aws

Guida alla migrazione

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: Guida alla migrazione

Table of Contents

Cos'è la guida alla migrazione di Amazon MWAA?

Amazon Managed Workflows for Apache Airflow è un servizio di orchestrazione gestito per Apache Airflow che semplifica la gestione di data pipeline nel cloud su larga scala. Amazon MWAA gestisce il provisioning e la manutenzione continua di Apache Airflow in modo da non doverti più preoccupare di applicare patch, scalare o proteggere le istanze.

Amazon MWAA ridimensiona automaticamente le risorse di elaborazione che eseguono le attività per fornire prestazioni coerenti su richiesta. Amazon MWAA protegge i tuoi dati per impostazione predefinita. I tuoi carichi di lavoro vengono eseguiti nel tuo ambiente cloud isolato e sicuro utilizzando Amazon Virtual Private Cloud. Ciò garantisce che i dati vengano crittografati automaticamente utilizzandoAWS Key Management Service.

Usa questa guida per migrare i tuoi flussi di lavoro Apache Airflow autogestiti su Amazon MWAA o aggiornare un ambiente Amazon MWAA esistente a una nuova versione di Apache Airflow. Il tutorial sulla migrazione descrive come creare o clonare un nuovo ambiente Amazon MWAA, migrare le risorse del flusso di lavoro e trasferire i metadati e i log del flusso di lavoro nel nuovo ambiente.

Prima di provare il tutorial sulla migrazione, ti consigliamo di leggere i seguenti argomenti.

- Architettura di rete
- Considerazioni chiave

Esplora l'architettura MWAA di rete Amazon

La sezione seguente descrive i componenti principali che compongono un MWAA ambiente Amazon e il set di AWS servizi con cui ogni ambiente si integra per gestire le proprie risorse, proteggere i dati e fornire monitoraggio e visibilità per i flussi di lavoro.

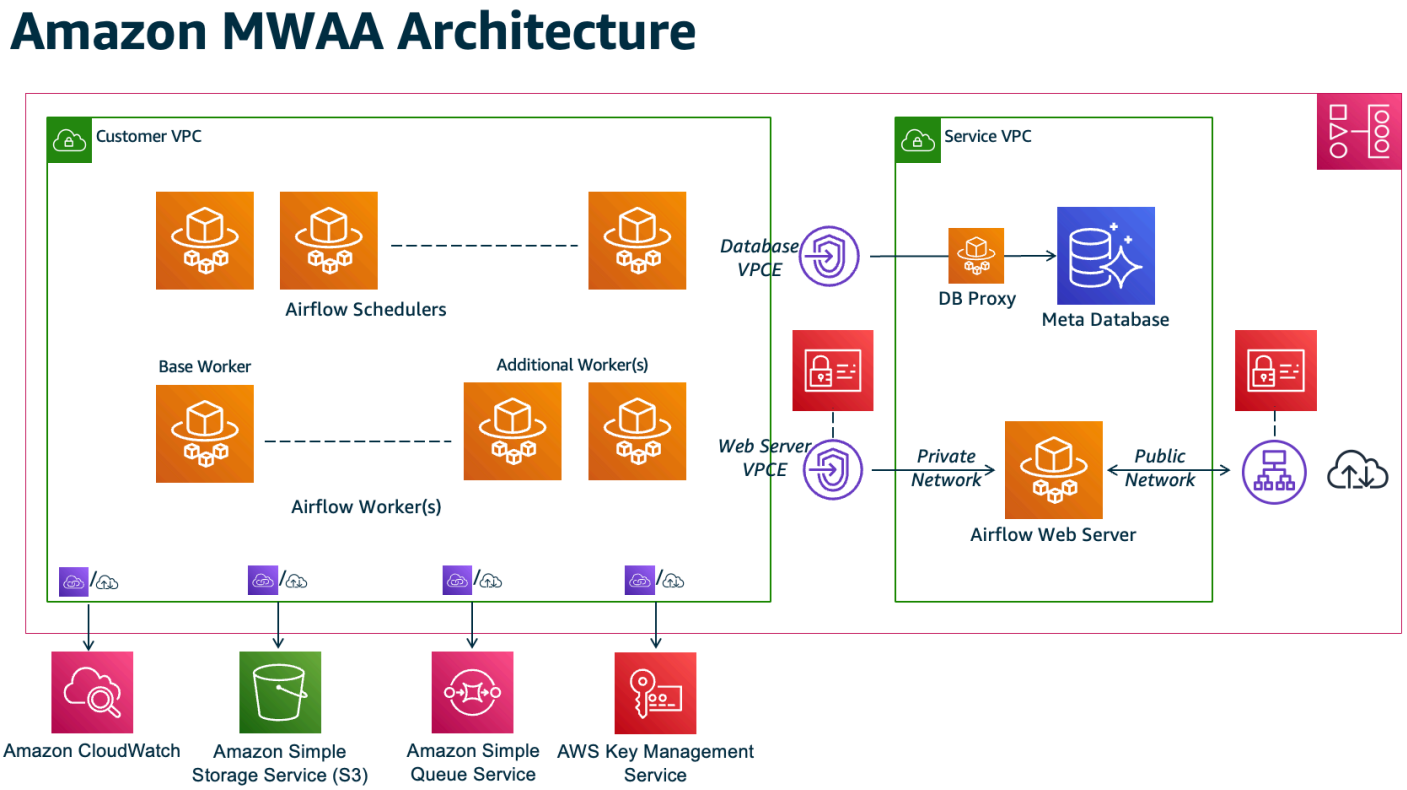
Argomenti

MWAAComponenti Amazon

MWAAGli ambienti Amazon sono costituiti dai seguenti quattro componenti principali:

1. Scheduler: analizza e monitora tutte le attività e mette in coda le DAGs attività per l'esecuzione quando vengono soddisfatte le dipendenze DAG di a. Amazon MWAA distribuisce lo scheduler come AWS Fargate cluster con un minimo di 2 scheduler. Puoi aumentare il numero di pianificatori fino a cinque, a seconda del carico di lavoro. Per ulteriori informazioni sulle classi di MWAA ambiente Amazon, consulta Amazon MWAA Environment Class.
2. Lavoratori: una o più attività Fargate che eseguono le attività pianificate. Il numero di lavoratori per l'ambiente è determinato da un intervallo tra il numero minimo e massimo specificato. Amazon MWAA inizia l'auto-scaling dei lavoratori quando il numero di attività in coda e in esecuzione è superiore a quello che i lavoratori esistenti sono in grado di gestire. Quando le attività in esecuzione e in coda si sommano a zero per più di due minuti, MWAA Amazon riduce il numero di lavoratori al minimo. Per ulteriori informazioni su come Amazon MWAA gestisce gli operatori di auto-scaling, consulta Amazon MWAA automatic scaling.
3. Server Web: esegue l'interfaccia utente web di Apache Airflow. È possibile configurare il server Web con accesso alla rete pubblica o privata. In entrambi i casi, l'accesso agli utenti di Apache Airflow è controllato dalla politica di controllo degli accessi definita in AWS Identity and Access Management ()IAM. Per ulteriori informazioni sulla configurazione delle politiche di IAM accesso per il tuo ambiente, consulta Accedere a un MWAA ambiente Amazon.
4. Database: archivia i metadati sull'ambiente Apache Airflow e sui flussi di lavoro, inclusa la cronologia delle esecuzioni. DAG Il database è un SQL database Aurora Postgre single-tenant gestito e accessibile ai container Fargate AWS di Scheduler e Workers tramite un endpoint Amazon protetto privatamente. VPC

Ogni MWAA ambiente Amazon interagisce anche con una serie di AWS servizi per gestire una varietà di attività, tra cui l'archiviazione e l'accesso DAGs e le dipendenze delle attività, la protezione dei dati inattivi e la registrazione e il monitoraggio dell'ambiente. Il diagramma seguente illustra i diversi componenti di un ambiente AmazonMWAA.



> **Note**
>
> Il servizio Amazon non VPC è condivisoVPC. Amazon ne MWAA crea uno AWS proprietario VPC per ogni ambiente che crei.

- Amazon S3: Amazon MWAA archivia tutte le risorse del flusso di lavoro, ad DAGs esempio requisiti e file di plug-in, in un bucket Amazon S3. Per ulteriori informazioni sulla creazione del bucket come parte della creazione dell'ambiente e sul caricamento delle MWAA risorse Amazon, consulta Create an Amazon S3 bucket for Amazon MWAA nella Amazon User Guide. MWAA
- Amazon SQS — Amazon MWAA utilizza Amazon SQS per mettere in coda le attività del flusso di lavoro con un esecutore Celery.
- AmazonECR: Amazon ECR ospita tutte le immagini di Apache Airflow. Amazon supporta MWAA solo immagini Apache Airflow AWS gestite.

- AWS KMS— Amazon MWAA utilizza AWS KMS per garantire che i tuoi dati siano al sicuro quando sono inattivi. Per impostazione predefinita, Amazon MWAA utilizza AWS KMS chiavi AWS gestite, ma puoi configurare il tuo ambiente per utilizzare la tua chiave gestita dal cliente AWS KMS . Per ulteriori informazioni sull'utilizzo della tua AWS KMS chiave gestita dal cliente, consulta Customer managed keys for Data Encryption nella Amazon MWAA User Guide.
- CloudWatch— Amazon MWAA si integra CloudWatch e fornisce i log e le metriche ambientali di Apache Airflow CloudWatch, consentendoti di monitorare le risorse Amazon MWAA e risolvere i problemi.

Connettività

Il tuo MWAA ambiente Amazon deve accedere a tutti i AWS servizi con cui si integra. Il ruolo di MWAA esecuzione di Amazon controlla il modo in cui viene concesso l'accesso MWAA ad Amazon per connettersi ad altri AWS servizi per tuo conto. Per la connettività di rete, puoi fornire l'accesso pubblico a Internet ad Amazon VPC o creare VPC endpoint Amazon. Per ulteriori informazioni sulla configurazione degli VPC endpoint Amazon (AWS PrivateLink) per il tuo ambiente, consulta Gestire l'accesso agli VPC endpoint su Amazon MWAA nella Amazon MWAA User Guide.

Amazon MWAA installa i requisiti sullo scheduler e sull'operatore. Se i tuoi requisiti provengono da un PyPiarchivio pubblico, il tuo ambiente necessita della connettività a Internet per scaricare le librerie richieste. Per gli ambienti privati, puoi utilizzare un PyPi repository privato o raggruppare le librerie in `.whl`file come plug-in personalizzati per il tuo ambiente.

Quando configuri Apache Airflow in modalità privata, l'interfaccia utente di Apache Airflow può essere accessibile al tuo Amazon solo tramite gli endpoint Amazon. VPC VPC

Per ulteriori informazioni sul networking, consulta Networking nella Amazon MWAA User Guide.

Considerazioni chiave per la migrazione a un nuovo ambiente MWAA

Scopri di più su considerazioni chiave, come l'autenticazione e il ruolo di MWAA esecuzione di Amazon, mentre pianifichi di migrare i tuoi carichi di lavoro Apache Airflow su Amazon. MWAA

Argomenti

Autenticazione

Amazon MWAA utilizza AWS Identity and Access Management (IAM) per controllare l'accesso all'interfaccia utente di Apache Airflow. Devi creare e gestire IAM politiche che concedano agli utenti di Apache Airflow l'autorizzazione ad accedere al server Web e gestirlo. DAGs È possibile gestire sia l'autenticazione che l'autorizzazione per i ruoli predefiniti di Apache Airflow utilizzando IAM account diversi.

Puoi gestire ulteriormente e limitare l'accesso degli utenti di Apache Airflow solo a un sottoinsieme del tuo flusso di lavoro DAGs creando ruoli Airflow personalizzati e mappandoli ai tuoi principali. IAM Per ulteriori informazioni e un step-by-step tutorial, consulta Tutorial: Limitazione dell'accesso di un MWAA utente Amazon a un sottoinsieme di. DAGs

Puoi anche configurare identità federate per accedere ad Amazon. MWAA Per ulteriori informazioni, consulta gli argomenti seguenti.

- MWAAAmbiente Amazon con accesso pubblico: utilizzo di Okta come provider di identità con Amazon MWAA sul AWS Compute Blog.
- MWAAAmbiente Amazon con accesso privato: accesso a un MWAA ambiente Amazon privato utilizzando identità federate.

Ruolo di esecuzione

Amazon MWAA utilizza un ruolo di esecuzione che concede le autorizzazioni al tuo ambiente per accedere ad altri AWS servizi. Puoi fornire al tuo flusso di lavoro l'accesso ai AWS servizi

aggiungendo le autorizzazioni pertinenti al ruolo. Se scegli l'opzione predefinita per creare un nuovo ruolo di esecuzione quando crei l'ambiente per la prima volta, Amazon MWAA assegna le autorizzazioni minime necessarie al ruolo, tranne nel caso di CloudWatch Logs per i quali Amazon MWAA aggiunge automaticamente tutti i gruppi di log.

Una volta creato il ruolo di esecuzione, Amazon MWAA non può gestire le sue politiche di autorizzazione per tuo conto. Per aggiornare il ruolo di esecuzione, devi modificare la politica per aggiungere e rimuovere le autorizzazioni necessarie. Ad esempio, puoi integrare il tuo MWAA ambiente Amazon AWS Secrets Manager come backend per archiviare in modo sicuro segreti e stringhe di connessione da utilizzare nei flussi di lavoro Apache Airflow. A tale scopo, allega la seguente politica di autorizzazione al ruolo di esecuzione del tuo ambiente.

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

L'integrazione con altri AWS servizi segue uno schema simile: aggiungi la politica di autorizzazione pertinente al tuo ruolo di MWAA esecuzione di Amazon, concedendo MWAA ad Amazon l'autorizzazione ad accedere al servizio. Per ulteriori informazioni sulla gestione del ruolo di MWAA esecuzione di Amazon e per visualizzare altri esempi, consulta il ruolo di MWAA esecuzione di Amazon nella Amazon MWAA User Guide.

Esegui la migrazione a un nuovo ambiente Amazon MWAA

Esplora i seguenti passaggi per migrare il tuo carico di lavoro Apache Airflow esistente in un nuovo ambiente Amazon. MWAA Puoi utilizzare questi passaggi per migrare da una versione precedente di Amazon MWAA a una nuova versione o migrare la tua distribuzione di Apache Airflow autogestita su Amazon. MWAA Questo tutorial presuppone che tu stia migrando da un Apache Airflow v1.10.12 esistente a un nuovo MWAA Amazon che esegue Apache Airflow v2.5.1, ma puoi utilizzare le stesse procedure per migrare da o verso diverse versioni di Apache Airflow.

Argomenti

Prerequisiti

Per completare i passaggi e migrare il tuo ambiente, avrai bisogno di quanto segue:

- Una distribuzione di Apache Airflow. Può trattarsi di un MWAA ambiente Amazon autogestito o esistente.
- Docker installato per il tuo sistema operativo locale.
- AWS Command Line Interface versione 2 installata.

Fase uno: creare un nuovo MWAA ambiente Amazon con l'ultima versione supportata di Apache Airflow

Puoi creare un ambiente seguendo i passaggi dettagliati in Getting started with Amazon MWAA nella Amazon MWAA User Guide o utilizzando un AWS CloudFormation modello. Se stai migrando da un

MWAA ambiente Amazon esistente e hai utilizzato un AWS CloudFormation modello per creare il tuo vecchio ambiente, puoi modificare la `AirflowVersion` proprietà per specificare la nuova versione.

```
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
```

In alternativa, se esegui la migrazione da un MWAA ambiente Amazon esistente, puoi copiare il seguente script Python che utilizza AWS SDKfor Python (Boto3) per clonare il tuo ambiente. Puoi anche scaricare lo script.

Script in 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)
```

Fase due: migra le risorse del flusso di lavoro

Apache Airflow v2 è una versione principale. Se state effettuando la migrazione da Apache Airflow v1, dovete preparare le risorse del flusso di lavoro e verificare le modifiche apportate ai vostri requisiti e plugin. DAGs A tale scopo, ti consigliamo di configurare una versione bridge di Apache Airflow sul tuo sistema operativo locale utilizzando Docker e Amazon local runner. MWAA Amazon MWAA local runner fornisce un'utilità di interfaccia a riga di comando (CLI) che replica un MWAA ambiente Amazon localmente.

Ogni volta che modifichi una versione di Apache Airflow, assicurati di fare riferimento a quella corretta nel tuo. `--constraint` URL `requirements.txt`

Per migrare le risorse del flusso di lavoro

1. Crea un fork del aws-mwaa-local-runnerrepository e clona una copia del runner locale di AmazonMWAA.
2. Dai un'occhiata al `v1.10.15` ramo del repository -runner. aws-mwaa-local Apache Airflow ha rilasciato la versione 1.10.15 come versione bridge per facilitare la migrazione ad Apache Airflow v2 e, sebbene Amazon MWAA non supporti la versione 1.10.15, puoi utilizzare Amazon local runner per testare le tue risorse. MWAA

3. Usa lo CLI strumento Amazon MWAA local runner per creare l'immagine Docker ed eseguire Apache Airflow localmente. Per ulteriori informazioni, consulta il runner README locale nel repository. GitHub
4. Utilizzando Apache Airflow in esecuzione localmente, segui i passaggi descritti in Aggiornamento da 1.10 a 2 nel sito Web della documentazione di Apache Airflow.

    a. Per aggiornare le tue `requirements.txt`, segui le best practice consigliate nella sezione Managing Python dependencies, nella Amazon MWAA User Guide.

    b. Se hai combinato gli operatori e i sensori personalizzati con i plugin per l'ambiente Apache Airflow v1.10.12 esistente, spostali nella tua cartella. DAG Per ulteriori informazioni sulle migliori pratiche di gestione dei moduli per Apache Airflow v2+, consulta Module Management nel sito Web della documentazione di Apache Airflow.
5. Dopo aver apportato le modifiche necessarie alle risorse del flusso di lavoro, consultate il `v2.5.1` ramo del repository aws-mwaa-local -runner e testate localmente il flusso di lavoro aggiornato, i requisiti e i plugin personalizzati. DAGs Se stai migrando a una versione diversa di Apache Airflow, puoi invece utilizzare il ramo runner locale appropriato per la tua versione.
6. Dopo aver testato con successo le risorse del flusso di lavoro, copia i tuoi DAGs e i plug-in nel bucket Amazon S3 che hai configurato con il tuo nuovo ambiente Amazon. `requirements.txt` MWAA

Fase tre: Esportazione dei metadati dall'ambiente esistente

Le tabelle di metadati di Apache Airflow `dag`, ad esempio `dag_tag`, `dag_code` vengono compilate automaticamente quando copi DAG i file aggiornati nel bucket Amazon S3 del tuo ambiente e lo scheduler li analizza. Le tabelle relative alle autorizzazioni vengono inoltre compilate automaticamente in base all'autorizzazione del ruolo di esecuzione. IAM Non è necessario migrarle.

È possibile migrare i dati relativi alla DAG cronologia, `variable`, `slot_pool` `sla_miss`, e, se necessario `xcom` `job`, alle tabelle. `log` Il registro delle istanze delle attività viene archiviato nei CloudWatch registri del gruppo di `airflow-`*`{environment_name}`* log. Se si desidera visualizzare i registri delle istanze di attività relativi alle esecuzioni precedenti, è necessario copiarli nel nuovo gruppo di registri di ambiente. Si consiglia di spostare solo i log di pochi giorni per ridurre i costi associati.

Se stai migrando da un MWAA ambiente Amazon esistente, non è possibile accedere direttamente al database dei metadati. Devi eseguire un DAG per esportare i metadati dal tuo MWAA ambiente

Amazon esistente in un bucket Amazon S3 di tua scelta. I seguenti passaggi possono essere utilizzati anche per esportare i metadati di Apache Airflow se stai migrando da un ambiente autogestito.

Dopo l'esportazione dei dati, puoi eseguire un file DAG nel nuovo ambiente per importare i dati. Durante il processo di esportazione e importazione, tutti gli altri DAGs vengono messi in pausa.

Per esportare i metadati dall'ambiente esistente

1. Crea un bucket Amazon S3 utilizzando AWS CLI per archiviare i dati esportati. Sostituisci il `UUID` e `region` con le tue informazioni.

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

> **Note**
>
> Se stai migrando dati sensibili, come le connessioni archiviate in variabili, ti consigliamo di abilitare la crittografia predefinita per il bucket Amazon S3.

2.
> **Note**
>
> Non si applica alla migrazione da un ambiente autogestito.

   Modifica il ruolo di esecuzione dell'ambiente esistente e aggiungi la seguente politica per concedere l'accesso in scrittura al bucket creato nel primo passaggio.

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

```
    ]
}
```

3. Clona il amazon-mwaa-examplesrepository e accedi alla `metadata-migration` sottodirectory per lo scenario di migrazione.

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

4. In`export_data.py`, sostituisci il valore della stringa `S3_BUCKET` con il bucket Amazon S3 che hai creato per archiviare i metadati esportati.

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

5. Individua il `requirements.txt` file nella directory. `metadata-migration` Se disponi già di un file dei requisiti per l'ambiente esistente, aggiungi i requisiti aggiuntivi `requirements.txt` specificati nel file. Se non disponi di un file dei requisiti esistente, puoi semplicemente utilizzare quello fornito nella `metadata-migration` directory.
6. Copia `export_data.py` nella DAG directory del bucket Amazon S3 associato all'ambiente esistente. Se stai migrando da un ambiente autogestito, copialo `export_data.py` nella tua cartella. `/dags`
7. Copia l'aggiornamento `requirements.txt` nel bucket Amazon S3 associato all'ambiente esistente, quindi modifica l'ambiente per specificare la nuova versione. `requirements.txt`
8. Dopo l'aggiornamento dell'ambiente, accedi all'interfaccia utente di Apache Airflow, riattiva la pausa e attiva l'esecuzione del db_`export` DAG flusso di lavoro.
9. Verifica che i metadati vengano esportati `data/migration/`*`existing-version`*`_to_`*`new-version`*`/export/` nel `mwaa-migration-`*`{UUID}`* bucket Amazon S3, con ogni tabella nel proprio file dedicato.

Fase quattro: Importazione dei metadati nel nuovo ambiente

Per importare i metadati nel nuovo ambiente

1. In`import_data.py`, sostituisci i valori di stringa per quanto segue con le tue informazioni.
   - Per la migrazione da un MWAA ambiente Amazon esistente:

```
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_DAYScontrolla per quanti giorni di file di registro il flusso di lavoro copia nel nuovo ambiente.

- Per la migrazione da un ambiente autogestito:

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

2. (Facoltativo) `import_data.py` copia solo i registri delle attività non riuscite. Se desiderate copiare tutti i log delle attività, modificate la `getDagTasks` funzione e rimuovetela `ti.state = 'failed'` come mostrato nel seguente frammento di codice.

```
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. Modifica il ruolo di esecuzione del nuovo ambiente e aggiungi la seguente politica. La politica di autorizzazione consente MWAA ad Amazon di leggere dal bucket Amazon S3 in cui hai esportato i metadati Apache Airflow e di copiare i log delle istanze delle attività dai gruppi di log esistenti. Sostituisci tutti i segnaposto con le tue informazioni.

> **Note**
>
> Se stai migrando da un ambiente autogestito, devi rimuovere le autorizzazioni relative ai CloudWatch registri dalla policy.

```
{
    "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. Copia `import_data.py` DAG nella directory del bucket Amazon S3 associato al nuovo ambiente, quindi accedi all'interfaccia utente di Apache Airflow per riattivare il flusso di lavoro e riavviare il flusso di lavoro. `db_import` DAG Il nuovo DAG verrà visualizzato nell'interfaccia utente di Apache Airflow tra pochi minuti.
5. Al termine dell'DAGesecuzione, verifica che la cronologia delle esecuzioni DAG venga copiata accedendo a ogni utente. DAG

Passaggi successivi

- Per ulteriori informazioni sulle classi e le funzionalità di MWAA ambiente Amazon disponibili, consulta la classe di MWAA ambiente Amazon nella Amazon MWAA User Guide.
- Per ulteriori informazioni su come Amazon MWAA gestisce gli operatori di autoscaling, consulta Amazon MWAAautomatic scaling nella Amazon User Guide. MWAA
- Per ulteriori informazioni su Amazon MWAA RESTAPI, consulta Amazon MWAA REST API.

Risorse correlate

- Modelli Apache Airflow (documentazione Apache Airflow): scopri di più sui modelli di database di metadati Apache Airflow.

Migra i carichi di lavoro da Amazon AWS Data Pipeline MWAA

AWS ha lanciato il AWS Data Pipeline servizio nel 2012. A quel tempo, i clienti desideravano un servizio che consentisse loro di utilizzare una varietà di opzioni di elaborazione per spostare i dati tra diverse fonti di dati. Man mano che le esigenze di trasferimento dei dati sono cambiate nel tempo, sono cambiate anche le soluzioni a tali esigenze. Ora avete la possibilità di scegliere la soluzione che più si avvicina alle vostre esigenze aziendali. Puoi migrare i tuoi carichi di lavoro verso uno qualsiasi dei seguenti servizi: AWS

- Usa Amazon Managed Workflows for Apache Airflow (AmazonMWAA) per gestire l'orchestrazione del flusso di lavoro per Apache Airflow.
- Usa Step Functions per orchestrare i flussi di lavoro tra più persone. Servizi AWS
- Utilizzalo AWS Glue per eseguire e orchestrare le applicazioni Apache Spark.

L'opzione scelta dipende dal carico di lavoro corrente su. AWS Data Pipeline Questo argomento spiega come eseguire la migrazione AWS Data Pipeline da AmazonMWAA.

Argomenti

Scegliere Amazon MWAA

Amazon Managed Workflows for Apache Airflow (AmazonMWAA) è un servizio di orchestrazione gestito per Apache Airflow che consente di configurare e gestire pipeline di end-to-end dati nel cloud su larga scala. Apache Airflow è uno strumento open source utilizzato per creare, pianificare e monitorare in modo programmatico sequenze di processi e attività denominate flussi di lavoro. Con AmazonMWAA, puoi utilizzare Apache Airflow e il linguaggio di programmazione Python per creare flussi di lavoro senza dover gestire l'infrastruttura sottostante per scalabilità, disponibilità e sicurezza. Amazon ridimensiona MWAA automaticamente la capacità del flusso di lavoro per soddisfare le

tue esigenze ed è integrato con i servizi di AWS sicurezza per aiutarti a fornirti un accesso rapido e sicuro ai tuoi dati.

Di seguito vengono evidenziati alcuni dei vantaggi della migrazione AWS Data Pipeline da AmazonMWAA:

- Scalabilità e prestazioni migliorate: Amazon MWAA fornisce un framework flessibile e scalabile per la definizione e l'esecuzione dei flussi di lavoro. Ciò consente agli utenti di gestire flussi di lavoro ampi e complessi con facilità e di sfruttare funzionalità come la pianificazione dinamica delle attività, i flussi di lavoro basati sui dati e il parallelismo.
- Monitoraggio e registrazione migliorati: Amazon MWAA si integra con Amazon CloudWatch per migliorare il monitoraggio e la registrazione dei flussi di lavoro. Amazon invia MWAA automaticamente i parametri e i log di sistema a. CloudWatch Ciò significa che puoi monitorare i progressi e le prestazioni dei tuoi flussi di lavoro in tempo reale e identificare eventuali problemi che si presentano.
- Migliori integrazioni con AWS servizi e software di terze parti : Amazon MWAA si integra con una varietà di altri AWS servizi, come Amazon S3 e Amazon Redshift AWS Glue, oltre a software di terze parti come Snowflake e Databricks DBT. Ciò consente di elaborare e trasferire dati tra diversi ambienti e servizi.
- Strumento di pipeline di dati open source: Amazon MWAA sfrutta lo stesso prodotto open source Apache Airflow che conosci. Apache Airflow è uno strumento appositamente progettato per gestire tutti gli aspetti della gestione della pipeline di dati, tra cui l'ingestione, l'elaborazione, il trasferimento, i test di integrità, i controlli di qualità e la garanzia della derivazione dei dati.
- Architettura moderna e flessibile: Amazon MWAA sfrutta la containerizzazione e le tecnologie serverless native per il cloud. Ciò significa maggiore flessibilità e portabilità, nonché una distribuzione e una gestione più semplici degli ambienti di flusso di lavoro.

Architettura e mappatura concettuale

AWS Data Pipeline e Amazon MWAA hanno architetture e componenti diversi, che possono influire sul processo di migrazione e sul modo in cui i flussi di lavoro vengono definiti ed eseguiti. Questa sezione presenta una panoramica dell'architettura e dei componenti di entrambi i servizi ed evidenzia alcune delle differenze principali.

Entrambi AWS Data Pipeline e Amazon MWAA sono servizi completamente gestiti. Quando migri i tuoi carichi di lavoro su Amazon, MWAA potresti aver bisogno di imparare nuovi concetti per

modellare i flussi di lavoro esistenti utilizzando Apache Airflow. Tuttavia, non sarà necessario gestire l'infrastruttura, applicare patch worker e gestire gli aggiornamenti del sistema operativo.

La tabella seguente associa i concetti chiave AWS Data Pipeline a quelli di AmazonMWAA. Usa queste informazioni come punto di partenza per progettare un piano di migrazione.

Concetto	AWS Data Pipeline	Amazon MWAA
Definizione della pipeline	AWS Data Pipeline utilizza un file di configurazione JSON basato che definisce il flusso di lavoro.	Amazon MWAA utilizza Directed Acyclic Graphs () DAGs basato su Python che definiscono il flusso di lavoro.
Ambiente di esecuzione della pipeline	I flussi di lavoro vengono eseguiti su EC2 istanze Amazon. AWS Data Pipeline effettua il provisioning e gestisce queste istanze per tuo conto.	Amazon MWAA utilizza ambienti ECS containerizzati Amazon per eseguire attività.
Componenti della pipeline	Le attività sono operazioni di elaborazione eseguite come parte del flusso di lavoro.	Gli operatori (attività) sono le unità di elaborazione fondamentali di un flusso di lavoro.
	Le precondizioni contengon o istruzioni condizionali che devono essere vere prima che un'attività possa essere eseguita.	I sensori (attività) rappresen tano istruzioni condizionali che possono attendere il completamento di una risorsa o di un'attività prima di essere eseguite.
	Una risorsa in AWS Data Pipeline si riferisce alla risorsa di AWS calcolo che esegue il lavoro specificato da un'attivi tà di pipeline. Amazon EC2	Utilizzando le attività in aDAG, puoi definire una varietà di risorse di elaborazione, tra cui Amazon ECSEMR, Amazon e AmazonEKS. Amazon MWAA esegue operazioni Python su

Concetto	AWS Data Pipeline	Amazon MWAA
	e Amazon EMR sono due risorse disponibili.	worker eseguiti su Amazon. ECS
Esecuzione pipeline	AWS Data Pipeline supporta la pianificazione di esecuzioni con schemi regolari basati sulla frequenza e su cronn.	Amazon MWAA supporta la pianificazione con espressioni cron e preimpostazioni, nonché orari personalizzati.
	Un'istanza si riferisce a ogni esecuzione della pipeline.	Un'DAGesecuzione si riferisce a ogni esecuzione di un flusso di lavoro Apache Airflow.
	Un tentativo si riferisce a un nuovo tentativo di eseguire un'operazione non riuscita.	Amazon MWAA supporta nuovi tentativi definiti a livello o a DAG livello di attività.

Implementazioni esemplificative

In molti casi sarai in grado di riutilizzare le risorse con cui stai attualmente orchestrando AWS Data Pipeline dopo la migrazione ad Amazon. MWAA L'elenco seguente contiene esempi di implementazioni che utilizzano Amazon MWAA per i casi AWS Data Pipeline d'uso più comuni.

- Esecuzione di un EMR lavoro su Amazon (AWS workshop)
- Creazione di un plug-in personalizzato per Apache Hive e Hadoop (Amazon MWAA User Guide)
- Copiare i dati da S3 a Redshift (workshop)AWS
- Esecuzione di uno script di shell su un'ECSistanza Amazon remota (Amazon MWAA User Guide)
- Orchestrazione di flussi di lavoro ibridi (on-premise) (post sul blog)

Per ulteriori tutorial ed esempi, consulta quanto segue:

- MWAATutorial Amazon
- Esempi di MWAA codice Amazon

Confronto dei prezzi

I prezzi di AWS Data Pipeline si basano sul numero di pipeline e sull'utilizzo di ciascuna pipeline. Le attività eseguite più di una volta al giorno (alta frequenza) costano 1 USD al mese per attività. Le attività che svolgi una volta al giorno o meno (bassa frequenza) costano 0,60 USD al mese per attività. Le pipeline inattive hanno un prezzo di 1 USD per pipeline. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Data Pipeline.

I prezzi di Amazon MWAA si basano sulla durata del periodo di esistenza dell'ambiente Apache Airflow gestito e su qualsiasi ulteriore scalabilità automatica richiesta per fornire più dipendenti o capacità di pianificazione. Paghi per MWAA l'utilizzo del tuo ambiente Amazon su base oraria (fatturato con una risoluzione di un secondo), con tariffe variabili a seconda delle dimensioni dell'ambiente. Amazon MWAA ridimensiona automaticamente il numero di lavoratori in base alla configurazione dell'ambiente. AWS calcola separatamente il costo dei lavoratori aggiuntivi. Per ulteriori informazioni sul costo orario dell'utilizzo di MWAA ambienti Amazon di diverse dimensioni, consulta la pagina MWAAdei prezzi di Amazon.

Risorse correlate

Per ulteriori informazioni e best practice per l'utilizzo di AmazonMWAA, consulta le seguenti risorse:

- Il MWAA API riferimento Amazon
- Pannelli di controllo e allarmi su Amazon MWAA
- Ottimizzazione delle prestazioni per Apache Airflow su Amazon MWAA

Cronologia documento documento

Nella seguente tabella sono descritte le aggiunte significative apportate alla guida Amazon MWAA a partire da marzo 2022.

Modifica	Descrizione	Data
Nuovo argomento sulla migrazione dei carichi di lavoroAWS Data Pipeline da Amazon MWAA	Sono state aggiunte nuove informazioni e linee guida sulla migrazione dei carichi di lavoro esistentiAWS Data Pipeline da Amazon MWAA. Utilizza queste informazioni per progettare un piano di migrazione. • Migra i carichi di lavoro da Amazon AWS Data Pipeline MWAA	14 aprile 2023
Lancio della guida alla migrazione di Amazon MWAA	Amazon MWAA offre ora una guida dettagliata sulla migrazione verso un nuovo ambiente Amazon MWAA. I passaggi descritti nella Amazon MWAA Migration Guide si applicano alla migrazione da un ambiente Amazon MWAA esistente o da una distribuzione Apache Airflow autogestita. • Informazioni sulla guida alla migrazione di Amazon MWAA	7 marzo 2022

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.