



Guida alla migrazione

Amazon Managed Workflows for Apache Airflow



Amazon Managed Workflows for Apache Airflow: Guida alla migrazione

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cos'è la guida alla migrazione?	1
Architettura di rete	2
Componenti Amazon MWAA	2
Connettività	4
Considerazioni chiave	5
Autenticazione	5
Ruolo di esecuzione	5
Migrazione a un nuovo ambiente Amazon MWAA	7
Prerequisiti	7
Fase 1: creare un nuovo ambiente	7
Fase due: Migrare le risorse del flusso di lavoro	14
Fase 3: esportazione dei metadati	15
Fase 4: importazione dei metadati di importazione dei metadati	17
Fasi successive	19
Risorse correlate	20
Migrazione dei carichi di lavoroAWS Data Pipeline da Amazon MWAA	21
Scegliere Amazon MWAA	21
Architettura e mappatura concettuale	22
Implementazioni di esempio	24
Confronto dei prezzi	25
Risorse correlate	25
Cronologia dei documenti	26
.....	xxvii

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 utilizzando AWS 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](#)

Architettura di rete Amazon MWAA

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

Argomenti

- [Componenti Amazon MWAA](#)
- [Connettività](#)

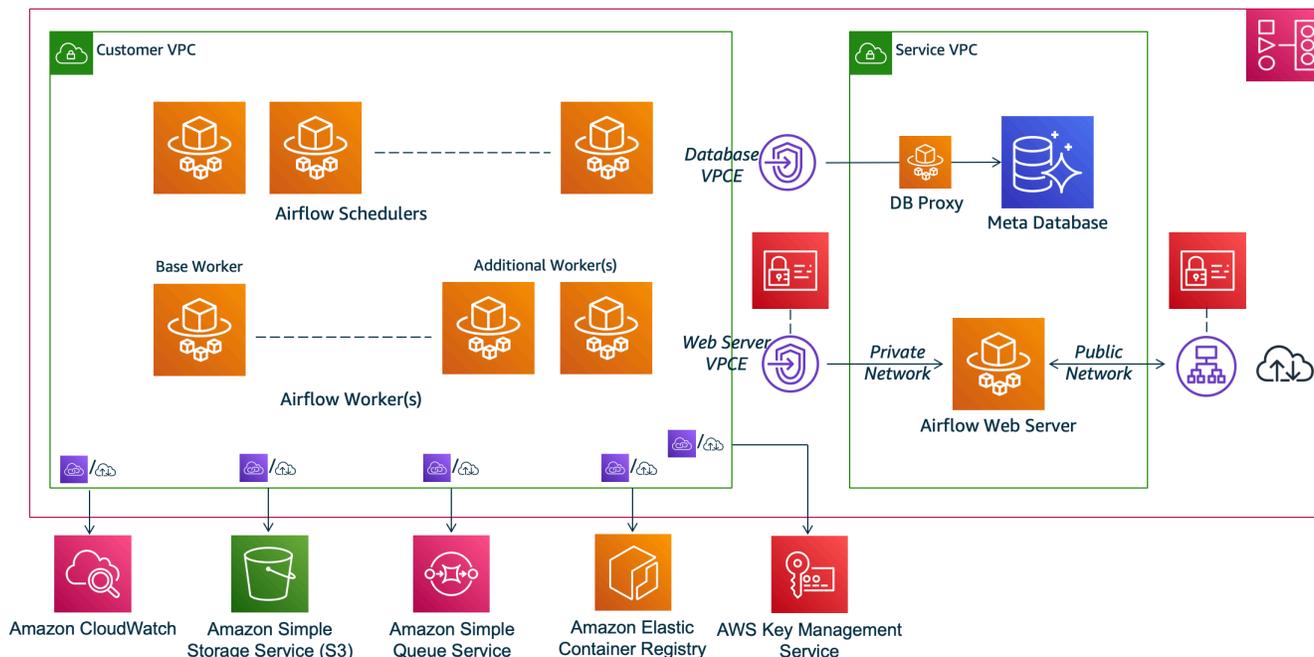
Componenti Amazon MWAA

Gli ambienti Amazon MWAA sono costituiti dai seguenti quattro componenti principali:

1. **Scheduler:** analizza e monitora tutti i DAG e mette in coda le attività per l'esecuzione quando vengono soddisfatte le dipendenze di un DAG. Amazon MWAA distribuisce lo scheduler come cluster con un minimo di 2 scheduler. AWS Fargate Puoi aumentare il numero di pianificatori fino a cinque, a seconda del carico di lavoro. Per ulteriori informazioni sulle classi di ambiente Amazon MWAA, 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 quelle in coda si sommano a zero per più di due minuti, Amazon MWAA riduce il numero di lavoratori al minimo. [Per ulteriori informazioni su come Amazon MWAA gestisce i lavoratori con scalabilità automatica, 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 policy di accesso IAM per il tuo ambiente, consulta [Accedere a un ambiente Amazon MWAA](#).
4. **Database:** archivia i metadati sull'ambiente Apache Airflow e sui flussi di lavoro, inclusa la cronologia di esecuzione di DAG. Il database è un database Aurora PostgreSQL single-tenant gestito AWS e accessibile ai contenitori Fargate di Scheduler e Workers tramite un endpoint Amazon VPC protetto privatamente.

Ogni ambiente Amazon MWAA interagisce anche con una serie di AWS servizi per gestire una serie di attività, tra cui l'archiviazione e l'accesso ai DAG e alle 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 Amazon MWAA.

Amazon MWAA Architecture



Note

Il servizio Amazon VPC non è un VPC condiviso. Amazon MWAA crea un AWS VPC di proprietà per ogni ambiente che crei.

- Amazon S3 — Amazon MWAA archivia tutte le risorse del flusso di lavoro, come DAG, 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 risorse Amazon MWAA, consulta [Create an Amazon S3 bucket per Amazon MWAA nella Amazon MWAA User Guide](#).
- Amazon SQS: [Amazon MWAA utilizza Amazon SQS per mettere in coda le attività del flusso di lavoro con un esecutore Celery.](#)
- Amazon ECR — Amazon ECR ospita tutte le immagini Apache Airflow. Amazon MWAA supporta solo immagini Apache Airflow gestite. AWS

- **AWS KMS**— Amazon MWAA lo utilizza AWS KMS per garantire la sicurezza dei dati quando sono inattivi. [Per impostazione predefinita, Amazon MWAA utilizza AWS KMS chiavi AWS gestite, ma è possibile configurare l'ambiente in modo da utilizzare una 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 log e parametri ambientali di Apache Airflow, consentendoti di monitorare le risorse Amazon MWAA e risolvere CloudWatch i problemi.

Connettività

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

Amazon MWAA installa i requisiti sullo scheduler e sul worker. Se i requisiti provengono da un [PyPi](#) archivio pubblico, l'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 [.whlfile](#) 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 VPC solo tramite gli endpoint Amazon VPC.

Per ulteriori informazioni sul networking, consulta [Networking in the Amazon MWAA User Guide.](#)

Considerazioni chiave

Consulta i seguenti argomenti prima di migrare a un nuovo ambiente Amazon MWAA.

Argomenti

- [Autenticazione](#)
- [Ruolo di esecuzione](#)

Autenticazione

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

Puoi gestire ulteriormente e limitare l'accesso degli utenti di Apache Airflow solo a un sottoinsieme dei DAG del flusso di lavoro 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 utente Amazon MWAA a un sottoinsieme di DAG](#).

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

- Ambiente Amazon MWAA con accesso pubblico: [utilizzo di Okta come provider di identità con Amazon MWAA](#) sul Compute Blog.AWS
- Ambiente Amazon MWAA con accesso privato: accesso a un ambiente [Amazon MWAA privato utilizzando identità federate](#).

Ruolo di esecuzione

Amazon MWAA utilizza un ruolo di esecuzione che concede autorizzazioni all'ambiente per accedere ad altri servizi. AWS 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 dei log per CloudWatch 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 policy per aggiungere e rimuovere le autorizzazioni necessarie. Ad esempio, puoi [integrare il tuo ambiente Amazon MWAA 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 esecuzione di Amazon MWAA, concedendo l'autorizzazione ad Amazon MWAA per accedere al servizio. Per ulteriori informazioni sulla gestione del ruolo di esecuzione di Amazon MWAA e per visualizzare altri esempi, consulta il ruolo di [esecuzione di Amazon MWAA nella Amazon MWAA User Guide](#).

Migrazione a un nuovo ambiente Amazon MWAA

L'argomento seguente descrive i passaggi per migrare il carico di lavoro Apache Airflow esistente in un nuovo ambiente Amazon MWAA. Puoi utilizzare i seguenti passaggi per migrare da una versione precedente di Amazon MWAA a una nuova versione o migrare la tua distribuzione autogestita di Apache Airflow ad Amazon MWAA. Questo tutorial presuppone la migrazione da un Apache Airflow v1.10.12 esistente a un nuovo Amazon MWAA con Apache Airflow v2.5.1, ma puoi utilizzare le stesse procedure per migrare da o verso diverse versioni di Apache Airflow.

Argomenti

- [Prerequisiti](#)
- [Fase uno: creare un nuovo ambiente Amazon MWAA con l'ultima versione supportata di Apache Airflow](#)
- [Fase due: Migrare le risorse del flusso di lavoro](#)
- [Fase tre: Esportazione dei metadati dall'ambiente esistente](#)
- [Fase quattro: importazione dei metadati nel nuovo ambiente](#)
- [Fasi successive](#)
- [Risorse correlate](#)

Prerequisiti

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

- Un'implementazione di Apache Airflow. Può trattarsi di un ambiente Amazon MWAA autogestito o esistente.
- [Docker installato](#) per il tuo sistema operativo locale.
- [AWS Command Line Interface versione 2](#) installata.

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

Puoi creare un ambiente seguendo i passaggi dettagliati in Guida [introduttiva ad Amazon MWAA](#) nella Guida per l'utente di Amazon MWAA o utilizzando un modello. AWS CloudFormation Se

stai eseguendo la migrazione da un ambiente Amazon MWAA 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 ambiente Amazon MWAA esistente, puoi copiare il seguente script Python che utilizza l'[AWSSDK for Python \(Boto3\) per clonare](#) il tuo ambiente. Puoi anche [scaricare lo 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 dependences 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()<>+|\\)))*\\)|[^\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
    ...
    mwaas = boto3.client('mwaas', 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#MWSAA.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/aws-labs/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 MWA 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/AmazonMWA-MyEnvironment-ExecutionRole")
    parser.add_argument('--envnamenew', type=validate_envname, required=True,
help="name of the destination MWA 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_mwa_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: Migrare le risorse del flusso di lavoro

Apache Airflow v2 è una versione principale. Se stai eseguendo la migrazione da Apache Airflow v1, devi preparare le risorse del flusso di lavoro e verificare le modifiche apportate ai DAG, ai requisiti e ai plugin. A tal fine, ti consigliamo di configurare una versione bridge di Apache Airflow sul tuo sistema operativo locale utilizzando Docker e [Amazon](#) MWAA local runner. Il runner locale Amazon MWAA fornisce un'utilità di interfaccia a riga di comando (CLI) che replica localmente un ambiente Amazon MWAA.

Ogni volta che modifichi le versioni di Apache Airflow, assicurati di fare [riferimento all'--constraintURL corretto](#) nel tuo `requirements.txt`

Per migrare le risorse del tuo flusso di lavoro

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

3. Usa lo strumento Amazon MWAA Local Runner CLI per creare l'immagine Docker ed eseguire Apache Airflow localmente. Per ulteriori informazioni, consulta il file [README](#) dell'GitHubarchivio.
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 tuerequirements.txt, segui le best practice consigliate nella sezione [Gestione delle dipendenze di Python](#), nella Guida per l'utente di Amazon MWAA.
 - b. Se hai abbinato operatori e sensori personalizzati ai plugin per l'ambiente Apache Airflow v1.10.12 esistente, spostali nella cartella DAG. Per ulteriori informazioni sulle migliori pratiche di gestione dei moduli per Apache Airflow v2+, vedere [Gestione dei moduli](#) nel sito Web della documentazione di Apache Airflow.
5. Dopo aver apportato le modifiche necessarie alle risorse del flusso di lavoro, controlla il v2.5.1 ramo del aws-mwaa-local-runner repository e verifica localmente i DAG, i requisiti e i plugin personalizzati del flusso di lavoro aggiornati. 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 DAG e i plugin nel bucket Amazon S3 che hai configurato con il tuo nuovo ambiente Amazon MWAA.
requirements.txt

Fase tre: Esportazione dei metadati dall'ambiente esistente

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

È possibile migrare i dati relativi alla cronologia del DAG,variable,slot_pool, esla_miss, se necessario xcomjob, e log alle tabelle. Il registro delle istanze dell'operazione viene archiviato nei CloudWatch registri del gruppo di airflow-*{environment_name}* log. Se si desidera visualizzare i registri delle istanze delle attività per le esecuzioni precedenti, è necessario copiarli nel nuovo gruppo di log di ambiente. Ti consigliamo di spostare solo alcuni giorni di log per ridurre i costi associati.

Se esegui la migrazione da un ambiente Amazon MWAA esistente, non è possibile accedere direttamente al database dei metadati. È necessario eseguire un DAG per esportare i metadati

dall'ambiente Amazon S3 MWAAs di tua scelta. I seguenti passaggi possono essere utilizzati anche per esportare i metadati di Apache Airflow se si esegue la migrazione da un ambiente autogestito.

Dopo l'esportazione dei dati, è possibile eseguire un DAG nel nuovo ambiente per importare i dati. Durante il processo di esportazione e importazione, tutti gli altri DAG vengono sospesi.

Per esportare i metadati dall'ambiente esistente

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

```
$ aws s3api create-bucket \  
  --bucket mwaas-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:::mwaas-migration-{UUID}/*"  
      ]  
    }  
  ]  
}
```

```
]
}
```

3. Clona il [amazon-mwaa-examples](https://github.com/aws-samples/amazon-mwaa-examples) repository e accedi alla metadata-migration sottodirectory per il tuo 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. Nelexport_data.py, sostituisci il valore della stringa per 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 metadata-migration directory. 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 directory DAG del bucket Amazon S3 associato al tuo ambiente esistente. Se stai migrando da un ambiente autogestito, copia nella tua export_data.py / dags cartella.
7. Copia requirements.txt l'aggiornamento nel bucket Amazon S3 associato al tuo ambiente esistente, quindi modifica l'ambiente per specificare la nuova requirements.txt versione.
8. Dopo l'aggiornamento dell'ambiente, accedi all'interfaccia utente di Apache Airflow, riattiva il db_export DAG e attiva l'esecuzione del flusso di lavoro.
9. Verifica che i metadati siano esportati data/migration/*existing-version_to_new-version*/export/ nel bucket mwaa-migration-*{UUID}* 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. Nelimport_data.py, sostituisci i valori della stringa per quanto segue con le tue informazioni.
 - Per la migrazione da un ambiente Amazon MWAA 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_DAYS controlla per quanti giorni di file di registro il flusso di lavoro copia nel nuovo ambiente.

- Per la migrazione da un ambiente gestito dal cliente in un ambiente gestito dal cliente:

```

S3_BUCKET = 'mwa-migration-{UUID}'
NEW_ENV_NAME='{new_environment_name}'

```

2. (Facoltativo) `import_data.py` copia solo i registri delle attività non riuscite. Se desideri copiare tutti i registri delle attività, modifica la `getDagTasks` funzione e rimuovila `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 tuo nuovo ambiente e aggiungi la seguente politica. La politica di autorizzazione consente ad Amazon MWA di leggere dal bucket Amazon S3 in cui sono stati esportati i metadati di Apache Airflow e di copiare i log delle istanze di attività da gruppi di log esistenti. Sostituisci tutti i segnaposti con le tue informazioni.

Note

Se si esegue la migrazione da un ambiente autogestito, è necessario rimuovere le autorizzazioni relative ai CloudWatch log 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` nella directory DAG del bucket Amazon S3 associato al tuo nuovo ambiente, quindi accedi all'interfaccia utente di Apache Airflow per riattivare il DAG e attivare il flusso di lavoro. `db_import` Il nuovo DAG apparirà nell'interfaccia utente di Apache Airflow tra pochi minuti.
5. Al termine dell'esecuzione del DAG, verifica che la cronologia delle esecuzioni del DAG sia stata copiata accedendo a ogni singolo DAG.

Fasi successive

- Per ulteriori informazioni sulle classi e le funzionalità di ambiente Amazon MWAA disponibili, consulta la [classe di ambiente Amazon MWAA nella Guida per l'utente di Amazon MWAA](#).
- Per ulteriori informazioni su come Amazon MWAA gestisce gli addetti alla scalabilità automatica, consulta la [scalabilità automatica di Amazon MWAA nella Guida per l'utente di Amazon MWAA](#).
- Per ulteriori informazioni su Amazon MWAA REST API, consulta l'API REST di [Amazon MWAA](#).

Risorse correlate

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

Migrazione dei carichi di lavoro AWS Data Pipeline da Amazon 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 per tali esigenze. Ora hai la possibilità di scegliere la soluzione più adatta alle tue esigenze aziendali. Puoi migrare i tuoi carichi di lavoro verso uno dei seguenti AWS servizi:

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

L'opzione scelta varia a seconda del carico di lavoro attuale AWS Data Pipeline. In questo argomento viene descritto come AWS Data Pipeline migrare da Amazon MWAA.

Argomenti

- [Scegliere Amazon MWAA](#)
- [Architettura e mappatura concettuale](#)
- [Implementazioni di esempio](#)
- [Confronto dei prezzi](#)
- [Risorse correlate](#)

Scegliere Amazon MWAA

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) è un servizio di orchestrazione gestito per Apache Airflow che semplifica la configurazione e l'utilizzo di data pipeline end-to-end 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 denominati flussi di lavoro. Con Amazon MWAA, puoi usare 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 MWAA ridimensiona automaticamente la capacità del flusso di lavoro per soddisfare le tue

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

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

- **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 MWAA invia automaticamente le metriche e i log di sistema a CloudWatch. Ciò significa che puoi monitorare l'avanzamento e le prestazioni dei tuoi flussi di lavoro in tempo reale e identificare eventuali problemi che si presentano.
- **Migliore integrazione 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, nonché con software di terze parti come [DBT](#), [Snowflake](#) e [Databricks](#). AWS Glue. Ciò consente di elaborare e trasferire dati in ambienti e servizi diversi.
- **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 del lignaggio 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, oltre a semplificare l'implementazione e la gestione 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 e evidenzia alcune delle principali differenze.

Entrambi AWS Data Pipeline e Amazon MWAA sono servizi completamente gestiti. Quando esegui la migrazione dei carichi di lavoro su Amazon MWAA, potresti dover apprendere nuovi concetti per

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

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

Concetto	AWS Data Pipeline	Amazon MWAA
Definizione di pipeline	AWS Data Pipeline utilizza un file di configurazione basato su JSON che definisce il flusso di lavoro.	Amazon MWAA utilizza grafici aciclici diretti (DAG) basati su Python che definiscono il flusso di lavoro.
Ambienti di esecuzione della pipeline	I flussi di lavoro vengono eseguiti su istanze Amazon EC2. AWS Data Pipeline provvede e gestisce queste istanze per tuo conto.	Amazon MWAA utilizza ambienti containerizzati Amazon ECS per eseguire attività.
Componenti della pipeline	Le attività sono attività 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 contengono istruzioni condizionali che devono essere vere prima che un'attività possa essere eseguita.	I sensori (attività) rappresentano istruzioni condizionali che possono attendere il completamento di una risorsa o di un'attività prima dell'esecuzione.
	Una risorsa in AWS Data Pipeline si riferisce alla risorsa di AWS calcolo che esegue il lavoro specificato da un'attività della pipeline. Amazon EC2 e Amazon EMR sono due risorse disponibili.	Utilizzando le attività in un DAG, puoi definire una varietà di risorse di elaborazione, tra cui Amazon ECS, Amazon EMR e Amazon EKS. Amazon MWAA esegue operazioni Python sui lavoratori che

Concetto	AWS Data Pipeline	Amazon MWAA
		vengono eseguiti su Amazon ECS.
Esecuzione pipeline	AWS Data Pipeline supporta la pianificazione delle corse con schemi regolari basati sulla tariffa e sui cron.	Amazon MWAA supporta la pianificazione con espressioni cron e preimpostazioni, nonché orari personalizzati.
	Un'istanza si riferisce a ogni esecuzione della pipeline.	Un' esecuzione DAG si riferisce a ogni esecuzione di un flusso di lavoro di Apache Airflow.
	Un tentativo si riferisce a un nuovo tentativo di un'operazione non riuscita.	Amazon MWAA supporta nuovi tentativi definiti a livello di DAG o a livello di attività.

Implementazioni di esempio

In molti casi potrai 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 d'uso più comuni di AWS Data Pipeline.

- [Esecuzione di un job Amazon EMR](#) (AWSworkshop)
- [Creazione di un plug-in personalizzato per Apache Hive e Hadoop](#) (Amazon MWAA User Guide)
- [Copiare dati da S3 a Redshift](#) (AWSworkshop)
- [Esecuzione di uno script shell su un'istanza Amazon ECS remota](#) (Amazon MWAA User Guide)
- [Orchestrare flussi di lavoro ibridi \(locali\)](#) (post sul blog)

Per ulteriori tutorial ed esempi, consulta la seguente documentazione:

- [Tutorial Amazon MWAA](#)
- [Esempi di codice Amazon MWAA](#)

Confronto dei prezzi

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

I prezzi di Amazon MWAA si basano sulla durata dell'ambiente Apache Airflow gestito e sull'eventuale scalabilità auto aggiuntiva necessaria per fornire più dipendenti o capacità di pianificazione. L'utilizzo dell'ambiente Amazon MWAA viene pagato 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 ambienti Amazon MWAA di varie dimensioni, consulta la pagina [dei prezzi di Amazon MWAA](#).

Risorse correlate

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

- [Il riferimento all'API Amazon MWAA](#)
- [Monitoraggio di dashboard 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	<p>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.</p> <ul style="list-style-type: none">• Migrazione dei carichi di lavoroAWS Data Pipeline da Amazon MWAA	14 aprile 2023
Lancio della guida alla migrazione di Amazon MWAA	<p>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.</p> <ul style="list-style-type: none">• 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.