aws

Guia de migração

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: Guia de migração

Table of Contents

O que é o Guia de migração do Amazon MWAA?

O Amazon Managed Workflows for Apache Airflow é um serviço de orquestração gerenciado para o Apache Airflow que permite que você opere pipelines de datas na nuvem em escala. O Amazon MWAA gerencia o provisionamento e a manutenção contínua do Apache Airflow para que você não precise mais se preocupar com a aplicação de patches, a escalabilidade ou a proteção de instâncias.

O Amazon MWAA realiza o ajuste de escala automático dos recursos computacionais que executam tarefas para fornecer desempenho consistente sob demanda. O Amazon MWAA protege seus dados por padrão. Suas workloads são executadas em seu próprio ambiente de nuvem isolado e seguro usando a Amazon Virtual Private Cloud. Isso garante que os dados sejam criptografados automaticamente usando AWS Key Management Service.

Use este guia para migrar seus fluxos de trabalho autogerenciados do Apache Airflow para o Amazon MWAA ou atualize um ambiente existente do Amazon MWAA para uma nova versão do Apache Airflow. O tutorial de migração descreve como você pode criar ou clonar um novo ambiente do Amazon MWAA, migrar seus recursos de fluxo de trabalho e transferir seus metadados e logs de fluxo de trabalho para seu novo ambiente.

Antes de tentar o tutorial de migração, recomendamos revisar os tópicos a seguir.

- Arquitetura de rede
- Considerações importantes

Como explorar a arquitetura de rede do Amazon MWAA

A seção a seguir descreve os principais componentes que compõem um ambiente Amazon MWAA e o conjunto de serviços da AWS aos quais cada ambiente se integra para gerenciar os próprios recursos, manter seus dados seguros e fornecer monitoramento e visibilidade para seus fluxos de trabalho.

Tópicos

Componentes do Amazon MWAA

Os ambientes Amazon MWAA consistem nos quatro componentes principais a seguir:

1. Programador: analisa e monitora todos os seus DAGs e enfileira tarefas para execução quando as dependências de um DAG são atendidas. O Amazon MWAA implanta o programador como um cluster AWS Fargate com no mínimo dois programadores. É possível aumentar a contagem de programadores em até cinco, dependendo da workload. Para obter mais informações sobre as classes de ambiente do Amazon MWAA, consulte Classe de ambiente do Amazon MWAA.
2. Operadores: uma ou mais tarefas do Fargate que executam suas tarefas programadas. O número de operadores em seu ambiente é determinado por um intervalo entre o número mínimo e máximo que você especifica. O Amazon MWAA começa os operadores de ajuste de escala automático quando o número de tarefas em fila e em execução é maior do que o número de tarefas que seus operadores atuais podem lidar. Quando as tarefas em execução e em fila somam zero por mais de dois minutos, o Amazon MWAA reduz a escala verticalmente do número de operadores ao mínimo. Para obter mais informações sobre como o Amazon MWAA lida com operadores de ajuste de escala automático, consulte Ajuste de escala automático do Amazon MWAA.
3. Servidor web: executa a IU web do Apache Airflow. É possível configurar o servidor web com acesso à rede pública ou privada. Em ambos os casos, o acesso aos seus usuários do Apache Airflow é controlado pela política de controle de acesso que você define em AWS Identity and Access Management (IAM). Para obter mais informações sobre como configurar as políticas de acesso do IAM para seu ambiente, consulte Como acessar um ambiente Amazon MWAA.
4. Banco de dados: armazena metadados sobre o ambiente do Apache Airflow e seus fluxos de trabalho, incluindo o histórico de execução do DAG. O banco de dados é um banco de dados

Aurora PostgreSQL de locatário único gerenciado pela AWS e acessível aos contêineres Fargate do agendador e do operador por meio de um endpoint da VPC da Amazon protegido de forma privada.

Cada ambiente do Amazon MWAA também interage com um conjunto de serviços da AWS para lidar com uma variedade de tarefas, incluindo armazenar e acessar DAGs e dependências de tarefas, proteger seus dados em repouso e registrar em log e monitorar seu ambiente. O diagrama a seguir demonstra os diferentes componentes de um ambiente do Amazon MWAA.

> **Note**
>
> O serviço Amazon VPC não é uma VPC compartilhada. O Amazon MWAA cria uma VPC pertencente à AWS para cada ambiente que você cria.

- Amazon S3: o Amazon MWAA armazena todos os seus recursos de fluxo de trabalho, como DAGs, requisitos e arquivos de plug-in em um bucket do Amazon S3. Para obter mais informações sobre a criação do bucket como parte da criação do ambiente e o upload de seus recursos do Amazon MWAA, consulte Criar um bucket do Amazon S3 para o Amazon MWAA no Guia do usuário do Amazon MWAA.
- Amazon SQS: o Amazon MWAA usa o Amazon SQS para enfileirar suas tarefas de fluxo de trabalho com um executor Celery.
- Amazon ECR: o Amazon ECR é o host de todas as imagens do Apache Airflow. O Amazon MWAA só oferece suporte a imagens do Apache Airflow gerenciadas pela AWS.
- AWS KMS: o Amazon MWAA usa AWS KMS para garantir que seus dados estejam seguros em repouso. Por padrão, o Amazon MWAA usa chaves AWS KMS gerenciadas pela AWS, mas é possível configurar seu ambiente para usar sua própria chave AWS KMS gerenciada pelo cliente. Para obter mais informações sobre como usar sua própria chave AWS KMS gerenciada pelo cliente, consulte Chaves gerenciadas pelo cliente para criptografia de dados no Guia do usuário do Amazon MWAA.
- CloudWatch: o Amazon MWAA se integra ao CloudWatch e fornece logs e métricas de ambiente do Apache Airflow para o CloudWatch, permitindo que você monitore seus recursos do Amazon MWAA e solucione problemas.

Conectividade

Seu ambiente Amazon MWAA precisa acessar todos os serviços da AWS aos quais se integra. O perfil de execução do Amazon MWAA controla como o acesso é concedido ao Amazon MWAA para se conectar a outros serviços AWS em seu nome. Para conectividade de rede, é possível fornecer acesso público à Internet à sua Amazon VPC ou criar Amazon VPC endpoints. Para obter mais informações sobre a configuração de endpoints da VPC da Amazon (AWS PrivateLink) para seu ambiente, consulte Gerenciamento do acesso aos endpoints da VPC no Amazon MWAA no Guia do usuário do Amazon MWAA.

O Amazon MWAA instala requisitos no programador e no operador. Se seus requisitos forem provenientes de um repositório público do PyPI, seu ambiente precisará de conectividade com a Internet para baixar as bibliotecas necessárias. Para ambientes privados, é possível usar um repositório PyPI privado ou empacotar bibliotecas em arquivos `.whl` como plug-ins personalizados para seu ambiente.

Quando você configura o Apache Airflow no modo privado, a IU do Apache Airflow só pode ser acessada pelo seu Amazon VPC por meio de Amazon VPC endpoints.

Para obter mais informações sobre redes, consulte Redes em Guia do usuário do Amazon MWAA.

Principais considerações ao migrar para um novo ambiente do MWAA

Saiba mais sobre as principais considerações, como a autenticação e o perfil de execução do Amazon MWAA, ao planejar migrar suas workloads do Apache Airflow para o Amazon MWAA.

Tópicos

Autenticação

O Amazon MWAA usa AWS Identity and Access Management (IAM) para controlar o acesso à interface do usuário do Apache Airflow. Você deve criar e gerenciar políticas do IAM que concedam aos usuários do Apache Airflow permissão para acessar o servidor web e gerenciar. DAGs É possível gerenciar tanto a autenticação quanto a autorização dos perfis padrão do Apache Airflow usando o IAM em diferentes contas.

Você pode gerenciar e restringir ainda mais os usuários do Apache Airflow para acessar somente um subconjunto do seu fluxo de trabalho criando funções personalizadas do DAGs Airflow e mapeando-as de acordo com seus diretores do IAM. Para obter mais informações e um step-by-step tutorial, consulte Tutorial: Restringindo o acesso de um usuário do Amazon MWAA a um subconjunto de. DAGs

É possível também configurar identidades federadas para acessar o Amazon MWAA. Para obter mais informações, consulte o seguinte:

- Ambiente do Amazon MWAA com acesso público: uso do Okta como provedor de identidade com o Amazon MWAA no Blog de computação da AWS .
- Ambiente Amazon MWAA com acesso privado: acesso a um ambiente privado do Amazon MWAA usando identidades federadas.

Perfil de execução

O Amazon MWAA usa uma função de execução que concede permissões ao seu ambiente para acessar outros AWS serviços. Você pode fornecer acesso aos AWS serviços ao seu fluxo de trabalho adicionando as permissões relevantes à função. Se você escolher a opção padrão para criar uma nova função de execução ao criar o ambiente pela primeira vez, o Amazon MWAA atribuirá as permissões mínimas necessárias à função, exceto no caso de CloudWatch registros para os quais o Amazon MWAA adiciona todos os grupos de log automaticamente.

Depois que o perfil de execução é criado, o Amazon MWAA não pode gerenciar as políticas de permissão do perfil em seu nome. Para atualizar o perfil de execução, você deve editar a política para adicionar e remover permissões conforme necessário. Por exemplo, é possível integrar seu ambiente Amazon MWAA com AWS Secrets Manager como um back-end para armazenar com segurança segredos e cadeias de conexão para usar em seus fluxos de trabalho do Apache Airflow. Para isso, anexe a seguinte política de permissão ao perfil de execução do seu ambiente.

JSON

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

A integração com outros AWS serviços segue um padrão semelhante: você adiciona a política de permissão relevante à sua função de execução do Amazon MWAA, concedendo permissão ao Amazon MWAA para acessar o serviço. Para obter mais informações sobre como gerenciar o perfil de execução do Amazon MWAA e ver exemplos adicionais, visite Perfil de execução do Amazon MWAA no Guia do usuário do Amazon MWAA.

Migração para um novo ambiente do Amazon MWAA

Examine as etapas a seguir para migrar a workload existente do Apache Airflow para um novo ambiente do Amazon MWAA. É possível usar essas etapas para migrar de uma versão mais antiga do Amazon MWAA para uma nova ou migrar sua implantação autogerenciada do Apache Airflow para o Amazon MWAA. Este tutorial pressupõe que você esteja migrando de um Apache Airflow v1.10.12 existente para um novo Amazon MWAA executando o Apache Airflow v2.5.1, mas é possível usar os mesmos procedimentos para migrar de ou para diferentes versões do Apache Airflow.

Tópicos

Pré-requisitos

Para poder concluir as etapas e migrar seu ambiente, você precisará de:

- Uma implantação do Apache Airflow. Isso pode ser um ambiente Amazon MWAA autogerenciado ou existente.
- O Docker instaladopara seu sistema operacional local.
- AWS Command Line Interface versão 2 instalada.

Etapa um: criar um novo ambiente Amazon MWAA executando a versão mais recente compatível do Apache Airflow

Você pode criar um ambiente usando as etapas detalhadas em Introdução ao Amazon MWAA no Guia do usuário do Amazon MWAA ou usando um modelo. CloudFormation Se você estiver

migrando de um ambiente Amazon MWAA existente e tiver usado um CloudFormation modelo para criar seu ambiente antigo, poderá alterar a `AirflowVersion` propriedade para especificar a nova versão.

```
MwaaEnvironment:
  Type: AWS::MWAA::Environment
  DependsOn: MwaaExecutionPolicy
  Properties:
    Name: !Sub "${AWS::StackName}-MwaaEnvironment"
    SourceBucketArn: !GetAtt EnvironmentBucket.Arn
    ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
    AirflowVersion: 2.5.1
    DagS3Path: dags
  NetworkConfiguration:
    SecurityGroupIds:
      - !GetAtt SecurityGroup.GroupId
    SubnetIds:
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet2
  WebserverAccessMode: PUBLIC_ONLY
  MaxWorkers: !Ref MaxWorkerNodes
  LoggingConfiguration:
    DagProcessingLogs:
      LogLevel: !Ref DagProcessingLogs
      Enabled: true
    SchedulerLogs:
      LogLevel: !Ref SchedulerLogsLevel
      Enabled: true
    TaskLogs:
      LogLevel: !Ref TaskLogsLevel
      Enabled: true
    WorkerLogs:
      LogLevel: !Ref WorkerLogsLevel
      Enabled: true
    WebserverLogs:
      LogLevel: !Ref WebserverLogsLevel
      Enabled: true
```

Como alternativa, se estiver migrando de um ambiente Amazon MWAA existente, é possível copiar o seguinte script Python que usa o AWS SDK for Python (Boto3) para clonar seu ambiente. Você também pode fazer download do script .

Script Python

```
# This Python file uses the following encoding: utf-8
'''
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: MIT-0

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify,
merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
'''
from __future__ import print_function
import argparse
import json
import socket
import time
import re
import sys
from datetime import timedelta
from datetime import datetime
import boto3
from botocore.exceptions import ClientError, ProfileNotFound
from boto3.session import Session
ENV_NAME = ""
REGION = ""

def verify_boto3(boto3_current_version):
    '''
    check if boto3 version is valid, must be 1.17.80 and up
    return true if all dependenceis are valid, false otherwise
    '''
    valid_starting_version = '1.17.80'
    if boto3_current_version == valid_starting_version:
        return True
```

```
    ver1 = boto3_current_version.split('.')
    ver2 = valid_starting_version.split('.')
    for i in range(max(len(ver1), len(ver2))):
        num1 = int(ver1[i]) if i < len(ver1) else 0
        num2 = int(ver2[i]) if i < len(ver2) else 0
        if num1 > num2:
            return True
        elif num1 < num2:
            return False
    return False


def get_account_id(env_info):
    '''
    Given the environment metadata, fetch the account id from the
    environment ARN
    '''
    return env_info['Arn'].split(":")[4]


def validate_envname(env_name):
    '''
    verify environment name doesn't have path to files or unexpected input
    '''
    if re.match(r"^[a-zA-Z][0-9a-zA-Z-_]*$", env_name):
        return env_name
    raise argparse.ArgumentTypeError("%s is an invalid environment name value" %
 env_name)


def validation_region(input_region):
    '''
    verify environment name doesn't have path to files or unexpected input
    REGION: example is us-east-1
    '''
    session = Session()
    mwaa_regions = session.get_available_regions('mwaa')
    if input_region in mwaa_regions:
        return input_region
    raise argparse.ArgumentTypeError("%s is an invalid REGION value" % input_region)


def validation_profile(profile_name):
    '''
```

```
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r"^[a-zA-Z0-9]*$", profile_name):
        return profile_name
    raise argparse.ArgumentTypeError("%s is an invalid profile name value" %
 profile_name)

def validation_version(version_name):
    '''
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r"[1-2].\d.\d", version_name):
        return version_name
    raise argparse.ArgumentTypeError("%s is an invalid version name value" %
 version_name)

def validation_execution_role(execution_role_arn):
    '''
    verify profile name doesn't have path to files or unexpected input
    '''
    if re.match(r'(?i)\b((?:[a-z][\w-]+:(?:/{1,3}|[a-z0-9%])|www\d{0,3}[.]|[a-z0-9.
\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|
(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:\'".,<>?«»“”‘’]))', execution_role_arn):
        return execution_role_arn
    raise argparse.ArgumentTypeError("%s is an invalid execution role ARN" %
 execution_role_arn)

def create_new_env(env):
    '''
    method to duplicate env
    '''
    mwaa = boto3.client('mwaa', region_name=REGION)

    print('Source Environment')
    print(env)
    if (env['AirflowVersion']=="1.10.12") and (VERSION=="2.2.2"):
        if env['AirflowConfigurationOptions']
['secrets.backend']=='airflow.contrib.secrets.aws_secrets_manager.SecretsManagerBackend':
            print('swapping',env['AirflowConfigurationOptions']['secrets.backend'])
            env['AirflowConfigurationOptions']
['secrets.backend']='airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
    env['LoggingConfiguration']['DagProcessingLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['SchedulerLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['TaskLogs'].pop('CloudWatchLogGroupArn')
```

```
    env['LoggingConfiguration']['WebserverLogs'].pop('CloudWatchLogGroupArn')
    env['LoggingConfiguration']['WorkerLogs'].pop('CloudWatchLogGroupArn')
    env['AirflowVersion']=VERSION
    env['ExecutionRoleArn']=EXECUTION_ROLE_ARN
    env['Name']=ENV_NAME_NEW
    env.pop('Arn')
    env.pop('CreatedAt')
    env.pop('LastUpdate')
    env.pop('ServiceRoleArn')
    env.pop('Status')
    env.pop('WebserverUrl')
    if not env['Tags']:
        env.pop('Tags')
    print('Destination Environment')
    print(env)

    return mwaa.create_environment(**env)

def get_mwaa_env(input_env_name):

    # https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/
mwaa.html#MWAA.Client.get_environment
    mwaa = boto3.client('mwaa', region_name=REGION)
    environment = mwaa.get_environment(
        Name=input_env_name
    )['Environment']

    return environment

def print_err_msg(c_err):
    '''short method to handle printing an error message if there is one'''
    print('Error Message: {}'.format(c_err.response['Error']['Message']))
    print('Request ID: {}'.format(c_err.response['ResponseMetadata']['RequestId']))
    print('Http code: {}'.format(c_err.response['ResponseMetadata']['HTTPStatusCode']))

#
# Main
#
# Usage:
# python3 clone_environment.py --envname MySourceEnv --envnamenew MyDestEnv --region
 us-west-2 --execution_role AmazonMWAA-MyDestEnv-ExecutionRole --version 2.2.2
#
# based on https://github.com/awslabs/aws-support-tools/blob/master/MWAA/verify_env/
verify_env.py
```

```
#

if __name__ == '__main__':
    if sys.version_info[0] < 3:
        print("python2 detected, please use python3. Will try to run anyway")
    if not verify_boto3(boto3.__version__):
        print("boto3 version ", boto3.__version__, "is not valid for this script. Need
 1.17.80 or higher")
        print("please run pip install boto3 --upgrade --user")
        sys.exit(1)
    parser = argparse.ArgumentParser()
    parser.add_argument('--envname', type=validate_envname, required=True, help="name
 of the source MWAA environment")
    parser.add_argument('--region', type=validation_region,
 default=boto3.session.Session().region_name,
                        required=False, help="region, Ex: us-east-1")
    parser.add_argument('--profile', type=validation_profile, default=None,
                        required=False, help="AWS CLI profile, Ex: dev")
    parser.add_argument('--version', type=validation_version, default="2.2.2",
                        required=False, help="Airflow destination version, Ex: 2.2.2")
    parser.add_argument('--execution_role', type=validation_execution_role,
 default=None,
                        required=True, help="New environment execution role ARN, Ex:
 arn:aws:iam::112233445566:role/service-role/AmazonMWAA-MyEnvironment-ExecutionRole")
    parser.add_argument('--envnamenew', type=validate_envname, required=True,
 help="name of the destination MWAA environment")

    args, _ = parser.parse_known_args()
    ENV_NAME = args.envname
    REGION = args.region
    PROFILE = args.profile
    VERSION = args.version
    EXECUTION_ROLE_ARN = args.execution_role
    ENV_NAME_NEW = args.envnamenew

    try:
        print("PROFILE",PROFILE)
        if PROFILE:
            boto3.setup_default_session(profile_name=PROFILE)
        env = get_mwaa_env(ENV_NAME)
        response = create_new_env(env)
        print(response)
    except ClientError as client_error:
        if client_error.response['Error']['Code'] == 'LimitExceededException':
```

```
            print_err_msg(client_error)
            print('please retry the script')
        elif client_error.response['Error']['Code'] in ['AccessDeniedException',
'NotAuthorized']:
            print_err_msg(client_error)
            print('please verify permissions used have permissions documented in
readme')
        elif client_error.response['Error']['Code'] == 'InternalFailure':
            print_err_msg(client_error)
            print('please retry the script')
        else:
            print_err_msg(client_error)
    except ProfileNotFound as profile_not_found:
        print('profile', PROFILE, 'does not exist; check the profile name')
    except IndexError as error:
        print("Error:", error)
```

Etapa 2: migrar seus recursos de fluxo de trabalho

O Apache Airflow v2 é um grande lançamento de versão. Se você estiver migrando do Apache Airflow v1, deverá preparar seus recursos de fluxo de trabalho e verificar as alterações feitas em DAGs seus requisitos e plug-ins. Para fazer isso, recomendamos configurar uma versão bridge do Apache Airflow em seu sistema operacional local usando o Docker e o executor local Amazon MWAA. O executor local do Amazon MWAA fornece um utilitário de interface de linha de comando (CLI) que replica um ambiente do Amazon MWAA localmente.

Sempre que você estiver alterando as versões do Apache Airflow, certifique-se de referenciar o URL correto do `--constraint` no seu `requirements.txt`.

Para migrar seus recursos de fluxo de trabalho

1. Crie uma bifurcação do aws-mwaa-local-runnerrepositório e clone uma cópia do executor local do Amazon MWAA.
2. Confira a `v1.10.15` ramificação do aws-mwaa-local-runner repositório. O Apache Airflow lançou a versão 1.10.15 como uma versão intermediária para ajudar na migração para o Apache Airflow v2 e, embora o Amazon MWAA não ofereça suporte à versão 1.10.15, é possível usar o executor local do Amazon MWAA para testar seus recursos.

3. Use a ferramenta CLI do executor local Amazon MWAA para criar a imagem do Docker e executar o Apache Airflow localmente. Para obter mais informações, consulte o executor local README no GitHub repositório.
4. Ao usar o Apache Airflow executado localmente, siga as etapas descritas em Atualização da versão 1.10 para a 2 no site de documentação do Apache Airflow.
    a. Para atualizar seu `requirements.txt`, siga as melhores práticas que recomendamos em Gerenciar dependências do Python, no Guia do usuário do Amazon MWAA.
    b. Se você tiver empacotado seus operadores e sensores personalizados com seus plug-ins para seu ambiente Apache Airflow v1.10.12 existente, mova-os para sua pasta DAG. Para obter mais informações sobre as melhores práticas de gerenciamento de módulos para o Apache Airflow v2+, consulte Gerenciamento de módulos no site de documentação do Apache Airflow.
5. Depois de fazer as alterações necessárias nos recursos do fluxo de trabalho, confira a `v2.5.1` ramificação do aws-mwaa-local-runner repositório e teste localmente o fluxo de trabalho atualizado DAGs, os requisitos e os plug-ins personalizados. Se você estiver migrando para uma versão diferente do Apache Airflow, é possível usar a ramificação local apropriada do executor para sua versão, em vez disso.
6. Depois de testar com sucesso seus recursos de fluxo de trabalho, copie seus DAGs plug-ins para o bucket do Amazon S3 que você configurou com seu novo ambiente Amazon MWAA. `requirements.txt`

Etapa três: exportar os metadados do seu ambiente existente

As tabelas de metadados do Apache Airflow, como `dag`, `dag_tag` e `dag_code` são preenchidas automaticamente quando você copia os arquivos DAG atualizados para o bucket Amazon S3 do seu ambiente e o agendador os analisa. As tabelas relacionadas à permissão também são preenchidas automaticamente com base na permissão da perfil de execução do IAM. Você não precisa migrar.

É possível migrar dados relacionados ao histórico do DAG, `variable`, `slot_pool`, `sla_misse`, se necessário, tabelas `xcom`, `job` e `log`. O registro da instância da tarefa é armazenado nos CloudWatch Registros, no grupo de `airflow-{environment_name}` registros. Se você quiser ver os logs da instância de tarefas de execuções mais antigas, esses logs devem ser copiados para o novo grupo de logs do ambiente. Recomendamos que você mova apenas alguns dias de logs para reduzir os custos associados.

Se você estiver migrando de um ambiente Amazon MWAA existente, não há acesso direto ao banco de dados de metadados. Você deve executar um DAG para exportar os metadados do seu ambiente do Amazon MWAA existente para um bucket do Amazon S3 de sua escolha. As etapas a seguir também podem ser usadas para exportar metadados do Apache Airflow se você estiver migrando de um ambiente autogerenciado.

Depois que os dados são exportados, é possível executar um DAG em seu novo ambiente para importar os dados. Durante o processo de exportação e importação, todos os outros DAGs são pausados.

Para exportar os metadados do seu ambiente existente

1. Crie um bucket do Amazon S3 usando o AWS CLI para armazenar os dados exportados. Substitua `UUID` e `region` por suas próprias informações.

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

> **Note**
>
> Se você estiver migrando dados confidenciais, como conexões armazenadas em variáveis, recomendamos que você habilite a criptografia padrão para o bucket do Amazon S3.

2.
> **Note**
>
> Não se aplica à migração de um ambiente autogerenciado.

Modifique o perfil de execução do ambiente existente e adicione a política a seguir para conceder acesso de gravação ao bucket que você criou na etapa um.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

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

3. Clone o amazon-mwaa-examplesrepositório e navegue até o `metadata-migration` subdiretório do seu cenário de migração.

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

4. Em `export_data.py`, substitua o valor da string pelo S3_BUCKET bucket do Amazon S3 que você criou para armazenar metadados exportados.

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

5. Localize o arquivo `requirements.txt` no diretório `metadata-migration`. Se você já tiver um arquivo de requisitos para seu ambiente existente, adicione os requisitos adicionais especificados em `requirements.txt` para seu arquivo. Se você não tiver um arquivo de requisitos existente, basta usar o fornecido no diretório `metadata-migration`.
6. Copie `export_data.py` para o diretório do DAG do bucket do Amazon S3 associado ao seu ambiente existente. Se estiver migrando de um ambiente autogerenciado, copie `export_data.py` para sua pasta `/dags`.
7. Copie seu `requirements.txt` atualizado para o bucket do Amazon S3 associado ao seu ambiente existente e, em seguida, edite o ambiente para especificar a nova versão `requirements.txt`.
8. Depois que o ambiente for atualizado, acesse a IU do Apache Airflow, retome o DAG `db_export` e acione a execução do fluxo de trabalho.
9. Verifique se os metadados são exportados para `data/migration/existing-version_to_new-version/export/` o bucket do Amazon S3 `mwaa-migration-{UUID}`, com cada tabela em seu próprio arquivo dedicado.

Etapa quatro: importar os metadados para seu novo ambiente

Para importar os metadados para seu novo ambiente

1. Em `import_data.py`, substitua os valores de string pelos seguintes com suas informações.

   - Para migração de um ambiente Amazon MWAA existente:

   ```
   S3_BUCKET = 'mwaa-migration-{UUID}'
          OLD_ENV_NAME='{old_environment_name}'
          NEW_ENV_NAME='{new_environment_name}'
          TI_LOG_MAX_DAYS = {number_of_days}
   ```

   MAX_DAYS controla quantos dias de arquivos de log o fluxo de trabalho copia para o novo ambiente.

   - Para migrar de um ambiente autogerenciado:

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

2. (Opcional) `import_data.py` copia somente logs de tarefas com falha. Se você quiser copiar todos os logs de tarefas, modifique a função `getDagTasks` e remova `ti.state = 'failed'` conforme mostrado no trecho de código a seguir.

   ```
   def getDagTasks():
        session = settings.Session()
        dagTasks = session.execute(f"select distinct ti.dag_id, ti.task_id,
    date(r.execution_date) as ed \
        from task_instance ti, dag_run r where r.execution_date > current_date -
    {TI_LOG_MAX_DAYS} and \
        ti.dag_id=r.dag_id and ti.run_id = r.run_id order by ti.dag_id,
    date(r.execution_date);").fetchall()
        return dagTasks
   ```

3. Modifique o perfil de execução do seu novo ambiente e adicione a política a seguir. A política de permissão permite que o Amazon MWAA leia do bucket Amazon S3 para o qual você exportou os metadados do Apache Airflow e copie os logs de instâncias de tarefas dos grupo de logs existentes. Substitua todos os marcadores por suas próprias informações.

> **Note**
>
> Se você estiver migrando de um ambiente autogerenciado, deverá remover as permissões relacionadas aos CloudWatch registros da política.

JSON

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

4. Copie `import_data.py` para o diretório DAG do bucket Amazon S3 associado ao seu novo ambiente e, em seguida, acesse a IU do Apache Airflow para interromper o DAG `db_import` e acionar o fluxo de trabalho. O novo DAG aparecerá na IU do Apache Airflow em alguns minutos.

5. Depois que a execução do DAG for concluída, verifique se o histórico de execução do DAG foi copiado acessando cada DAG individual.

Próximas etapas

- Para obter mais informações sobre as classes e capacidades de ambiente do Amazon MWAA disponíveis, consulte Classe de ambiente do Amazon MWAA no Guia do usuário do Amazon MWAA.
- Para obter mais informações sobre como o Amazon MWAA lida com operadores de ajuste de escala automático, consulte Ajuste de escala automático do Amazon MWAA no Guia do usuário do Amazon MWAA.
- Para obter mais informações sobre a API REST do Amazon MWAA, consulte a API REST do Amazon MWAA.

Migre cargas de trabalho do AWS Data Pipeline Amazon MWAA

AWS lançou o AWS Data Pipeline serviço em 2012. Naquela época, os clientes queriam um serviço que lhes permitisse usar uma variedade de opções de computação para mover dados entre diferentes fontes de dados. À medida que as necessidades de transferência de dados mudaram com o tempo, as soluções para essas necessidades também mudaram. Agora você tem a opção de escolher a solução que melhor atenda às suas necessidades comerciais. Você pode migrar suas cargas de trabalho para qualquer um dos seguintes serviços: AWS

- Use o Amazon Managed Workflows for Apache Airflow (Amazon MWAA) para gerenciar a orquestração do fluxo de trabalho do Apache Airflow.
- Use o Step Functions para orquestrar fluxos de trabalho entre vários Serviços da AWS.
- Use AWS Glue para executar e orquestrar aplicativos Apache Spark.

A opção escolhida depende de sua workload atual em AWS Data Pipeline. Este tópico explica como migrar do AWS Data Pipeline Amazon MWAA.

Tópicos

Escolher o Amazon MWAA

O Amazon Managed Workflows for Apache Airflow (Amazon MWAA) é um serviço gerenciado de orquestração para o Apache Airflow que permite configurar e operar pipelines de dados na nuvem em grande escala. end-to-end O Apache Airflow é uma ferramenta de código aberto usada para criar, agendar e monitorar programaticamente sequências de processos e tarefas chamadas de fluxos de trabalho. Com o Amazon MWAA, é possível usar o Apache Airflow e a linguagem de programação Python para criar fluxos de trabalho sem precisar gerenciar a infraestrutura subjacente para fins de escalabilidade, disponibilidade e segurança. O Amazon MWAA escala automaticamente sua

capacidade de fluxo de trabalho para atender às suas necessidades e é integrado aos serviços de AWS segurança para ajudar a fornecer acesso rápido e seguro aos seus dados.

A seguir, destacamos alguns dos benefícios de AWS Data Pipeline migrar do Amazon MWAA:

- Escalabilidade e desempenho aprimorados: o Amazon MWAA fornece uma estrutura flexível e escalável para definir e executar fluxos de trabalho. Isso permite que os usuários lidem com fluxos de trabalho grandes e complexos com facilidade e aproveitem atributos como agendamento dinâmico de tarefas, fluxos de trabalho orientados por dados e paralelismo.
- Monitoramento e registro aprimorados — O Amazon MWAA se integra CloudWatch à Amazon para aprimorar o monitoramento e o registro de seus fluxos de trabalho. O Amazon MWAA envia automaticamente métricas e registros do sistema para o. CloudWatch Isso significa que é possível acompanhar o progresso e o desempenho de seus fluxos de trabalho em tempo real e identificar quaisquer problemas que surjam.
- Melhores integrações com AWS serviços e software de terceiros — O Amazon MWAA se integra a uma variedade de outros AWS serviços, como Amazon S3 e AWS Glue Amazon Redshift, além de softwares de terceiros, como DBT, Snowflake e Databricks. Isso permite que você processe e transfira dados em diferentes ambientes e serviços.
- Ferramenta de pipeline de dados de código aberto: o Amazon MWAA utiliza o mesmo produto de código aberto do Apache Airflow com o qual você está familiarizado. O Apache Airflow é uma ferramenta desenvolvida especificamente para lidar com todos os aspectos do gerenciamento do pipeline de dados, incluindo ingestão, processamento, transferência, testes de integridade, verificações de qualidade e garantia da linhagem de dados.
- Arquitetura moderna e flexível: o Amazon MWAA aproveita a conteinerização e as tecnologias nativas de nuvem e tecnologias sem servidor. Isso significa mais flexibilidade e portabilidade, além de facilitar a implantação e o gerenciamento de seus ambientes de fluxo de trabalho.

Arquitetura e mapeamento de conceitos

AWS Data Pipeline e o Amazon MWAA têm arquiteturas e componentes diferentes, que podem afetar o processo de migração e a forma como os fluxos de trabalho são definidos e executados. Esta seção apresenta uma visão geral da arquitetura e dos componentes de ambos os serviços e destaca algumas das principais diferenças.

Ambos AWS Data Pipeline e o Amazon MWAA são serviços totalmente gerenciados. Ao migrar suas workloads para o Amazon MWAA, talvez seja necessário aprender novos conceitos para

modelar seus fluxos de trabalho existentes usando o Apache Airflow. No entanto, você não precisará gerenciar a infraestrutura, corrigir os operadores e gerenciar as atualizações do sistema operacional.

A tabela a seguir associa os principais conceitos AWS Data Pipeline aos do Amazon MWAA. Use essas informações como ponto de partida para criar um plano de migração.

Conceito	AWS Data Pipeline	Amazon MWAA
Definição de pipeline	AWS Data Pipeline usa o arquivo de configuração baseado em JSON que define o fluxo de trabalho.	O Amazon MWAA usa gráficos acíclicos direcionados () baseados em Python que definem o fluxo de trabalho. DAGs
Ambiente de execução do pipeline	Os fluxos de trabalho são executados em instâncias do Amazon EC2. AWS Data Pipeline provisiona e gerencia essas instâncias em seu nome.	O Amazon MWAA usa ambientes em contêineres do Amazon ECS para executar tarefas.
Componentes do pipeline	As atividades são tarefas de processamento que são executadas como parte do fluxo de trabalho.	Os operadores (tarefas) são as unidades fundamentais de processamento de um fluxo de trabalho.
	Precondições contêm instruções condicionais que precisam ser verdadeiras para que uma atividade possa ser executada.	Os sensores (tarefas) representam declarações condicionais que podem aguardar a conclusão de um recurso ou tarefa antes de serem executados.
	Um recurso em AWS Data Pipeline se refere ao recurso AWS computacional que executa o trabalho que uma atividade de pipeline especific	Ao usar tarefas em um DAG, é possível definir uma variedade de atributos computacionais, incluindo Amazon ECS, Amazon EMR e Amazon

Conceito	AWS Data Pipeline	Amazon MWAA
	a. O Amazon EC2 e o Amazon EMR são dois atributos disponíveis.	EKS. O Amazon MWAA executa operações de Python em operadores que são executados no Amazon ECS.
Execução do pipeline	AWS Data Pipeline suporta execuções de agendamento com padrões regulares baseados em taxas e cron.	O Amazon MWAA é compatível ao agendamento com expressões cron e predefinições, bem como calendários personalizados.
	Uma instância se refere a cada execução do pipeline.	Uma execução do DAG se refere a cada execução de um fluxo de trabalho do Apache Airflow.
	Uma tentativa se refere a uma nova tentativa de uma operação com falha.	O Amazon MWAA é compatível a novas tentativas que você define no nível do DAG ou no nível da tarefa.

Exemplos de implementações

Em muitos casos, você poderá reutilizar os recursos com os quais está orquestrando atualmente AWS Data Pipeline após a migração para o Amazon MWAA. A lista a seguir contém exemplos de implementações usando o Amazon MWAA para os casos de uso mais comuns AWS Data Pipeline .

- Executando um trabalho do Amazon EMR (workshop)AWS
- Criação de um plug-in personalizado para Apache Hive e Hadoop (Guia do usuário do Amazon MWAA)
- Copiar dados do S3 para o RedshiftAWS (workshop)
- Executar um script de shell em uma instância remota do Amazon ECS (Guia do usuário do Amazon MWAA)
- Orquestração de fluxos de trabalho híbridos (locais) (publicação no blog)

Para obter exemplos e tutoriais adicionais, consulte o seguinte:

- Tutoriais do Amazon MWAA
- Exemplos de código do Amazon MWAA

Comparação de preços

AWS Data Pipeline O preço do é baseado no número de pipelines, bem como no quanto você usa cada pipeline. As atividades que você executa mais de uma vez por dia (alta frequência) custam 1 dólar por mês por atividade. As atividades que você executa uma vez por dia ou menos (alta frequência) custam 0,60 dólar por mês por atividade. Os pipelines inativos custam 1 dólar por pipeline. Para obter mais informações, consulte a página de definição de preços do AWS Data Pipeline.

O preço do Amazon MWAA é baseado no tempo de existência do seu ambiente do Apache Airflow gerenciado e em qualquer ajuste de escala automático adicional necessário para fornecer mais capacidade de operadores ou agendador. Você paga pelo uso do ambiente do Amazon MWAA por hora (cobrado com resolução de um segundo), com taxas variáveis dependendo do tamanho do ambiente. O Amazon MWAA faz o ajuste de escala automático do número de operadores com base na configuração do seu ambiente. AWS calcula o custo de operadores adicionais separadamente. Para obter mais informações sobre o custo por hora do uso de vários tamanhos de ambiente do Amazon MWAA, consulte a página de preços do Amazon MWAA.

Recursos relacionados

Para obter mais informações e conhecer as melhores práticas para usar o Amazon MWAA, consulte os seguintes recursos:

- A Referência de API do Amazon MWAA
- Painéis de monitoramento e alarmes no Amazon MWAA
- Ajuste de desempenho para o Apache Airflow no Amazon MWAA

Histórico de documentos do Amazon MWAA

A tabela a seguir descreve adições importantes feitas ao Manual de migração do Amazon MWAA, a partir de março de 2022.

Alteração	Descrição	Data
Novo tópico sobre a migração de workloads de AWS Data Pipeline para o Amazon MWAA	Adicionadas novas informaçõ es e orientações sobre a migração de workloads existentes de AWS Data Pipeline para o Amazon MWAA. Use essas informaçõ es para ajudar a criar um plano de migração. • Migre cargas de trabalho do AWS Data Pipeline Amazon MWAA	14 de abril de 2023
Lançamento do Guia de migração do Amazon MWAA	O Amazon MWAA agora oferece orientação detalhada sobre a migração para um novo ambiente do Amazon MWAA. As etapas descritas no Guia de migração do Amazon MWAA se aplicam à migração de um ambiente Amazon MWAA existente ou de uma implantação autogerenciada do Apache Airflow. • Sobre o Guia de migração do Amazon MWAA	7 de março de 2022

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.