Migrationshandbuch

Amazon Managed Workflows für Apache Airflow

Amazon Managed Workflows für Apache Airflow: Migrationshandbuch

Table of Contents

Was ist der Amazon MWAA-Migrationsleitfaden?

Amazon Managed Workflows for Apache Airflow ist ein verwalteter Orchestrierungsservice für Apache Airflow, mit dem Sie Daten-Pipelines in der Cloud in großem Umfang betreiben können. Amazon MWAA verwaltet die Bereitstellung und laufende Wartung von Apache Airflow, sodass Sie sich keine Gedanken mehr über das Patchen, Skalieren oder Sichern von Instances machen müssen.

Amazon MWAA skaliert automatisch die Rechenressourcen, die Aufgaben ausführen, um bei Bedarf eine konsistente Leistung zu gewährleisten. Amazon MWAA schützt Ihre Daten standardmäßig. Ihre Workloads werden mithilfe von Amazon Virtual Private Cloud in Ihrer eigenen isolierten und sicheren Cloud-Umgebung ausgeführt. Dadurch wird sichergestellt, dass Daten automatisch verschlüsselt werden mit AWS Key Management Service.

Verwenden Sie dieses Handbuch, um Ihre selbstverwalteten Apache Airflow-Workflows auf Amazon MWAA zu migrieren oder eine bestehende Amazon MWAA-Umgebung auf eine neue Apache Airflow-Version zu aktualisieren. Das Migrations-Tutorial beschreibt, wie Sie eine neue Amazon MWAA-Umgebung erstellen oder klonen, Ihre Workflow-Ressourcen migrieren und Ihre Workflow-Metadaten und -Protokolle in Ihre neue Umgebung übertragen können.

Bevor Sie das Migrations-Tutorial ausprobieren, empfehlen wir Ihnen, sich mit den folgenden Themen vertraut zu machen.

- Netzwerkarchitektur
- Die wichtigsten Überlegungen

Erkunden Sie die Amazon MWAA-Netzwerkarchitektur

Im folgenden Abschnitt werden die Hauptkomponenten einer Amazon MWAA-Umgebung sowie die AWS Services beschrieben, in die jede Umgebung integriert ist, um ihre Ressourcen zu verwalten, Ihre Daten zu schützen und Ihre Workflows zu überwachen und sichtbar zu machen.

Themen

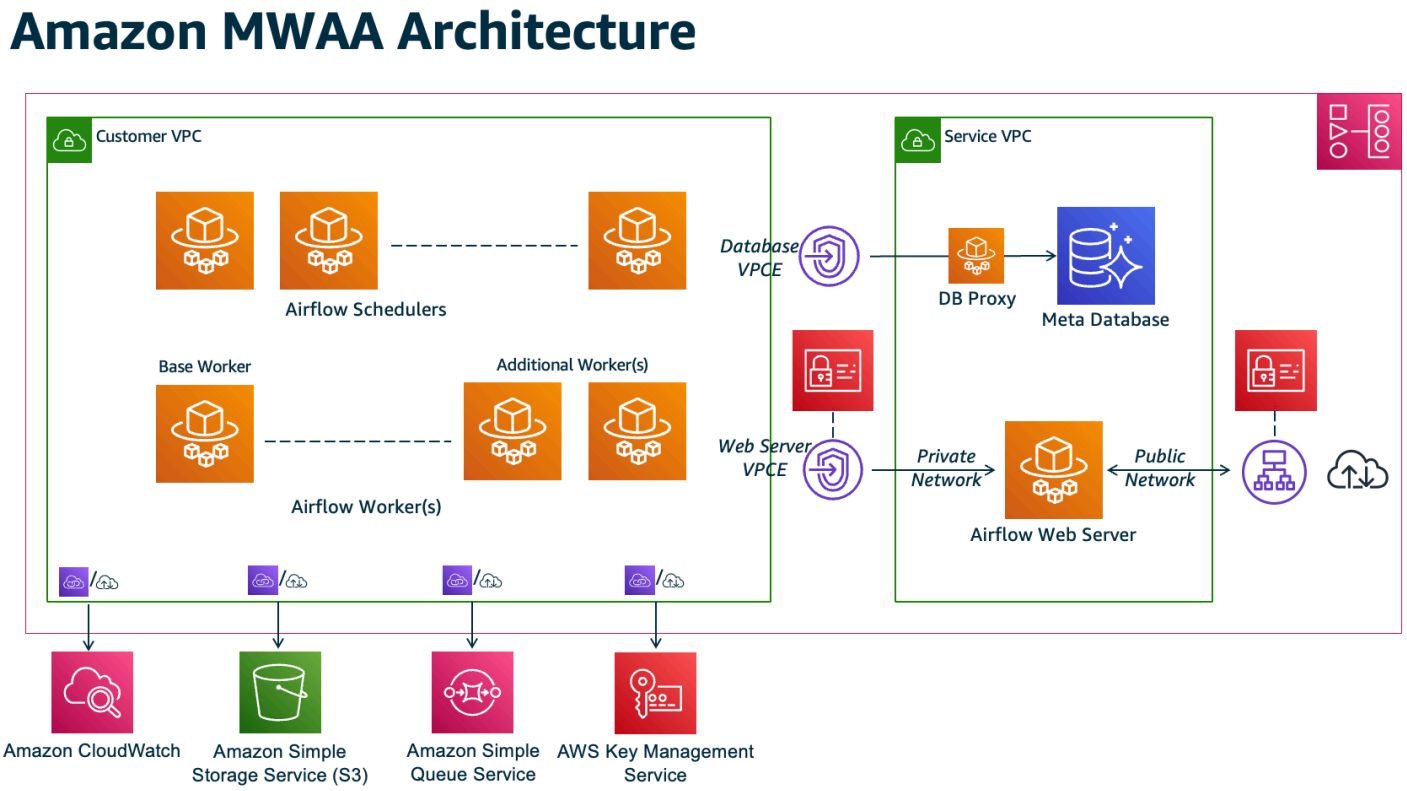
Amazon MWAA-Komponenten

Amazon MWAA-Umgebungen bestehen aus den folgenden vier Hauptkomponenten:

1. Scheduler — Analysiert und überwacht all Ihre Aufgaben und stellt Aufgaben zur Ausführung in eine Warteschlange DAGs, wenn die Abhängigkeiten einer DAG erfüllt sind. Amazon MWAA stellt den Scheduler als AWS Fargate Cluster mit mindestens 2 Schedulern bereit. Sie können die Anzahl der Scheduler je nach Arbeitslast auf bis zu fünf erhöhen. Weitere Informationen zu Amazon MWAA-Umgebungsklassen finden Sie unter Amazon MWAA-Umgebungsklasse.
2. Mitarbeiter — Eine oder mehrere Fargate-Aufgaben, die Ihre geplanten Aufgaben ausführen. Die Anzahl der Mitarbeiter für Ihre Umgebung wird durch einen Bereich zwischen einer von Ihnen angegebenen Mindest - und Höchstzahl bestimmt. Amazon MWAA beginnt mit der auto-scaling von Workern, wenn die Anzahl der in der Warteschlange stehenden und laufenden Aufgaben höher ist, als Ihre vorhandenen Worker bewältigen können. Wenn die Summe laufender Aufgaben und Aufgaben in der Warteschlange länger als zwei Minuten Null ergibt, reduziert Amazon MWAA die Anzahl der Mitarbeiter auf ein Minimum. Weitere Informationen darüber, wie Amazon MWAA mit automatischer Skalierung von Workern umgeht, finden Sie unter auto-scaling von Amazon MWAA.
3. Webserver — Führt die Apache Airflow-Weboberfläche aus. Sie können den Webserver mit privatem oder öffentlichem Netzwerkzugriff konfigurieren. In beiden Fällen wird der Zugriff auf Ihre Apache Airflow-Benutzer durch die Zugriffskontrollrichtlinie gesteuert, die Sie in AWS Identity and Access Management (IAM) definieren. Weitere Informationen zur Konfiguration von IAM-Zugriffsrichtlinien für Ihre Umgebung finden Sie unter Zugreifen auf eine Amazon MWAA-Umgebung.

4. Datenbank — Speichert Metadaten über die Apache Airflow-Umgebung und Ihre Workflows, einschließlich des DAG-Ausführungsverlaufs. Bei der Datenbank handelt es sich um eine Aurora PostgreSQL Single-Tenant-Datenbank AWS, die von den Containern Scheduler und Workers 'Fargate verwaltet wird und auf die über einen privat gesicherten Amazon VPC-Endpunkt zugegriffen werden kann.

Jede Amazon MWAA-Umgebung interagiert auch mit einer Reihe von AWS Services, um eine Vielzahl von Aufgaben zu bewältigen, darunter Speichern DAGs und Zugreifen sowie Aufgabenabhängigkeiten, Sicherung Ihrer Daten im Ruhezustand sowie Protokollierung und Überwachung Ihrer Umgebung. Das folgende Diagramm zeigt die verschiedenen Komponenten einer Amazon MWAA-Umgebung.



> **Note**
>
> Der Service Amazon VPC ist keine gemeinsam genutzte VPC. Amazon MWAA erstellt für jede Umgebung, die Sie erstellen, eine AWS eigene VPC.

- Amazon S3 — Amazon MWAA speichert all Ihre Workflow-Ressourcen wie DAGs Anforderungen und Plugin-Dateien in einem Amazon S3 S3-Bucket. Weitere Informationen zum Erstellen des Buckets als Teil der Umgebungserstellung und zum Hochladen Ihrer Amazon MWAA-Ressourcen finden Sie unter Erstellen eines Amazon S3 S3-Buckets für Amazon MWAA im Amazon MWAA-Benutzerhandbuch.
- Amazon SQS — Amazon MWAA verwendet Amazon SQS, um Ihre Workflow-Aufgaben mit einem Celery Executor in die Warteschlange zu stellen.
- Amazon ECR — Amazon ECR hostet alle Apache Airflow-Images. Amazon MWAA unterstützt nur AWS verwaltete Apache Airflow-Images.
- AWS KMS— Amazon MWAA verwendet AWS KMS , um sicherzustellen, dass Ihre Daten im Ruhezustand sicher sind. Standardmäßig verwendet Amazon MWAA AWS verwaltete AWS KMS Schlüssel, aber Sie können Ihre Umgebung so konfigurieren, dass Ihr eigener, vom AWS KMS Kunden verwalteter Schlüssel verwendet wird. Weitere Informationen zur Verwendung Ihres eigenen kundenverwalteten AWS KMS Schlüssels finden Sie unter Vom Kunden verwaltete Schlüssel für Datenverschlüsselung im Amazon MWAA-Benutzerhandbuch.
- CloudWatch— Amazon MWAA ist in Apache Airflow-Protokolle CloudWatch und Umgebungsmetriken integriert und stellt diese bereit CloudWatch, sodass Sie Ihre Amazon MWAA-Ressourcen überwachen und Probleme beheben können.

Konnektivität

Ihre Amazon MWAA-Umgebung benötigt Zugriff auf alle AWS Dienste, in die sie integriert ist. Die Amazon MWAA-Ausführungsrolle steuert, wie Amazon MWAA Zugriff gewährt wird, um in Ihrem Namen eine Verbindung zu anderen AWS Diensten herzustellen. Für die Netzwerkkonnektivität können Sie entweder öffentlichen Internetzugang für Ihre Amazon VPC bereitstellen oder Amazon VPC-Endpunkte erstellen. Weitere Informationen zur Konfiguration von Amazon VPC-Endpunkten (AWS PrivateLink) für Ihre Umgebung finden Sie unter Verwaltung des Zugriffs auf VPC-Endpunkte auf Amazon MWAA im Amazon MWAA-Benutzerhandbuch.

Amazon MWAA installiert Anforderungen für den Scheduler und den Worker. Wenn Ihre Anforderungen aus einem öffentlichen PyPiRepository stammen, benötigt Ihre Umgebung eine Verbindung zum Internet, um die erforderlichen Bibliotheken herunterladen zu können. Für private Umgebungen können Sie entweder ein privates PyPi Repository verwenden oder die Bibliotheken in `.whl`Dateien als benutzerdefinierte Plugins für Ihre Umgebung bündeln.

Wenn Sie Apache Airflow im privaten Modus konfigurieren, kann Ihre Amazon VPC nur über Amazon VPC-Endpunkte auf die Apache Airflow-Benutzeroberfläche zugreifen.

Weitere Informationen zu Netzwerken finden Sie unter Networking im Amazon MWAA-Benutzerhandbuch.

Wichtige Überlegungen zur Migration zu einer neuen MWAA-Umgebung

Erfahren Sie mehr über wichtige Überlegungen, wie Authentifizierung und die Amazon MWAA-Ausführungsrolle, wenn Sie planen, Ihre Apache Airflow-Workloads zu Amazon MWAA zu migrieren.

Themen

Authentifizierung

Amazon MWAA verwendet AWS Identity and Access Management (IAM), um den Zugriff auf die Apache Airflow-Benutzeroberfläche zu steuern. Sie müssen IAM-Richtlinien erstellen und verwalten, die Ihren Apache Airflow-Benutzern Zugriff auf den Webserver und dessen Verwaltung gewähren. DAGs Sie können sowohl die Authentifizierung als auch die Autorisierung für die Standardrollen von Apache Airflow mithilfe von IAM für verschiedene Konten verwalten.

Sie können Apache Airflow-Benutzer weiter verwalten und einschränken, sodass sie nur auf einen Teil Ihres Workflows zugreifen können, DAGs indem Sie benutzerdefinierte Airflow-Rollen erstellen und sie Ihren IAM-Prinzipalen zuordnen. Weitere Informationen und ein step-by-step Tutorial finden Sie unter Tutorial: Beschränken des Zugriffs eines Amazon MWAA-Benutzers auf eine Teilmenge von. DAGs

Sie können auch föderierte Identitäten für den Zugriff auf Amazon MWAA konfigurieren. Weitere Informationen finden Sie im Folgenden.

- Amazon MWAA-Umgebung mit öffentlichem Zugriff — Verwendung von Okta als Identitätsanbieter mit Amazon MWAA im Compute-Blog.AWS
- Amazon MWAA-Umgebung mit privatem Zugriff — Zugriff auf eine private Amazon MWAA-Umgebung mit föderierten Identitäten.

Ausführungsrolle

Amazon MWAA verwendet eine Ausführungsrolle, die Ihrer Umgebung Berechtigungen für den Zugriff auf andere AWS Services gewährt. Sie können Ihrem Workflow Zugriff auf AWS Dienste gewähren, indem Sie der Rolle die entsprechenden Berechtigungen hinzufügen. Wenn Sie bei der ersten Erstellung der Umgebung die Standardoption zum Erstellen einer neuen Ausführungsrolle wählen, fügt Amazon MWAA der Rolle die erforderlichen Mindestberechtigungen zu, außer im Fall von CloudWatch Logs, für die Amazon MWAA alle Protokollgruppen automatisch hinzufügt.

Sobald die Ausführungsrolle erstellt wurde, kann Amazon MWAA seine Berechtigungsrichtlinien nicht mehr in Ihrem Namen verwalten. Um die Ausführungsrolle zu aktualisieren, müssen Sie die Richtlinie bearbeiten, um nach Bedarf Berechtigungen hinzuzufügen oder zu entfernen. Sie können beispielsweise Ihre Amazon MWAA-Umgebung AWS Secrets Manager als Backend integrieren, um Geheimnisse und Verbindungszeichenfolgen sicher zu speichern, die Sie in Ihren Apache Airflow-Workflows verwenden können. Fügen Sie dazu der Ausführungsrolle Ihrer Umgebung die folgende Berechtigungsrichtlinie hinzu.

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

Die Integration mit anderen AWS Services folgt einem ähnlichen Muster: Sie fügen die entsprechende Berechtigungsrichtlinie zu Ihrer Amazon MWAA-Ausführungsrolle hinzu und

gewähren Amazon MWAA die Erlaubnis, auf den Service zuzugreifen. Weitere Informationen zur Verwaltung der Amazon MWAA-Ausführungsrolle und weitere Beispiele finden Sie unter Amazon MWAA-Ausführungsrolle im Amazon MWAA-Benutzerhandbuch.

Migrieren Sie zu einer neuen Amazon MWAA-Umgebung

Erkunden Sie die folgenden Schritte, um Ihren vorhandenen Apache Airflow-Workload auf eine neue Amazon MWAA-Umgebung zu migrieren. Sie können diese Schritte verwenden, um von einer älteren Version von Amazon MWAA zu einer neuen Version zu migrieren oder Ihre selbstverwaltete Apache Airflow-Bereitstellung zu Amazon MWAA zu migrieren. In diesem Tutorial wird davon ausgegangen, dass Sie von einem vorhandenen Apache Airflow v1.10.12 zu einem neuen Amazon MWAA migrieren, auf dem Apache Airflow v2.5.1 ausgeführt wird. Sie können jedoch dieselben Verfahren verwenden, um von oder zu verschiedenen Apache Airflow-Versionen zu migrieren.

Themen

Voraussetzungen

Um die Schritte abschließen und Ihre Umgebung migrieren zu können, benötigen Sie Folgendes:

- Eine Apache Airflow-Bereitstellung. Dabei kann es sich um eine selbstverwaltete oder bestehende Amazon MWAA-Umgebung handeln.
- Docker ist für Ihr lokales Betriebssystem installiert.
- AWS Command Line Interface Version 2 installiert.

Schritt eins: Erstellen Sie eine neue Amazon MWAA-Umgebung, in der die neueste unterstützte Apache Airflow-Version ausgeführt wird

Sie können eine Umgebung mithilfe der detaillierten Schritte unter Erste Schritte mit Amazon MWAA im Amazon MWAA-Benutzerhandbuch oder mithilfe einer Vorlage erstellen. AWS CloudFormation Wenn Sie aus einer bestehenden Amazon MWAA-Umgebung migrieren und eine AWS CloudFormation Vorlage verwendet haben, um Ihre alte Umgebung zu erstellen, können Sie die `AirflowVersion` Eigenschaft ändern, um die neue Version anzugeben.

```
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
```

Wenn Sie aus einer vorhandenen Amazon MWAA-Umgebung migrieren, können Sie alternativ das folgende Python-Skript kopieren, das das AWS SDK für Python (Boto3) verwendet, um Ihre Umgebung zu klonen. Sie können das Skript auch herunterladen.

Python-Skript

```
# 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)
```

Schritt zwei: Migrieren Sie Ihre Workflow-Ressourcen

Apache Airflow v2 ist eine Hauptversion. Wenn Sie von Apache Airflow v1 migrieren, müssen Sie Ihre Workflow-Ressourcen vorbereiten und die Änderungen überprüfen DAGs, die Sie an Ihren Anforderungen und Plugins vornehmen. Zu diesem Zweck empfehlen wir, eine Bridge-Version von Apache Airflow auf Ihrem lokalen Betriebssystem mithilfe von Docker und dem Amazon MWAA Local Runner zu konfigurieren. Der Amazon MWAA Local Runner bietet ein Befehlszeilenschnittstellen-Hilfsprogramm (CLI), das eine Amazon MWAA-Umgebung lokal repliziert.

Wenn Sie Apache Airflow-Versionen ändern, stellen Sie sicher, dass Sie in Ihrer Version auf die richtige URL verweisen. `--constraint` `requirements.txt`

Um Ihre Workflow-Ressourcen zu migrieren

1. Erstellen Sie einen Fork des aws-mwaa-local-runnerRepositorys und klonen Sie eine Kopie des lokalen Amazon MWAA-Runners.
2. Checken Sie den `v1.10.15` Zweig des Repositorys aus. aws-mwaa-local-runner Apache Airflow hat v1.10.15 als Bridge-Version veröffentlicht, um die Migration zu Apache Airflow v2 zu unterstützen. Obwohl Amazon MWAA v1.10.15 nicht unterstützt, können Sie den lokalen Amazon MWAA-Runner verwenden, um Ihre Ressourcen zu testen.
3. Verwenden Sie das Amazon MWAA Local Runner CLI-Tool, um das Docker-Image zu erstellen und Apache Airflow lokal auszuführen. Weitere Informationen finden Sie in der README-Datei für den lokalen Runner im Repository. GitHub
4. Wenn Apache Airflow lokal ausgeführt wird, folgen Sie den Schritten, die unter Upgrade von 1.10 auf 2 auf der Apache Airflow-Dokumentationswebsite beschrieben sind.
    a. Um Ihre zu aktualisieren`requirements.txt`, folgen Sie den bewährten Methoden, die wir unter Verwaltung von Python-Abhängigkeiten im Amazon MWAA-Benutzerhandbuch empfehlen.
    b. Wenn Sie Ihre benutzerdefinierten Operatoren und Sensoren mit Ihren Plugins für Ihre bestehende Apache Airflow v1.10.12-Umgebung gebündelt haben, verschieben Sie sie in Ihren DAG-Ordner. Weitere Informationen zu bewährten Methoden zur Modulverwaltung für Apache Airflow v2+ finden Sie unter Modulverwaltung auf der Apache Airflow-Dokumentationswebsite.
5. Nachdem Sie die erforderlichen Änderungen an Ihren Workflow-Ressourcen vorgenommen haben, checken Sie den `v2.5.1` Zweig des aws-mwaa-local-runner Repositorys aus und testen Sie Ihren aktualisierten Workflow DAGs, Ihre Anforderungen und Ihre benutzerdefinierten Plugins lokal. Wenn Sie zu einer anderen Apache Airflow-Version migrieren, können Sie stattdessen den entsprechenden lokalen Runner-Branch für Ihre Version verwenden.
6. Nachdem Sie Ihre Workflow-Ressourcen erfolgreich getestet haben, kopieren Sie Ihre DAGs`requirements.txt`, und Plugins in den Amazon S3 S3-Bucket, den Sie mit Ihrer neuen Amazon MWAA-Umgebung konfiguriert haben.

Schritt drei: Exportieren der Metadaten aus Ihrer bestehenden Umgebung

Apache Airflow-Metadatentabellen wie`dag`,`dag_tag`, und `dag_code` werden automatisch aufgefüllt, wenn Sie die aktualisierten DAG-Dateien in den Amazon S3 S3-Bucket Ihrer Umgebung kopieren und der Scheduler sie analysiert. Auch berechtigungsbezogene Tabellen werden basierend auf Ihrer IAM-Ausführungsrolle automatisch aufgefüllt. Sie müssen sie nicht migrieren.

Sie können Daten migrieren, die sich auf den DAG-Verlauf `variable`,`slot_pool`,`sla_miss`,, und, falls erforderlich`xcom`,`job`, und `log` Tabellen beziehen. Das Protokoll der Taskinstanz wird in den CloudWatch Protokollen unter der `airflow-`*`{environment_name}`* Protokollgruppe gespeichert. Wenn Sie die Protokolle der Task-Instanz für ältere Läufe sehen möchten, müssen diese Protokolle in die neue Umgebungsprotokollgruppe kopiert werden. Wir empfehlen, dass Sie nur Protokolle für einige Tage verschieben, um die damit verbundenen Kosten zu senken.

Wenn Sie aus einer bestehenden Amazon MWAA-Umgebung migrieren, gibt es keinen direkten Zugriff auf die Metadaten-Datenbank. Sie müssen eine DAG ausführen, um die Metadaten aus Ihrer bestehenden Amazon MWAA-Umgebung in einen Amazon S3 S3-Bucket Ihrer Wahl zu exportieren. Die folgenden Schritte können auch verwendet werden, um Apache Airflow-Metadaten zu exportieren, wenn Sie aus einer selbstverwalteten Umgebung migrieren.

Nachdem die Daten exportiert wurden, können Sie in Ihrer neuen Umgebung eine DAG ausführen, um die Daten zu importieren. Während des Export- und Importvorgangs DAGs werden alle anderen angehalten.

Um die Metadaten aus Ihrer vorhandenen Umgebung zu exportieren

1. Erstellen Sie einen Amazon S3 S3-Bucket mit dem AWS CLI , um die exportierten Daten zu speichern. Ersetzen Sie das `UUID` und `region` durch Ihre Informationen.

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

> **Note**
>
> Wenn Sie sensible Daten migrieren, z. B. Verbindungen, die Sie in Variablen speichern, empfehlen wir Ihnen, die Standardverschlüsselung für den Amazon S3 S3-Bucket zu aktivieren.

2.
> **Note**
>
> Gilt nicht für die Migration aus einer selbstverwalteten Umgebung.

Ändern Sie die Ausführungsrolle der vorhandenen Umgebung und fügen Sie die folgende Richtlinie hinzu, um Schreibzugriff auf den Bucket zu gewähren, den Sie in Schritt 1 erstellt haben.

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

3. Klonen Sie das amazon-mwaa-examplesRepository und navigieren Sie zum `metadata-migration` Unterverzeichnis für Ihr Migrationsszenario.

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

4. Ersetzen Sie in `export_data.py` den Zeichenkettenwert für `S3_BUCKET` durch den Amazon S3 S3-Bucket, den Sie zum Speichern exportierter Metadaten erstellt haben.

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

5. Suchen Sie die `requirements.txt` Datei im `metadata-migration` Verzeichnis. Wenn Sie bereits über eine Anforderungsdatei für Ihre bestehende Umgebung verfügen, fügen Sie Ihrer Datei die zusätzlichen Anforderungen `requirements.txt` hinzu, die in angegeben sind. Wenn Sie noch keine Anforderungsdatei haben, können Sie einfach die Datei verwenden, die im `metadata-migration` Verzeichnis bereitgestellt wird.
6. Kopieren `export_data.py` Sie in das DAG-Verzeichnis des Amazon S3 S3-Buckets, der mit Ihrer vorhandenen Umgebung verknüpft ist. Wenn Sie aus einer selbstverwalteten Umgebung migrieren, kopieren Sie `export_data.py` in Ihren `/dags` Ordner.
7. Kopieren Sie Ihre Aktualisierung `requirements.txt` in den Amazon S3 S3-Bucket, der mit Ihrer vorhandenen Umgebung verknüpft ist, und bearbeiten Sie dann die Umgebung, um die neue `requirements.txt` Version anzugeben.
8. Nachdem die Umgebung aktualisiert wurde, greifen Sie auf die Apache Airflow-Benutzeroberfläche zu, heben Sie die Pause der `db_export` DAG auf und lösen Sie die Ausführung des Workflows aus.
9. Stellen Sie sicher, dass die Metadaten `data/migration/`*`existing-version`*`_to_`*`new-version`*`/export/` in den `mwaa-migration-`*`{UUID}`* Amazon S3 S3-Bucket exportiert werden, wobei sich jede Tabelle in einer eigenen Datei befindet.

Schritt vier: Importieren der Metadaten in Ihre neue Umgebung

Um die Metadaten in Ihre neue Umgebung zu importieren

1. Ersetzen Sie in `import_data.py` die folgenden Zeichenfolgenwerte durch Ihre Informationen.
   - Für die Migration aus einer bestehenden Amazon MWAA-Umgebung:

     ```
     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`steuert, wie viele Tage Protokolldateien der Workflow in die neue Umgebung kopiert.
   - Für die Migration aus einer selbstverwalteten Umgebung:

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

2. (Optional) `import_data.py` kopiert nur die Protokolle fehlgeschlagener Aufgaben. Wenn Sie alle Aufgabenprotokolle kopieren möchten, ändern Sie die `getDagTasks` Funktion und entfernen Sie `ti.state = 'failed'` sie, wie im folgenden Codeausschnitt gezeigt.

```
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. Ändern Sie die Ausführungsrolle Ihrer neuen Umgebung und fügen Sie die folgende Richtlinie hinzu. Die Berechtigungsrichtlinie ermöglicht Amazon MWAA, aus dem Amazon S3 S3-Bucket zu lesen, in den Sie die Apache Airflow-Metadaten exportiert haben, und Task-Instance-Protokolle aus vorhandenen Protokollgruppen zu kopieren. Ersetzen Sie alle Platzhalter durch Ihre Informationen.

> **Note**
>
> Wenn Sie aus einer selbstverwalteten Umgebung migrieren, müssen Sie die Berechtigungen für CloudWatch Logs aus der Richtlinie entfernen.

```
{
    "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. Kopieren Sie `import_data.py` in das DAG-Verzeichnis des Amazon S3 S3-Buckets, der mit Ihrer neuen Umgebung verknüpft ist, und greifen Sie dann auf die Apache Airflow-Benutzeroberfläche zu, um die `db_import` DAG zu unterbrechen und den Workflow auszulösen. Die neue DAG wird in wenigen Minuten in der Apache Airflow-Benutzeroberfläche angezeigt.
5. Stellen Sie nach Abschluss der DAG-Ausführung sicher, dass Ihr DAG-Ausführungsverlauf kopiert wurde, indem Sie auf jede einzelne DAG zugreifen.

Nächste Schritte

- Weitere Informationen zu verfügbaren Amazon MWAA-Umgebungsklassen und Funktionen finden Sie unter Amazon MWAA-Umgebungsklasse im Amazon MWAA-Benutzerhandbuch.
- Weitere Informationen darüber, wie Amazon MWAA mit Autoscaling-Workern umgeht, finden Sie unter Amazon MWAA Automatic Scaling im Amazon MWAA-Benutzerhandbuch.
- Weitere Informationen zur Amazon MWAA REST API finden Sie unter Amazon MWAA REST API.

Zugehörige Ressourcen

- Apache Airflow-Modelle (Apache Airflow-Dokumentation) — Erfahren Sie mehr über Apache Airflow-Metadaten-Datenbankmodelle.

Migrieren Sie Workloads von AWS Data Pipeline zu Amazon MWAA

AWS hat den AWS Data Pipeline Dienst 2012 gestartet. Zu dieser Zeit wünschten sich Kunden einen Service, mit dem sie eine Vielzahl von Rechenoptionen nutzen konnten, um Daten zwischen verschiedenen Datenquellen zu verschieben. Da sich die Anforderungen an die Datenübertragung im Laufe der Zeit geändert haben, haben sich auch die Lösungen für diese Anforderungen geändert. Sie haben jetzt die Möglichkeit, die Lösung zu wählen, die Ihren Geschäftsanforderungen am besten entspricht. Sie können Ihre Workloads auf jeden der folgenden AWS Dienste migrieren:

- Verwenden Sie Amazon Managed Workflows for Apache Airflow (Amazon MWAA), um die Workflow-Orchestrierung für Apache Airflow zu verwalten.
- Verwenden Sie Step Functions, um Workflows zwischen mehreren AWS-Services zu orchestrieren.
- Wird verwendet AWS Glue , um Apache Spark-Anwendungen auszuführen und zu orchestrieren.

Welche Option Sie wählen, hängt von Ihrer aktuellen Arbeitslast ab AWS Data Pipeline. In diesem Thema wird erklärt, wie Sie von AWS Data Pipeline zu Amazon MWAA migrieren.

Themen

Amazon MWAA wählen

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) ist ein verwalteter Orchestrierungsservice für Apache Airflow, mit dem Sie end-to-end Daten-Pipelines in der Cloud in großem Umfang einrichten und betreiben können. Apache Airflow ist ein Open-Source-Tool, mit dem Sie Abläufe und Aufgaben, sogenannte Workflows, programmgesteuert erstellen, planen und überwachen können. Mit Amazon MWAA können Sie Apache Airflow und die Programmiersprache Python verwenden, um Workflows zu erstellen, ohne die zugrunde liegende Infrastruktur im Hinblick

auf Skalierbarkeit, Verfügbarkeit und Sicherheit verwalten zu müssen. Amazon MWAA passt seine Workflow-Kapazität automatisch an Ihre Bedürfnisse an und ist in AWS Sicherheitsservices integriert, um Ihnen einen schnellen und sicheren Zugriff auf Ihre Daten zu ermöglichen.

Im Folgenden werden einige der Vorteile einer Migration von Amazon AWS Data Pipeline MWAA hervorgehoben:

- Verbesserte Skalierbarkeit und Leistung — Amazon MWAA bietet ein flexibles und skalierbares Framework für die Definition und Ausführung von Workflows. Dies ermöglicht es Benutzern, große und komplexe Workflows mühelos zu handhaben und Funktionen wie dynamische Aufgabenplanung, datengesteuerte Workflows und Parallelität zu nutzen.
- Verbesserte Überwachung und Protokollierung — Amazon MWAA lässt sich in Amazon integrieren CloudWatch , um die Überwachung und Protokollierung Ihrer Workflows zu verbessern. Amazon MWAA sendet automatisch Systemmetriken und Protokolle an. CloudWatch Das bedeutet, dass Sie den Fortschritt und die Leistung Ihrer Workflows in Echtzeit verfolgen und auftretende Probleme identifizieren können.
- Bessere Integrationen mit AWS Diensten und Software von Drittanbietern — Amazon MWAA lässt sich in eine Vielzahl anderer AWS Dienste wie Amazon S3 und Amazon Redshift sowie in Software von Drittanbietern wie DBT, Snowflake und Databricks integrieren. AWS Glue Auf diese Weise können Sie Daten über verschiedene Umgebungen und Dienste hinweg verarbeiten und übertragen.
- Open-Source-Daten-Pipeline-Tool — Amazon MWAA nutzt dasselbe Open-Source-Produkt Apache Airflow, mit dem Sie vertraut sind. Apache Airflow ist ein speziell entwickeltes Tool, das für alle Aspekte des Daten-Pipeline-Managements entwickelt wurde, einschließlich Aufnahme, Verarbeitung, Übertragung, Integritätstests, Qualitätsprüfungen und Sicherstellung der Datenherkunft.
- Moderne und flexible Architektur — Amazon MWAA nutzt Containerisierung und Cloud-native, serverlose Technologien. Das bedeutet mehr Flexibilität und Portabilität sowie eine einfachere Bereitstellung und Verwaltung Ihrer Workflow-Umgebungen.

Architektur und Konzeptkartierung

AWS Data Pipeline und Amazon MWAA haben unterschiedliche Architekturen und Komponenten, die sich auf den Migrationsprozess und die Art und Weise, wie Workflows definiert und ausgeführt werden, auswirken können. In diesem Abschnitt werden die Architektur und die Komponenten für beide Dienste beschrieben und einige der wichtigsten Unterschiede hervorgehoben.

AWS Data Pipeline Sowohl Amazon MWAA als auch Amazon sind vollständig verwaltete Services. Wenn Sie Ihre Workloads zu Amazon MWAA migrieren, müssen Sie sich möglicherweise mit neuen Konzepten vertraut machen, um Ihre bestehenden Workflows mit Apache Airflow zu modellieren. Sie müssen sich jedoch nicht um die Infrastruktur kümmern, Worker patchen und Betriebssystem-Updates verwalten.

In der folgenden Tabelle werden die wichtigsten Konzepte AWS Data Pipeline denen in Amazon MWAA zugeordnet. Verwenden Sie diese Informationen als Ausgangspunkt für die Erstellung eines Migrationsplans.

Konzept	AWS Data Pipeline	Amazon MWAA
Definition der Pipeline	AWS Data Pipeline verwendet eine JSON-basierte Konfigurationsdatei, die den Workflow definiert.	Amazon MWAA verwendet Python-basierte Directed Acyclic Graphs (), die den Workflow definieren. DAGs
Umgebung zur Pipeline-Ausführung	Workflows werden auf EC2 Amazon-Instances ausgeführt. AWS Data Pipeline stellt diese Instances in Ihrem Namen bereit und verwaltet sie.	Amazon MWAA verwendet containerisierte Amazon ECS-Umgebungen zur Ausführung von Aufgaben.
Pipeline-Komponenten	Aktivitäten sind Verarbeitungsaufgaben, die als Teil des Workflows ausgeführt werden.	Operatoren (Aufgaben) sind die grundlegenden Verarbeitungseinheiten eines Workflows.
	Vorbedingungen enthalten bedingte Anweisungen, die erfüllt sein müssen, bevor eine Aktivität ausgeführt werden kann.	Sensoren (Aufgaben) stellen bedingte Anweisungen dar, die darauf warten können, dass eine Ressource oder Aufgabe abgeschlossen ist, bevor sie ausgeführt werden.
	Eine Ressource in AWS Data Pipeline bezieht sich auf die AWS Rechenressource, die	Mithilfe von Aufgaben in einer DAG können Sie eine Vielzahl von Rechenressourcen

Konzept	AWS Data Pipeline	Amazon MWAA
	die in einer Pipeline-Aktivität festgelegte Arbeit ausführt. Amazon EC2 und Amazon EMR sind zwei verfügbare Ressourcen.	definieren, darunter Amazon ECS, Amazon EMR und Amazon EKS. Amazon MWAA führt Python-Operationen auf Workern aus, die auf Amazon ECS ausgeführt werden.
Pipeline-Ausführung	AWS Data Pipeline unterstützt die Planung von Läufen mit regulären ratenbasierten und cron-basierten Mustern.	Amazon MWAA unterstützt die Planung mit Cron-Ausdrücken und Voreinstellungen sowie benutzerdefinierten Zeitplänen.
	Eine Instanz bezieht sich auf jeden Lauf der Pipeline.	Ein DAG-Lauf bezieht sich auf jeden Lauf eines Apache Airflow-Workflows.
	Ein Versuch bezieht sich auf die Wiederholung eines fehlgeschlagenen Vorgangs.	Amazon MWAA unterstützt Wiederholungen, die Sie entweder auf DAG-Ebene oder auf Task-Ebene definieren.

Beispielimplementierungen

In vielen Fällen können Sie Ressourcen, mit denen Sie derzeit orchestrieren, AWS Data Pipeline nach der Migration zu Amazon MWAA wiederverwenden. Die folgende Liste enthält Beispielimplementierungen mit Amazon MWAA für die häufigsten Anwendungsfälle. AWS Data Pipeline

- Einen Amazon EMR-Job ausführen (AWS Workshop)
- Erstellen eines benutzerdefinierten Plugins für Apache Hive und Hadoop (Amazon MWAA-Benutzerhandbuch)
- Daten von S3 nach Redshift kopieren (AWS Workshop)

- Ausführen eines Shell-Skripts auf einer Amazon ECS-Remoteinstanz (Amazon MWAA-Benutzerhandbuch)
- Orchestrierung hybrider (vor Ort) Workflows (Blogbeitrag)

Weitere Tutorials und Beispiele finden Sie im Folgenden:

- Amazon MWAA-Tutorials
- Amazon MWAA-Codebeispiele

Vergleich der Preise

Die Preisgestaltung für AWS Data Pipeline richtet sich nach der Anzahl der Pipelines sowie nach der Nutzungsdauer der einzelnen Pipelines. Aktivitäten, die Sie mehr als einmal am Tag ausführen (hohe Frequenz), kosten 1 USD pro Monat und Aktivität. Aktivitäten, die Sie einmal am Tag oder weniger (niedrige Frequenz) ausführen, kosten 0,60 USD pro Monat und Aktivität. Inaktive Pipelines kosten 1$ pro Pipeline. Weitere Informationen finden Sie in der AWS Data Pipeline Preisliste.

Die Preise für Amazon MWAA basieren auf der Dauer, in der Ihre verwaltete Apache Airflow-Umgebung existiert, sowie auf der zusätzlichen auto Skalierung, die erforderlich ist, um mehr Mitarbeiter bereitzustellen, oder auf der Kapazität des Planers. Sie zahlen für die Nutzung Ihrer Amazon MWAA-Umgebung auf Stundenbasis (mit einer Auflösung von einer Sekunde), wobei die Gebühren je nach Größe der Umgebung variieren. Amazon MWAA skaliert die Anzahl der Worker automatisch auf Grundlage Ihrer Umgebungskonfiguration. AWS berechnet die Kosten für zusätzliche Mitarbeiter separat. Weitere Informationen zu den stündlichen Kosten für die Nutzung verschiedener Amazon MWAA-Umgebungsgrößen finden Sie auf der Seite mit den Amazon MWAA-Preisen.

Zugehörige Ressourcen

Weitere Informationen und bewährte Methoden für die Verwendung von Amazon MWAA finden Sie in den folgenden Ressourcen:

- Die Amazon MWAA API-Referenz
- Überwachen von Dashboards und Alarmen auf Amazon MWAA
- Leistungsoptimierung für Apache Airflow auf Amazon MWAA

Amazon MWAA-Dokumentenverlauf

In der folgenden Tabelle werden wichtige Ergänzungen des Amazon MWAA-Migrationsleitfadens beschrieben, der im März 2022 beginnt.

Änderung	Beschreibung	Datum
Neues Thema zur Migration von Workloads von AWS Data Pipeline zu Amazon MWAA	Es wurden neue Informationen und Anleitungen zur Migration vorhandener Workloads von AWS Data Pipeline zu Amazon MWAA hinzugefügt. Verwenden Sie diese Informationen, um einen Migrationsplan zu entwerfen. • Migrieren Sie Workloads von AWS Data Pipeline zu Amazon MWAA	14. April 2023
Einführung des Amazon MWAA-Migrationsleitfadens	Amazon MWAA bietet jetzt detaillierte Anleitungen zur Migration zu einer neuen Amazon MWAA-Umgebung. Die im Amazon MWAA-Migrationsleitfaden beschriebenen Schritte gelten für die Migration aus einer bestehenden Amazon MWAA-Umgebung oder aus einer selbstverwalteten Apache Airflow-Bereitstellung. • Über den Amazon MWAA-Migrationsleitfaden	7. März 2022

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.