute

Concept	AWS Data Pipeline	Amazon MWAA
	EMR sont deux ressources disponibles.	des opérations Python sur des travailleurs exécutés sur Amazon ECS.
Exécution du pipeline	AWS Data Pipeline prend en charge les cycles de planification avec des modèles réguliers basés sur les taux et basés sur des crons.	Amazon MWAA prend en charge la planification avec des expressions cron et des préréglages, ainsi que des horaires personnalisés.
	Une instance fait référence à chaque exécution du pipeline.	Une exécution DAG fait référence à chaque exécution d'un flux de travail Apache Airflow.
	Une tentative fait référence à une nouvelle tentative d'une opération qui a échoué.	Amazon MWAA prend en charge les nouvelles tentative s que vous définissez soit au niveau du DAG, soit au niveau de la tâche.

Exemples d'implémentations

Dans de nombreux cas, vous pourrez réutiliser les ressources avec lesquelles vous êtes en train d'orchestrer AWS Data Pipeline après avoir migré vers Amazon MWAA. La liste suivante contient des exemples d'implémentations utilisant Amazon MWAA pour les cas d' AWS Data Pipeline utilisation les plus courants.

- Exécution d'une tâche Amazon EMR (atelier)AWS
- Création d'un plugin personnalisé pour Apache Hive et Hadoop (Guide de l'utilisateur Amazon MWAA)
- Copier des données de S3 vers Redshift (atelier)AWS
- Exécution d'un script shell sur une instance Amazon ECS distante (Amazon MWAA User Guide)
- Orchestration de flux de travail hybrides (sur site) (article de blog)

Pour obtenir des didacticiels et des exemples supplémentaires, consultez les documents suivants :

- Tutoriels Amazon MWAA
- Exemples de code Amazon MWAA

Comparaison des prix

AWS Data Pipeline La tarification est basée sur le nombre de pipelines, ainsi que sur la quantité utilisée par chaque pipeline. Les activités que vous organisez plus d'une fois par jour (fréquence élevée) coûtent 1$ par mois et par activité. Les activités que vous exécutez une fois par jour ou moins (basse fréquence) coûtent 0,60$ par mois et par activité. Le prix des pipelines inactifs est de 1$ par pipeline. Pour plus d'informations, consultez la page de AWS Data Pipeline tarification.

La tarification d'Amazon MWAA est basée sur la durée pendant laquelle votre environnement Apache Airflow géré existe et sur toute mise à l'échelle automatique supplémentaire requise pour fournir davantage de personnel ou de capacité de planification. Vous payez l'utilisation de votre environnement Amazon MWAA sur une base horaire (facturée à une seconde de résolution), avec des frais variables en fonction de la taille de l'environnement. Amazon MWAA adapte automatiquement le nombre de travailleurs en fonction de la configuration de votre environnement. AWS calcule le coût des travailleurs supplémentaires séparément. Pour plus d'informations sur le coût horaire lié à l'utilisation de différentes tailles d'environnement Amazon MWAA, consultez la page de tarification Amazon MWAA.

Ressources connexes

Pour plus d'informations et pour connaître les meilleures pratiques relatives à l'utilisation d'Amazon MWAA, consultez les ressources suivantes :

- La référence de l'API Amazon MWAA
- Tableaux de bord de surveillance et alarmes sur Amazon MWAA
- Optimisation des performances pour Apache Airflow sur Amazon MWAA

Historique du document Amazon MWAA

Le tableau suivant décrit les ajouts importants au guide de migration Amazon MWAA, à compter de mars 2022.

Modification	Description	Date
Nouveau sujet sur la migration des charges de travail depuis AWS Data Pipeline Amazon MWAA	Ajout de nouvelles informations et directives sur la migration des charges de travail existantes depuis AWS Data Pipeline Amazon MWAA. Utilisez ces informations pour vous aider à concevoir un plan de migration. • Migrer les charges de travail depuis Amazon AWS Data Pipeline MWAA	14 avril 2023
Lancement du guide de migration Amazon MWAA	Amazon MWAA propose désormais des conseils détaillés sur la migration vers un nouvel environnement Amazon MWAA. Les étapes décrites dans le guide de migration Amazon MWAA s'appliquent à la migration depuis un environnement Amazon MWAA existant ou depuis un déploiement Apache Airflow autogéré. • À propos du guide de migration Amazon MWAA	7 mars 2022

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.