aws

Manual do usuário

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: Manual do usuário

Table of Contents

O que é Amazon Managed Workflows for Apache Airflow?

O Amazon Managed Workflows for Apache Airflow é um serviço de orquestração gerenciado para o Apache Airflow que você pode configurar e operar pipelines de dados na nuvem em escala. 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, você pode usar o Apache Airflow e o 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 execução de fluxo de trabalho para atender às suas necessidades. O Amazon MWAA se integra aos serviços de AWS segurança para ajudar a fornecer acesso rápido e seguro aos seus dados.

Conteúdo

Atributos

- Configuração automática do Airflow: configure rapidamente o Apache Airflow escolhendo uma versão do Apache Airflow ao criar um ambiente Amazon MWAA. O Amazon MWAA configura o Apache Airflow para você usando a mesma interface de usuário e código-fonte aberto do Apache Airflow que você pode baixar na Internet.
- Ajuste de escala automático: realiza o ajuste de escala automático dos operadores do Apache Airflow definindo o número mínimo e máximo de operadores que são executados em seu ambiente. O Amazon MWAA monitora os Operadores em seu ambiente e usa seu componente de ajuste de escala automático para adicionar Operadores para atender à demanda, até atingir o número máximo de Operadores que você definiu.
- Autenticação integrada — Habilite a autenticação e autorização baseadas em funções para seu servidor Web Apache Airflow definindo as políticas de controle de acesso no AWS Identity and Access Management (IAM). Os Apache Airflow Workers assumem essas políticas para acesso seguro aos AWS serviços.

- Segurança integrada: os Operadores e Programadores do Apache Airflow são executados no Amazon VPC do Amazon MWAA. Os dados também são criptografados automaticamente usando AWS Key Management Service, portanto, seu ambiente está seguro por padrão.
- Modos de acesso público ou privado: acesse seu servidor Web do Apache Airflow usando um modo de acesso privado ou público. O modo de acesso à rede pública usa um endpoint da VPC para seu servidor Web do Apache Airflow que pode ser acessado pela Internet. O modo de acesso à rede privada usa um endpoint da VPC para seu servidor Web do Apache Airflow que pode ser acessado em sua VPC. Em ambos os casos, o acesso dos usuários do Apache Airflow é controlado pela política de controle de acesso que você define em AWS Identity and Access Management (IAM) e AWS pelo SSO.
- Atualizações e patches simplificados: o Amazon MWAA fornece novas versões do Apache Airflow periodicamente. A equipe do Amazon MWAA atualizará e corrigirá as imagens para essas versões.
- Monitoramento do fluxo de trabalho — Veja os registros do Apache Airflow e as métricas do Apache Airflow na Amazon CloudWatch para identificar atrasos nas tarefas do Apache Airflow ou erros no fluxo de trabalho sem a necessidade de ferramentas adicionais de terceiros. O Amazon MWAA envia automaticamente as métricas do ambiente e, se habilitado, os registros do Apache Airflow para. CloudWatch
- AWS integração — O Amazon MWA oferece suporte a integrações de código aberto com Amazon Athena, Amazon AWS Batch, Amazon CloudWatch DynamoDB, Amazon AWS DataSync EMR, Amazon EKS, AWS Fargate Amazon Data Firehose,,, Amazon AWS Glue AWS Lambda Redshift, Amazon SQS, Amazon SNS, Amazon e Amazon S3, bem como centenas de operadores e sensores criados pela comunidade SageMaker.
- Frotas de operadores: o Amazon MWAA oferece suporte ao uso de contêineres para escalar a frota de operadores sob demanda e reduzir as interrupções do programador usando o Amazon ECS em AWS Fargate. Há suporte para operadores que invocam tarefas em contêineres do Amazon ECS e operadores Kubernetes que criam e executam pods em um cluster Kubernetes.

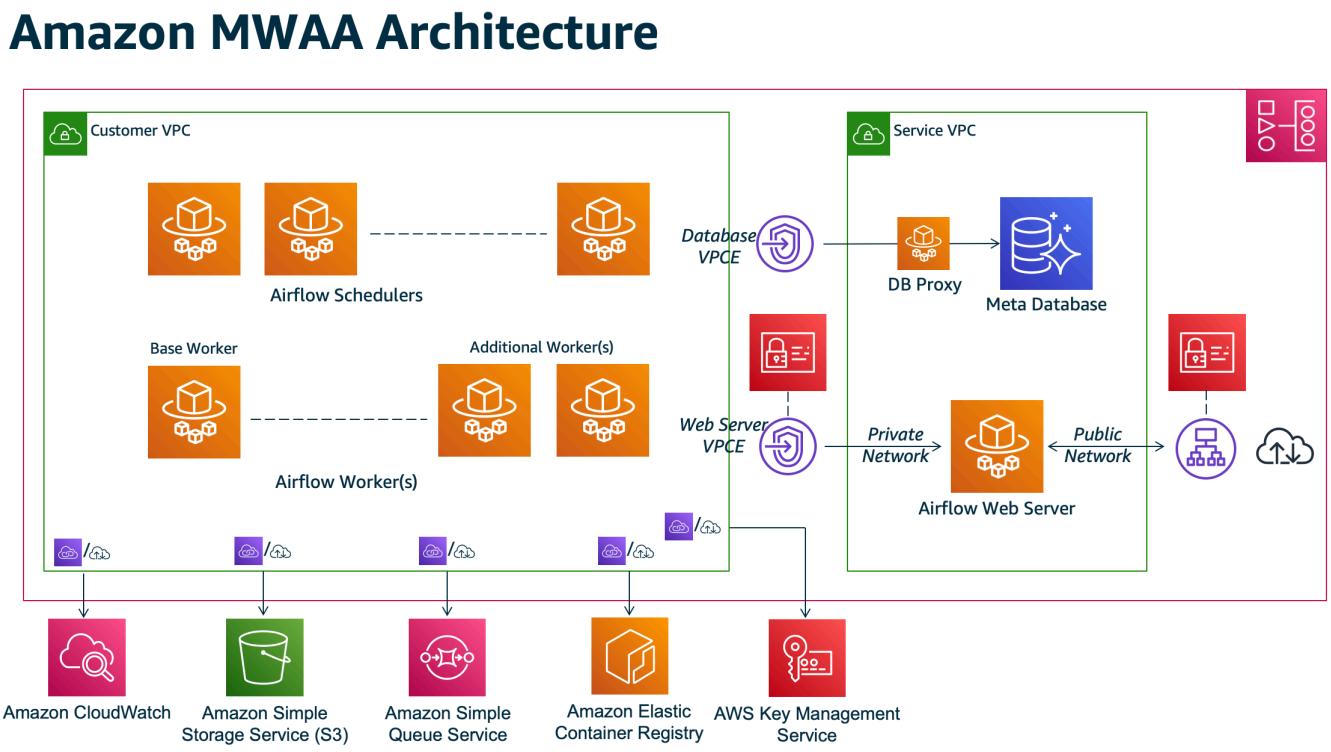
Arquitetura

Todos os componentes contidos na caixa externa (na imagem a seguir) aparecem como um único ambiente Amazon MWAA em sua conta. O Apache Airflow Scheduler e o Workers são AWS Fargate (Fargate) contêineres que se conectam às sub-redes privadas na Amazon VPC do seu ambiente. Cada ambiente tem seu próprio banco de dados Apache Airflow gerenciado por AWS esse que pode ser acessado pelos contêineres Scheduler e Workers Fargate por meio de um endpoint VPC protegido de forma privada.

Amazon CloudWatch, Amazon S3, Amazon SQS, Amazon ECR AWS KMS e Amazon são separados do Amazon MWAA e precisam estar acessíveis a partir do (s) agendador (es) do Apache Airflow e do Workers nos contêineres Fargate.

O servidor Web do Apache Airflow ainda pode ser acessado pela Internet selecionando o modo de acesso de rede pública do Apache Airflow ou dentro da sua VPC selecionando o modo de acesso de rede privada do Apache Airflow. Em ambos os casos, o acesso dos usuários do Apache Airflow é controlado pela política de controle de acesso que você define em AWS Identity and Access Management (IAM).

> **Note**
>
> Vários Programadores do Apache Airflow estão disponíveis apenas com o Apache Airflow v2 e superior. Saiba mais sobre o ciclo de vida das tarefas do Apache Airflow em Conceitos no guia de referência do Apache Airflow.



Integração

A comunidade ativa e crescente de código aberto do Apache Airflow fornece operadores (plug-ins que simplificam as conexões com os serviços) para que o Apache Airflow se integre aos serviços. AWS Isso inclui serviços como Amazon S3, Amazon Redshift, Amazon AWS Batch EMR SageMaker e Amazon, bem como serviços em outras plataformas de nuvem.

O uso do Apache Airflow com o Amazon MWAA oferece suporte total à integração com AWS serviços e ferramentas populares de terceiros, como Apache Hadoop, Presto, Hive e Spark, para realizar tarefas de processamento de dados. O Amazon MWAA está comprometido em manter a compatibilidade com a API do Amazon MWAA, e o Amazon MWAA pretende fornecer integrações confiáveis aos AWS serviços e disponibilizá-los para a comunidade, além de se envolver no desenvolvimento de recursos da comunidade.

Para obter o código de exemplo, consulte Exemplos de código para o Amazon Managed Workflows for Apache Airflow.

Versões compatíveis

O Amazon MWAA oferece suporte a várias versões do Apache Airflow. Para obter mais informações sobre as versões do Apache Airflow que oferecemos suporte e os componentes do Apache Airflow incluídos em cada versão, consulte Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow..

Próximas etapas

- Comece com um único AWS CloudFormation modelo que cria um bucket Amazon S3 para seus DAGs do Airflow e arquivos de suporte, um Amazon VPC com roteamento público e um ambiente Amazon MWAA no. Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow
- Comece de forma incremental criando um bucket Amazon S3 para seus DAGs do Airflow e arquivos de suporte, escolhendo uma das três opções de rede Amazon VPC e criando um ambiente Amazon MWAA em Comece a usar o Amazon Managed Workflows for Apache Airflow.

Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow

Este tutorial de início rápido usa um AWS CloudFormation modelo que cria a infraestrutura Amazon VPC, um bucket Amazon S3 com uma pasta e `dags` um ambiente Amazon Managed Workflows for Apache Airflow ao mesmo tempo.

Tópicos

Neste tutorial

Este tutorial mostra três AWS Command Line Interface (AWS CLI) comandos para fazer upload de um DAG para o Amazon S3, executar o DAG no Apache Airflow e visualizar os logins. CloudWatch Conclui mostrando as etapas para criar uma política do IAM para uma equipe de desenvolvimento do Apache Airflow.

> **Note**
>
> O AWS CloudFormation modelo nesta página cria um ambiente Amazon Managed Workflows for Apache Airflow para a versão mais recente do Apache Airflow disponível em. AWS CloudFormation A versão mais recente disponível é o Apache Airflow v2.8.1.

O AWS CloudFormation modelo nesta página cria o seguinte:

- Infraestrutura da VPC. O modelo usa Roteamento público pela Internet. Usa o Modo de acesso à rede pública para o servidor Web do Apache Airflow em `WebserverAccessMode: PUBLIC_ONLY`.

- Bucket do Amazon S3. O modelo cria um bucket do Amazon S3 com uma pasta `dags`. Ele está configurado para bloquear todo o acesso público, com o Versionamento de bucket habilitado, conforme definido em Criar um bucket do Amazon S3 para o Amazon MWAA.
- Ambiente do Amazon MWAA. O modelo cria um ambiente Amazon MWAA associado à `dags` pasta no bucket do Amazon S3, uma função de execução com permissão AWS para serviços usados pelo Amazon MWAA e o padrão para criptografia usando AWS uma chave própria, conforme definido em. Criar um ambiente do Amazon MWAA
- CloudWatch Registros. O modelo permite que o Apache Airflow faça login CloudWatch no nível "INFO" ou superior para o grupo de registros do agendador do Airflow, o grupo de registros do servidor web do Airflow, o grupo de registros de trabalho do Airflow, o grupo de registros de processamento do Airflow DAG e o grupo de registros de tarefas do Airflow, conforme definido em. Visualizando registros de fluxo de ar na Amazon CloudWatch

Você concluirá as seguintes etapas neste tutorial:

- Fazer upload e executar um DAG. Faça o upload do tutorial DAG do Apache Airflow para a versão mais recente do Apache Airflow compatível com o Amazon MWAA para o Amazon S3 e, em seguida, execute na IU do Apache Airflow, conforme definido em Como adicionar ou atualizar DAGs.
- Visualizar logs. Visualize o grupo de registros do servidor web Airflow em CloudWatch Logs, conforme definido emVisualizando registros de fluxo de ar na Amazon CloudWatch.
- Criar uma política de controle de acesso. Crie uma política de controle de acesso no IAM para sua equipe de desenvolvimento do Apache Airflow, conforme definido em Como acessar um ambiente do Amazon MWAA.

Pré-requisitos

O AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com AWS serviços usando comandos em seu shell de linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI — Instale a versão 2.
- AWS CLI — Configuração rápida com `aws configure`.

Etapa 1: salvar o AWS CloudFormation modelo localmente

- Copie o conteúdo do modelo a seguir e salve localmente como `mwaa-public-network.yml`. Também é possível baixar o modelo.

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:

  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: MWAAEnvironment

  VpcCIDR:
    Description: The IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.21.0/24
  MaxWorkerNodes:
    Description: The maximum number of workers that can run in the environment
```

```
    Type: Number
    Default: 2
  DagProcessingLogs:
    Description: Log level for DagProcessing
    Type: String
    Default: INFO
  SchedulerLogsLevel:
    Description: Log level for SchedulerLogs
    Type: String
    Default: INFO
  TaskLogsLevel:
    Description: Log level for TaskLogs
    Type: String
    Default: INFO
  WorkerLogsLevel:
    Description: Log level for WorkerLogs
    Type: String
    Default: INFO
  WebserverLogsLevel:
    Description: Log level for WebserverLogs
    Type: String
    Default: INFO

Resources:

 ##################################################################################################
  # CREATE VPC

 ##################################################################################################

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: MWAAEnvironment

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
```

```
      - Key: Name
        Value: MWAAEnvironment

InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs  '' ]
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs  '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

PrivateSubnet2:
  Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
```

```
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1

  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
```

```
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  EnvironmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true


##################################################################################################
 # CREATE MWAA
```

```
###############################################################################################

 MwaaEnvironment:
   Type: AWS::MWAA::Environment
   DependsOn: MwaaExecutionPolicy
   Properties:
     Name: !Sub "${AWS::StackName}-MwaaEnvironment"
     SourceBucketArn: !GetAtt EnvironmentBucket.Arn
     ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
     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
 SecurityGroup:
   Type: AWS::EC2::SecurityGroup
   Properties:
     VpcId: !Ref VPC
     GroupDescription: !Sub "Security Group for Amazon MWAA Environment
${AWS::StackName}-MwaaEnvironment"
     GroupName: !Sub "airflow-security-group-${AWS::StackName}-MwaaEnvironment"

 SecurityGroupIngress:
```

```
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  SecurityGroupEgress:
    Type: AWS::EC2::SecurityGroupEgress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      CidrIp: "0.0.0.0/0"

  MwaaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - airflow-env.amazonaws.com
                - airflow.amazonaws.com
            Action:
             - "sts:AssumeRole"
      Path: "/service-role/"

  MwaaExecutionPolicy:
    DependsOn: EnvironmentBucket
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
        - !Ref MwaaExecutionRole
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action: airflow:PublishMetrics
            Resource:
              - !Sub "arn:aws:airflow:${AWS::Region}:${AWS::AccountId}:environment/
${EnvironmentName}"
          - Effect: Deny
            Action: s3:ListAllMyBuckets
```

```
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"

          - Effect: Allow
            Action:
              - "s3:GetObject*"
              - "s3:GetBucket*"
              - "s3:List*"
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"
          - Effect: Allow
            Action:
              - logs:DescribeLogGroups
            Resource: "*"

          - Effect: Allow
            Action:
              - logs:CreateLogStream
              - logs:CreateLogGroup
              - logs:PutLogEvents
              - logs:GetLogEvents
              - logs:GetLogRecord
              - logs:GetLogGroupFields
              - logs:GetQueryResults
              - logs:DescribeLogGroups
            Resource:
              - !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:airflow-${AWS::StackName}*"
          - Effect: Allow
            Action: cloudwatch:PutMetricData
            Resource: "*"
          - Effect: Allow
            Action:
              - sqs:ChangeMessageVisibility
              - sqs:DeleteMessage
              - sqs:GetQueueAttributes
              - sqs:GetQueueUrl
              - sqs:ReceiveMessage
              - sqs:SendMessage
            Resource:
              - !Sub "arn:aws:sqs:${AWS::Region}:*:airflow-celery-*"
          - Effect: Allow
```

```
            Action:
              - kms:Decrypt
              - kms:DescribeKey
              - "kms:GenerateDataKey*"
              - kms:Encrypt
            NotResource: !Sub "arn:aws:kms:*:${AWS::AccountId}:key/*"
            Condition:
              StringLike:
                "kms:ViaService":
                  - !Sub "sqs.${AWS::Region}.amazonaws.com"
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress

  MwaaApacheAirflowUI:
```

```
    Description: MWAA Environment
    Value: !Sub  "https://${MwaaEnvironment.WebserverUrl}"
```

Etapa 2: criar a pilha usando o AWS CLI

1. No prompt de comando, navegue até o diretório em que `mwaa-public-network.yml` está armazenado. Por exemplo: .

```
cd mwaaproject
```

2. Use o comando `aws cloudformation create-stack` para criar a pilha usando o AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-environment-public-network --
template-body file://mwaa-public-network.yml --capabilities CAPABILITY_IAM
```

> **Note**
>
> São necessários mais de 30 minutos para criar a infraestrutura Amazon VPC, o bucket do Amazon S3 e o ambiente do Amazon MWAA.

Etapa 3: faça o upload de um DAG para o Amazon S3 e execute-o na IU do Apache Airflow

1. Copie o conteúdo do arquivo `tutorial.py` para a versão mais recente compatível do Apache Airflow e salve localmente como `tutorial.py`.
2. No prompt de comando, navegue até o diretório em que `tutorial.py` está armazenado. Por exemplo: .

```
cd mwaaproject
```

3. Use o comando a seguir para listar todos os seus buckets do Amazon S3.

```
aws s3 ls
```

4. Use o seguinte comando para listar os arquivos e pastas no bucket do Amazon S3 para seu ambiente.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

5. Use o script a seguir para fazer upload do arquivo `tutorial.py` para sua pasta `dags`. Substitua o valor da amostra em *YOUR_S3_BUCKET_NAME*.

```
aws s3 cp tutorial.py s3://YOUR_S3_BUCKET_NAME/dags/
```

6. Abra a página Ambientes no console do Amazon MWAA.
7. Escolha um ambiente.
8. Escolha Abrir a IU do Airflow.
9. Na IU do Apache Airflow, na lista de DAGs disponíveis, escolha o tutorial DAG.
10. Na página de detalhes do DAG, escolha o botão Pausar/Reiniciar o DAG ao lado do nome do DAG para retomar o DAG.
11. Escolha Acionar DAG.

Etapa quatro: Exibir registros em CloudWatch Registros

Você pode visualizar os registros do Apache Airflow no CloudWatch console para todos os registros do Apache Airflow que foram habilitados pela pilha. AWS CloudFormation A seção a seguir mostra como visualizar os logs do grupo de logs do servidor web Airflow.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Grupo de logs do servidor web no Airflow no painel Monitoramento.
4. Escolha o log `webserver_console_ip` em Fluxos de logs.

Próximas etapas

- Saiba mais sobre como fazer upload de DAGs, especificar dependências do Python em um `requirements.txt` e plug-ins personalizados em um `plugins.zip` em Como trabalhar com DAGs no Amazon MWAA.
- Saiba mais sobre as melhores práticas que recomendamos para ajustar o desempenho do seu ambiente em Ajuste de desempenho para o Apache Airflow no Amazon MWAA.

- Crie um painel de monitoramento para seu ambiente em Painéis de monitoramento e alarmes no Amazon MWAA.
- Execute alguns exemplos de código do DAG em Exemplos de código para o Amazon Managed Workflows for Apache Airflow.

Comece a usar o Amazon Managed Workflows for Apache Airflow

O Amazon Managed Workflows para Apache Airflow usa o Amazon VPC, o código DAG e os arquivos complementares em seu bucket de armazenamento do Amazon S3 para criar um ambiente. Este guia descreve os pré-requisitos e os recursos da AWS necessários para começar a usar o Amazon MWAA.

Tópicos

Pré-requisitos

Para criar um ambiente Amazon MWAA, talvez você queira tomar outras medidas para garantir que você tenha permissão para os recursos da AWS que precise criar.

- Conta da AWS: uma conta da AWScom permissão para usar o Amazon MWAA e os serviços e recursos da AWS usados pelo seu ambiente.

Sobre este Guia

Esta seção descreve a infraestrutura da AWS e os recursos que você criará neste guia.

- Amazon VPC: os componentes de rede Amazon VPC exigidos por um ambiente Amazon MWAA. Você pode configurar uma VPC existente que atenda a esses requisitos (avançados), conforme visto em Sobre a rede do Amazon MWAA, ou criar a VPC e os componentes de rede, conforme definido em the section called “Criar a rede VPC”.

- Bucket do Amazon S3: um bucket do Amazon S3 para armazenar seus DAGs e arquivos associados, como `plugins.zip` e `requirements.txt`. Seu bucket do Amazon S3 deve ser configurado para bloquear todo o acesso público, com o versionamento do bucket ativado, conforme definido em Criar um bucket do Amazon S3 para o Amazon MWAA.
- Ambiente Amazon MWAA: um ambiente Amazon MWAA configurado com a localização do seu bucket do Amazon S3, o caminho para seu código DAG e quaisquer plug-ins personalizados ou dependências do Python, e seu Amazon VPC e seu grupo de segurança, conforme definido em Criar um ambiente do Amazon MWAA.

Antes de começar

Para criar um ambiente Amazon MWAA, talvez você queira tomar medidas adicionais para criar e configurar outros recursos da AWS antes de criar seu ambiente.

Para criar um ambiente, você precisará fazer o seguinte:

- Chave AWS KMS: uma chave AWS KMS para criptografia de dados em seu ambiente. Você pode escolher a opção padrão no console do Amazon MWAA para criar uma chave pertencente à AWS ao criar um ambiente ou especificar uma chave gerenciada pelo cliente existente com permissões para outros serviços da AWS usados pelo seu ambiente configurados (avançado). Para saber mais, consulte Como usar chaves gerenciadas pelo cliente para criptografia.
- Perfil de execução: um perfil de execução que permite que o Amazon MWAA acesse recursos da AWS em seu ambiente. Você pode escolher a opção padrão no console do Amazon MWAA para criar um perfil de execução ao criar um ambiente. Para saber mais, consulte Perfil de execução do Amazon MWAA.
- Grupo de segurança VPC: um grupo de segurança VPC que permite que o Amazon MWAA acesse outros recursos da AWS em sua rede VPC. Você pode escolher a opção padrão no console do Amazon MWAA para criar um grupo de segurança ao criar um ambiente ou fornecer a um grupo de segurança as regras de entrada e saída apropriadas (avançadas). Para saber mais, consulte Segurança em sua VPC no Amazon MWAA.

Regiões disponíveis

O Amazon MWAA está disponível nas seguintes Regiões AWS.

- Europa (Estocolmo): eu-north-1

- Europa (Frankfurt): eu-central-1
- Europa (Irlanda): eu-west-1
- Europa (Londres): eu-west-2
- Europa (Paris): eu-west-3
- Ásia-Pacífico (Mumbai): ap-south-1
- Ásia-Pacífico (Singapura): ap-southeast-1
- Ásia-Pacífico (Sydney) – ap-southeast-2
- Ásia-Pacífico (Tóquio) – ap-northeast-1
- Ásia-Pacífico (Seul): ap-northeast-2
- Leste dos EUA (Norte da Virgínia) – us-east-1
- Leste dos EUA (Ohio): us-east-2
- Oeste dos EUA (Oregon): us-west-2
- Canadá (Central): ca-central-1
- América do Sul (São Paulo): sa-east-1

Criar um bucket do Amazon S3 para o Amazon MWAA

Este guia descreve as etapas para criar um bucket do Amazon S3 para armazenar seus Gráficos acíclicos direcionados (Directed Acyclic Graphs, DAGs) do Apache Airflow, plug-ins personalizados em `plugins.zip` e dependências do Python em um arquivo `requirements.txt`.

Sumário

Antes de começar

- O nome de um bucket do Amazon S3 não pode ser alterado depois de criar o bucket. Para obter mais informações, acesse Regras de atribuição de nomes de buckets no Manual do usuário do Amazon Simple Storage Service.
- Um bucket do Amazon S3 usado para um ambiente Amazon MWAA deve ser configurado para bloquear todo o acesso público, com o Versionamento do bucket ativado.

- Um bucket do Amazon S3 usado para um ambiente do Amazon MWAA deve estar localizado na mesma região da AWS de um ambiente do Amazon MWAA. Para ver uma lista de AWS regiões do Amazon MWAA, consulte os Endpoints e cotas do Amazon MWAA no. Referência geral da AWS

Crie o bucket

Esta seção descreve as etapas para criar o bucket do Amazon S3 para o seu ambiente.

Para criar um bucket

1. Faça login no AWS Management Console e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha Create bucket (Criar bucket).
3. Em Bucket name (Nome do bucket), insira um nome compatível com o DNS para seu bucket.

   O nome do bucket deve:

   - Seja exclusivo em todo o Amazon S3.
   - Ter entre 3 e 63 caracteres.
   - Não contém caracteres maiúsculos.
   - Começar com uma letra minúscula ou um número.

   > **Important**
   >
   > Evite incluir informações confidenciais, como números de conta, no nome do bucket. O nome do bucket é visível nos URLs que apontam para os objetos no bucket.

4. Escolha uma região AWS em Região. Essa deve ser a mesma região AWS do seu ambiente Amazon MWAA.

   - Recomendamos escolher uma região próxima de você para minimizar a latência e os custos e atender aos requisitos regulatórios.

5. Selecione Block all public access (Bloquear todo o acesso público).
6. Escolha Ativar em Versionamento de bucket.
7. Opcional - Etiquetas. Adicione pares de etiquetas de chave-valor para identificar seu bucket do Amazon S3 em Etiquetas. Por exemplo, `Bucket : Staging`.

8. Opcional - Criptografia do lado do servidor. Como opção, você pode habilitar uma das seguintes opções de criptografia em seu bucket do Amazon S3.

    a. Escolha Chave do Amazon S3 (SSE-S3 em Criptografia do lado do servidor para habilitar a criptografia do lado do servidor para o bucket.

    b. Escolha a AWS Key Management Service chave (SSE-KMS) para usar uma chave AWS KMS para criptografia em seu bucket Amazon S3:

        i. Chave gerenciada pela AWS (aws/s3): se você escolher essa opção, poderá usar uma chave pertencente à AWS gerenciada pelo Amazon MWAA ou especificar uma chave gerenciada pelo cliente para criptografar seu ambiente Amazon MWAA.

        ii. Escolha entre suas chaves AWS KMS ou Enter AWS KMS key ARN (Inserir o ARN da chave): se você optar por especificar uma chave gerenciada pelo cliente nesta etapa, deverá especificar um ID ou ARN de chave AWS KMS. Chaves aliases e multirregionais AWS KMS não são compatíveis com o Amazon MWAA. A chave AWS KMS que você especificar também deve ser usada para criptografia em seu ambiente Amazon MWAA.

9. Opcional - Configurações avançadas. Se você quiser ativar o bloqueio de objetos do Amazon S3:

    a. Selecione Configurações avançadas, Habilitar.

    > **Important**
    >
    > A ativação do bloqueio de objetos permitirá permanentemente que os objetos desse bucket sejam bloqueados. Para saber mais, consulte Bloqueio de objetos usando o bloqueio de objetos do Amazon S3 em Guia do usuário do Amazon Simple Storage Service.

    b. Escolha a confirmação.

10. Escolha Create bucket (Criar bucket).

Próximas etapas

- Saiba como criar a rede da Amazon VPC necessária para um ambiente em Criar a rede VPC.
- Saiba como gerenciar as permissões de acesso em Como faço para definir as permissões do bucket da ACL?

- Saiba como excluir um bucket armazenado em Como eu faço para excluir um bucket do S3?.

Criar a rede VPC

O Amazon Managed Workflows for Apache Airflow requer uma Amazon VPC e componentes da rede específicos para oferecer suporte a um ambiente. Este guia descreve as diferentes opções da criação da rede da Amazon VPC para o ambiente do Amazon Managed Workflows for Apache Airflow.

> **Note**
>
> O Apache Airflow funciona melhor em um ambiente de rede de baixa latência. Se você estiver usando uma Amazon VPC existente que roteia o tráfego para outra região ou para um ambiente on-premises, recomendamos adicionar endpoints AWS PrivateLink para Amazon SQS, CloudWatch, Amazon S3, AWS KMS e Amazon ECR. Para obter mais informações sobre a configuração AWS PrivateLink do Amazon MWAA, consulte Criação de uma rede Amazon VPC sem acesso à Internet.

Sumário

Pré-requisitos

A AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com os serviços da AWS usando comandos no shell da linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI: instalar a versão 2.

- AWS CLI: configuração rápida com `aws configure`.

Antes de começar

- A rede VPC que você especifica para seu ambiente não pode ser alterada após a criação do ambiente.
- Você pode usar roteamento privado ou público para seu servidor Web da Amazon VPC e do Apache Airflow. Para visualizar uma lista de opções, consulte the section called “Exemplos de casos de uso para um modo de acesso Amazon VPC e Apache Airflow”.

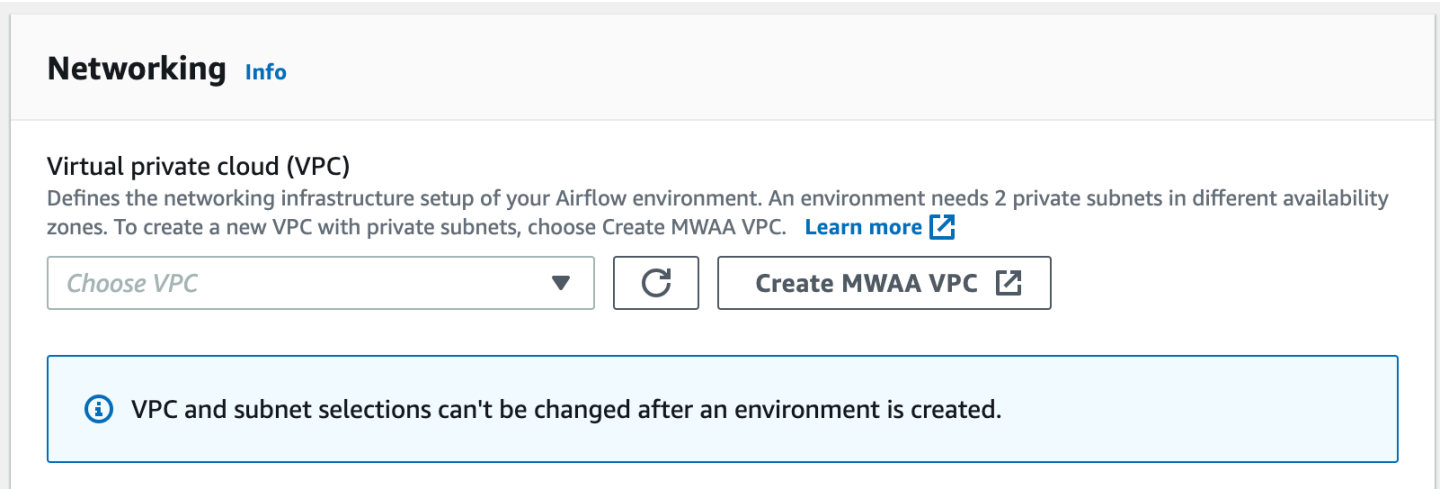
Opções para criar a rede Amazon VPC

A seção a seguir descreve as opções disponíveis para criar a rede Amazon VPC para um ambiente.

Opção um: criar a rede VPC no console Amazon MWAA

A seção a seguir mostra como criar uma rede da Amazon VPC no console do Amazon MWAA. Esta opção usa Roteamento público pela Internet. Pode ser usado para um servidor Web do Apache Airflow com os modos de acesso à rede privada ou à rede pública.

A imagem a seguir mostra onde você pode encontrar o botão Criar MWAA VPC no console Amazon MWAA.



Opção dois: criar uma rede Amazon VPC com acesso à Internet

O modelo AWS CloudFormation a seguir cria uma rede Amazon VPC com acesso à Internet em sua Região padrão AWS. Esta opção usa Roteamento público pela Internet. Este modelo pode ser usado para um servidor Web do Apache Airflow com os modos de acesso à rede privada ou à rede pública.

1. Copie o conteúdo do modelo a seguir e salve localmente como `cfn-vpc-public-private.yaml`. Também é possível baixar o modelo.

```
Description:  This template deploys a VPC, with a pair of public and private
 subnets spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: mwaa-

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the first Availability Zone
    Type: String
    Default: 10.192.20.0/24
```

```
  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
```

```
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
```

```
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1

  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC
```

```
  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress
```

2. No prompt de comando, navegue até o diretório em que `cfn-vpc-public-private.yaml` está armazenado. Por exemplo:

```
cd mwaaproject
```

3. Use o comando `aws cloudformation create-stack` para criar a pilha usando o AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-environment --template-body
 file://cfn-vpc-public-private.yaml
```

> **Note**
>
> A criação da infraestrutura da Amazon VPC requer cerca de 30 minutos.

Opção três: criar uma rede Amazon VPC sem acesso à Internet

O modelo AWS CloudFormation a seguir cria uma rede Amazon VPC sem acesso à Internet em sua região padrão AWS.

> **Important**
>
> Ao usar uma Amazon VPC sem acesso à Internet, você deve conceder permissão ao Amazon ECR para acessar o Amazon S3 usando um endpoint de gateway. É possível criar um endpoint do gateway da seguinte maneira:
>
> 1. Copie a seguinte política do IAM `JSON` e salve-a localmente como `s3-gw-endpoint-policy.json`. A política concede a permissão mínima necessária para que o Amazon ECR acesse os recursos do Amazon S3.
>
> ```
> {
>   "Statement": [
>     {
>       "Sid": "Access-to-specific-bucket-only",
>       "Principal": "*",
>       "Action": [
>         "s3:GetObject"
>       ],
>       "Effect": "Allow",
>       "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
>     }
>   ]
> ```

```
}
```

2. Crie o endpoint usando o comando AWS CLI a seguir. Substitua os valores de `--vpc-id` e `--route-table-ids` com as informações da sua Amazon VPC. Substitua `--service-name` pelo nome de acordo com sua região.

```
$ aws ec2 create-vpc-endpoint --vpc-id vpc-1a2b3c4d \
--service-name com.amazonaws.us-west-2.s3 \
--route-table-ids rtb-11aa22bb \
--vpc-endpoint-type Gateway \
--policy-document file://s3-gw-endpoint-policy.json
```

Para obter mais informações sobre a criação de endpoints de gateway para o Amazon ECR, consulte Create the Amazon S3 gateway endpoint (Criar endpoint de gateway para o Amazon S3) no Amazon Elastic Container Registry User Guide (Guia do usuário do Amazon Elastic Container Registry).

Esta opção usa Roteamento privado sem acesso à Internet. Este modelo pode ser usado para um servidor Web Apache Airflow somente com o modo de acesso à rede privada. Ele cria os endpoints da VPC necessários para os serviços da AWS usados por um ambiente.

1. Copie o conteúdo do modelo a seguir e salve localmente como `cfn-vpc-private.yaml`. Também é possível baixar o modelo.

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:
   VpcCIDR:
     Description: The IP range (CIDR notation) for this VPC
     Type: String
     Default: 10.192.0.0/16

   PrivateSubnet1CIDR:
     Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
     Type: String
     Default: 10.192.10.0/24

   PrivateSubnet2CIDR:
```

```
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
       - Key: Name
         Value: !Ref AWS::StackName

  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName}-route-table"

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName} Private Subnet (AZ1)"

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
```

```
          Value: !Sub "${AWS::StackName} Private Subnet (AZ2)"

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet1

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet2

  S3VpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.s3"
      VpcEndpointType: Gateway
      VpcId: !Ref VPC
      RouteTableIds:
       - !Ref RouteTable

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId: !Ref VPC
      GroupDescription: Security Group for Amazon MWAA Environments to access VPC
endpoints
      GroupName: !Sub "${AWS::StackName}-mwaa-vpc-endpoints"

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  SqsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.sqs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
```

```
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchLogsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.logs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchMonitoringVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.monitoring"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  KmsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.kms"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
```

```
      - !Ref SecurityGroup

  EcrApiVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.ecr.api"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  EcrDkrVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.ecr.dkr"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  AirflowApiVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.airflow.api"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  AirflowEnvVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
```

```
        ServiceName: !Sub "com.amazonaws.${AWS::Region}.airflow.env"
        VpcEndpointType: Interface
        VpcId: !Ref VPC
        PrivateDnsEnabled: true
        SubnetIds:
         - !Ref PrivateSubnet1
         - !Ref PrivateSubnet2
        SecurityGroupIds:
         - !Ref SecurityGroup

Outputs:
   VPC:
     Description: A reference to the created VPC
     Value: !Ref VPC

   MwaaSecurityGroupId:
     Description: Associates the Security Group to the environment to allow access
 to the VPC endpoints
     Value: !Ref SecurityGroup

   PrivateSubnets:
     Description: A list of the private subnets
     Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

   PrivateSubnet1:
     Description: A reference to the private subnet in the 1st Availability Zone
     Value: !Ref PrivateSubnet1

   PrivateSubnet2:
     Description: A reference to the private subnet in the 2nd Availability Zone
     Value: !Ref PrivateSubnet2
```

2. No prompt de comando, navegue até o diretório em que `cfn-vpc-private.yml` está armazenado. Por exemplo:

```
cd mwaaproject
```

3. Use o comando [aws cloudformation create-stack] para criar a pilha usando o AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-private-environment --template-
body file://cfn-vpc-private.yml
```

> **Note**
>
> A criação da infraestrutura da Amazon VPC requer cerca de 30 minutos.

4. Você precisará criar um mecanismo para acessar esses endpoints da VPC a partir do seu computador. Para saber mais, consulte Gerenciando o acesso a endpoints Amazon VPC específicos de serviços no Amazon MWAA.

> **Note**
>
> Você pode restringir ainda mais o acesso de saída no CIDR do seu grupo de segurança do Amazon MWAA. Por exemplo, você pode se restringir adicionando uma regra de saída de autorreferência, a prefix list (lista de prefixos) do Amazon S3 e o CIDR da sua Amazon VPC.

Próximas etapas

- Saiba como criar um ambiente do Amazon MWAA em Criar um ambiente do Amazon MWAA.
- Aprenda a criar um túnel VPN do seu computador para a Amazon VPC com roteamento privado em Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN.

Criar um ambiente do Amazon MWAA

O Amazon Managed Workflows for Apache Airflow configura o Apache Airflow em um ambiente na versão escolhida usando o mesmo Apache Airflow de código aberto e a mesma interface de usuário disponíveis no Apache. Este guia descreve as etapas para criar um ambiente do Amazon MWAA.

Sumário

Antes de começar

- A rede VPC que você especifica para seu ambiente não pode ser modificada após a criação do ambiente.
- Você precisa de um bucket do Amazon S3 configurado para bloquear todo o acesso público, com o controle de versionamento do bucket ativado.
- Você precisa de uma AWS conta com permissões para usar o Amazon MWAA e permissão no AWS Identity and Access Management (IAM) para criar funções do IAM. Se você escolher o modo de acesso à rede privada para o servidor web Apache Airflow, que limita o acesso ao Apache Airflow dentro do seu Amazon VPC, você precisará de permissão no IAM para criar endpoints do Amazon VPC.

Versões do Apache Airflow

As seguintes versões do Apache Airflow são compatíveis no Amazon Managed Workflows for Apache Airflow.

> **Note**
>
> - A partir do Apache Airflow v2.2.2, o Amazon MWAA oferece suporte à instalação de requisitos de Python, pacotes de provedores e plug-ins personalizados diretamente no servidor web Apache Airflow.
> - A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.
>
>   Para obter mais informações sobre como configurar restrições em seu arquivo de requisitos, consulte Instalando dependências do Python.

Versão do Apache Airflow	Guia do Apache Airflow	Restrições do Apache Airflow	Versão do Python
v2.8.1	Guia de referência do Apache Airflow v2.8.1	Arquivo de restrições do Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guia de referência do Apache Airflow v2.7.2	Arquivo de restrições do Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guia de referência do Apache Airflow v2.6.3	Arquivo de restrições do Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guia de referência do Apache Airflow v2.5.1	Arquivo de restrições do Apache Airflow v2.5.1	Python 3.10
v2.4.3	Guia de referência do Apache Airflow v2.4.3	Arquivo de restrições do Apache Airflow v2.4.3	Python 3.10
v2.2.2	Guia de referência do Apache Airflow v2.2.2	Arquivo de restrições do Apache Airflow v2.2.2	Python 3.7
v2.0.2	Guia de referência do Apache Airflow v2.0.2	Arquivo de restrições do Apache Airflow v2.0.2	Python 3.7

Para obter mais informações sobre como migrar suas implantações autogerenciadas do Apache Airflow ou migrar um ambiente Amazon MWAA existente, incluindo instruções para fazer backup do seu banco de dados de metadados, consulte Guia de migração do Amazon MWAA.

Criar um ambiente

A seção a seguir descreve as etapas para criar um ambiente do Amazon MWAA.

Etapa um: especificar detalhes

Para especificar detalhes do ambiente

1. Abra o console do Amazon MWAA.
2. Use o seletor de AWS região para selecionar sua região.
3. Selecione Create environment (Criar ambiente).
4. Na página Especificar detalhes, em Detalhes do ambiente:
   a. Digite um nome exclusivo para seu ambiente em Nome.
   b. Escolha a versão Apache Airflow na versão Airflow.

   > **Note**
   >
   > Se nenhum valor for especificado, a versão será padronizada para a mais recente do Airflow. A versão mais recente disponível é o Apache Airflow v2.8.1.

5. Em Código DAG no Amazon S3, selecione seu bucket do Amazon S3, especifique o seguinte:
   a. S3 Bucket. Escolha Browse S3 e selecione seu bucket do Amazon S3 ou insira o URI do Amazon S3.
   b. Pasta DAGs. Escolha Browse S3 e selecione a pasta `dags` em seu bucket do Amazon S3 ou insira o URI do Amazon S3.
   c. Arquivo de plug-ins - opcional. Escolha Browse S3 e selecione o arquivo `plugins.zip` em seu bucket do Amazon S3 ou insira o URI do Amazon S3.
   d. Arquivo de requisitos - opcional. Escolha Browse S3 e selecione o arquivo `requirements.txt` em seu bucket do Amazon S3 ou insira o URI do Amazon S3.
   e. Arquivo de script de inicialização - opcional, escolha Browse S3 e selecione o arquivo de script em seu bucket do Amazon S3 ou insira o URI do Amazon S3.
6. Escolha Próximo.

Etapa 2: definir as configurações avançadas

Para definir configurações avançadas

1. Na página Definir configurações avançadas, em Rede:

- Escolha sua Amazon VPC.

  Essa etapa preenche duas das sub-redes privadas em seu Amazon VPC.

2. Em Acesso ao servidor Web, selecione seu modo de acesso preferido do Apache Airflow:

   a. Rede privada. Isso limita o acesso da interface do usuário do Apache Airflow aos usuários dentro de seu Amazon VPC que receberam acesso à política do IAM para seu ambiente. Você precisa de permissão para criar endpoints da VPC Amazon para esta etapa.

   > **Note**
   >
   > Escolha a opção de rede privada se sua IU do Apache Airflow for acessada somente dentro de uma rede corporativa e você não precisar de acesso a repositórios públicos para a instalação dos requisitos do servidor web. Se escolher essa opção de modo de acesso, precisará criar um mecanismo para acessar seu servidor Web do Apache Airflow em seu Amazon VPC. Para ter mais informações, consulte Como acessar o endpoint da VPC para seu servidor Web Apache Airflow (acesso à rede privada).

   b. Rede pública. Isso permite que a IU do Apache Airflow seja acessada pela Internet por usuários com acesso à política do IAM do seu ambiente.

3. Em Grupo (s) de segurança, escolha o grupo de segurança usado para proteger sua Amazon VPC:

   a. Por padrão, o Amazon MWAA cria um grupo de segurança em sua Amazon VPC com regras específicas de entrada e saída em Criar novo grupo de segurança.

   b. Opcional. Desmarque a caixa de seleção em Criar novo grupo de segurança para selecionar até 5 grupos de segurança.

   > **Note**
   >
   > Um grupo de segurança existente do Amazon VPC deve ser configurado com regras específicas de entrada e saída para permitir o tráfego na rede. Para saber mais, consulte Segurança em sua VPC no Amazon MWAA.

4. Em Classe de ambiente, escolha uma classe de ambiente.

Recomendamos escolher o menor tamanho necessário para dar suporte a sua workload. É possível fazer alterações na classe de ambiente a qualquer momento.

5. Em Contagem máxima de operadores, especifique o número máximo de operadores do Apache Airflow a serem executados no ambiente.

   Para ter mais informações, consulte Exemplo de caso de uso de alto desempenho.

6. Especifique a contagem máxima de servidores web e a contagem mínima de servidores web para configurar como o Amazon MWAA escala os servidores web Apache Airflow em seu ambiente.

   Para obter mais informações sobre o escalonamento automático de servidores web, consultethe section called “Configurando o escalonamento automático do servidor web”.

7. Em Criptografia, escolha uma opção de criptografia de dados:

   a. Por padrão, o Amazon MWAA usa uma chave AWS própria para criptografar seus dados.

   b. Opcional. Escolha Personalizar configurações de criptografia (avançadas) para escolher uma AWS KMS chave diferente. Se você optar por especificar uma chave gerenciada pelo cliente nesta etapa, deverá especificar um ID de AWS KMS chave ou ARN. AWS KMS aliases e chaves multirregionais não são compatíveis com o Amazon MWAA. Se você especificou uma chave Amazon S3 para criptografia do lado do servidor em seu bucket Amazon S3, você deve especificar a mesma chave para seu ambiente Amazon MWAA.

      > **Note**
      >
      > Você deve ter permissões para a chave para selecioná-la no console do Amazon MWAA. Você também deve conceder permissões para que o Amazon MWAA use a chave anexando a política descrita em Anexar política de chave.

8. Recomendado. Em Monitoramento, escolha uma ou mais categorias de registro para a configuração de registro do Airflow para enviar os registros do Apache Airflow para os registros: CloudWatch

   a. Logs de tarefas do Airflow. Escolha o tipo de registros de tarefas do Apache Airflow a serem enviados para CloudWatch Logs no nível de registro.

   b. Logs do servidor web Airflow. Escolha o tipo de registros do servidor web Apache Airflow a serem enviados para o nível CloudWatch Logs in Log.

c. Logs do agendador de Airflow. Escolha o tipo de registros do agendador do Apache Airflow a serem enviados para CloudWatch Logs no nível de registro.

d. Logs de operadores do Airflow. Escolha o tipo de registros de trabalho do Apache Airflow a serem enviados para CloudWatch Logs no nível de registro.

e. Logs de processamento do Airflow DAG. Escolha o tipo de registros de processamento do Apache Airflow DAG a serem enviados para CloudWatch Logs no nível de registro.

9. Opcional. Para opções de configuração do Airflow, escolha Adicionar opção de configuração personalizada.

   É possível escolher na lista suspensa sugerida das opções de configuração do Apache Airflow para sua versão do Apache Airflow ou especificar opções de configuração personalizadas. Por exemplo, `core.default_task_retries` : `3`.

10. Opcional. Em Tags, escolha Adicionar nova tag para associar tags ao seu ambiente. Por exemplo, `Environment`: `Staging`.

11. Em Permissões, escolha um perfil de execução:

    a. Por padrão, o Amazon MWAA cria um perfil de execução em Create a new role (Criar um nova perfil). Você deve ter permissão para criar perfis do IAM.

    b. Opcional. Escolha Inserir ARN de perfil (ARN) para inserir o nome do recurso da Amazon (ARN) de um perfil de execução existente.

12. Escolha Next (Próximo).

Etapa 3: Revisar e criar

Para revisar um resumo do ambiente

- Revise o resumo do ambiente e escolha Criar ambiente.

> **Note**
>
> Leva cerca de vinte a trinta minutos para criar um ambiente.

Próximas etapas

- Saiba como criar um bucket do Amazon S3 em Criar um bucket do Amazon S3 para o Amazon MWAA.

Gerenciamento de acesso a um ambiente do Amazon MWAA

O Amazon Managed Workflows para Apache Airflow precisa ter permissão para usar outros AWS serviços e recursos usados por um ambiente. Você também precisa ter permissão para acessar um ambiente Amazon MWAA e sua interface de usuário do Apache Airflow in AWS Identity and Access Management (IAM). Esta seção descreve a função de execução usada para conceder acesso aos AWS recursos do seu ambiente e como adicionar permissões e as permissões de AWS conta necessárias para acessar seu ambiente Amazon MWAA e a interface de usuário do Apache Airflow.

Tópicos

Como acessar um ambiente do Amazon MWAA

Para usar o Amazon Managed Workflows for Apache Airflow, você deve usar uma conta e entidades do IAM com as permissões necessárias. Esta página descreve as políticas de acesso que você pode anexar à sua equipe de desenvolvimento do Apache Airflow e aos usuários do Apache Airflow para seu ambiente Amazon Managed Workflows for Apache Airflow.

Recomendamos usar credenciais temporárias e configurar identidades federadas com grupos e perfis para acessar seus recursos do Amazon MWAA. Como prática recomendada, evite anexar políticas diretamente aos usuários do IAM e, em vez disso, defina grupos ou funções para fornecer acesso temporário aos AWS recursos.

Um perfil do IAM é uma identidade do IAM que você pode criar em sua conta que tem permissões específicas. Uma função do IAM é semelhante à de um usuário do IAM, pois é uma AWS identidade com políticas de permissões que determinam o que a identidade pode ou não fazer AWS. No entanto, em vez de ser exclusivamente associada a uma pessoa, o propósito do perfil é ser assumido por qualquer pessoa que precisar dele. Além disso, um perfil não tem credenciais de longo

prazo padrão associadas a ele, como senha ou chaves de acesso. Em vez disso, quando você assumir um perfil, ele fornecerá credenciais de segurança temporárias para sua sessão de perfil.

Para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para o perfil. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criando um Perfil para um Provedor de Identidades Terceirizado no Guia do Usuário do IAM. Se você usa o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no AWS IAM Identity Center Manual do Usuário.

Você pode usar uma função do IAM em sua conta para conceder outras Conta da AWS permissões para acessar os recursos da sua conta. Para ver um exemplo, consulte Tutorial: Delegar acesso ao Contas da AWS uso de funções do IAM no Guia do usuário do IAM.

Seções

Como funciona

Os recursos e serviços usados em um ambiente Amazon MWAA não estão acessíveis a todas as entidades AWS Identity and Access Management (IAM). Você deve criar uma política que conceda permissão aos usuários do Apache Airflow para acessar esses recursos. Por exemplo, é preciso conceder acesso à sua equipe de desenvolvimento do Apache Airflow.

O Amazon MWAA usa essas políticas para validar se um usuário tem as permissões necessárias para realizar uma ação no AWS console ou por meio das APIs usadas por um ambiente.

Você pode usar as políticas JSON desse tópico para criar uma política para seus usuários do Apache Airflow no IAM e, em seguida, anexar a política a um usuário, grupo ou perfil no IAM.

- FullConsoleAcesso ao AmazonMWAA — Use essa política para conceder permissão para configurar um ambiente no console do Amazon MWAA.
- Acesso ao AmazonMWAA — Use essa política para conceder FullApi acesso a todas as APIs do Amazon MWAA usadas para gerenciar um ambiente.
- ReadOnlyAcesso ao AmazonMWAA — Use essa política para conceder acesso e visualizar os recursos usados por um ambiente no console do Amazon MWAA.
- WebServerAcesso ao AmazonMWAA — Use essa política para conceder acesso ao servidor web Apache Airflow.
- Acesso ao AmazonMWAA — Use essa política para conceder AirflowCli acesso para executar comandos da CLI do Apache Airflow.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar um perfil para um provedor de identidades de terceiros (federação) no Guia do usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Política completa de acesso ao console: FullConsole AmazonMWAA Access

Um usuário pode precisar acessar a política de permissões `AmazonMWAAFullConsoleAccess` se precisar configurar um ambiente no console do Amazon MWAA.

> **Note**
>
> Sua política completa de acesso ao console deve incluir permissões para executar `iam:PassRole`. Isso permite que o usuário transfira perfis vinculados a serviços e perfis de execução ao Amazon MWAA. O Amazon MWAA assume cada função para chamar outros AWS serviços em seu nome. O exemplo a seguir usa a chave de condição `iam:PassedToService` para especificar a entidade principal de serviço do Amazon MWAA (`airflow.amazonaws.com`) como o serviço para o qual um perfil pode ser transferido. Para obter mais informações sobre isso`iam:PassRole`, consulte Conceder permissões a um usuário para passar uma função para um AWS serviço no Guia do usuário do IAM.

Use a política a seguir se quiser criar e gerenciar seus ambientes do Amazon MWAA usando uma Chave pertencente à AWS para criptografia em repouso.

Usando um Chave pertencente à AWS

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
```

```
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreatePolicy"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:policy/service-role/MWAA-Execution-
Policy*"
      },
            {
         "Effect":"Allow",
         "Action":[
            "iam:AttachRolePolicy",
            "iam:CreateRole"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:role/service-role/AmazonMWAA*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:ListBucketVersions"
         ],
         "Resource":"*"
      },
```

```
    {
       "Effect":"Allow",
       "Action":[
          "s3:CreateBucket",
          "s3:PutObject",
          "s3:GetEncryptionConfiguration"
       ],
       "Resource":"arn:aws:s3:::*"
    },
    {
       "Effect":"Allow",
       "Action":[
          "ec2:DescribeSecurityGroups",
          "ec2:DescribeSubnets",
          "ec2:DescribeVpcs",
          "ec2:DescribeRouteTables"
       ],
       "Resource":"*"
    },
    {
       "Effect":"Allow",
       "Action":[
          "ec2:AuthorizeSecurityGroupIngress",
          "ec2:CreateSecurityGroup"
       ],
       "Resource":"arn:aws:ec2:*:*:security-group/airflow-security-group-*"
    },
    {
       "Effect":"Allow",
       "Action":[
          "kms:ListAliases"
       ],
       "Resource":"*"
    },
    {
       "Effect":"Allow",
       "Action":"ec2:CreateVpcEndpoint",
       "Resource":[
          "arn:aws:ec2:*:*:vpc-endpoint/*",
          "arn:aws:ec2:*:*:vpc/*",
          "arn:aws:ec2:*:*:subnet/*",
          "arn:aws:ec2:*:*:security-group/*"
       ]
    },
```

```
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateNetworkInterface"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:network-interface/*"
         ]
      }
   ]
}
```

Use a política a seguir se quiser criar e gerenciar seus ambientes do Amazon MWAA usando uma chave gerenciada pelo cliente para criptografia em repouso. Para usar uma chave gerenciada pelo cliente, o diretor do IAM deve ter permissão para acessar AWS KMS recursos usando a chave armazenada em sua conta.

Com usar uma chave gerenciada pelo cliente.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "iam:ListRoles"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreatePolicy"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:policy/service-role/MWAA-Execution-
Policy*"
      },
            {
         "Effect":"Allow",
         "Action":[
            "iam:AttachRolePolicy",
            "iam:CreateRole"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:role/service-role/AmazonMWAA*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:ListBucketVersions"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutObject",
            "s3:GetEncryptionConfiguration"
```

```
            ],
            "Resource":"arn:aws:s3:::*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup"
            ],
            "Resource":"arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "kms:ListAliases"
            ],
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "kms:DescribeKey",
                "kms:ListGrants",
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource":"arn:aws:kms:*:YOUR_ACCOUNT_ID:key/YOUR_KMS_ID"
        },
        {
            "Effect":"Allow",
```

```
            "Action":"ec2:CreateVpcEndpoint",
            "Resource":[
               "arn:aws:ec2:*:*:vpc-endpoint/*",
               "arn:aws:ec2:*:*:vpc/*",
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:security-group/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "ec2:CreateNetworkInterface"
            ],
            "Resource":[
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:network-interface/*"
            ]
         }
      ]
}
```

Política completa de acesso à API e ao console: FullApi AmazonMWAA Access

Um usuário pode precisar acessar a política de permissões de `AmazonMWAAFullApiAccess` se precisar de acesso a todas as APIs do Amazon MWAA usadas para gerenciar um ambiente. Ela não concede permissões para acesso da IU do Apache Airflow.

> **Note**
>
> Uma política de acesso completa à API deve incluir permissões para executar `iam:PassRole`. Isso permite que o usuário transfira perfis vinculados a serviços e perfis de execução ao Amazon MWAA. O Amazon MWAA assume cada função para chamar outros AWS serviços em seu nome. O exemplo a seguir usa a chave de condição `iam:PassedToService` para especificar a entidade principal de serviço do Amazon MWAA (`airflow.amazonaws.com`) como o serviço para o qual um perfil pode ser transferido. Para obter mais informações sobre isso`iam:PassRole`, consulte Conceder permissões a um usuário para passar uma função para um AWS serviço no Guia do usuário do IAM.

Use a política a seguir se quiser criar e gerenciar seus ambientes Amazon MWAA usando uma Chave pertencente à AWS para criptografia em repouso.

Usando um Chave pertencente à AWS

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
         ],
         "Resource":"*"
      },
      {
```

```
         "Effect":"Allow",
         "Action":[
            "s3:GetEncryptionConfiguration"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Effect":"Allow",
         "Action":"ec2:CreateVpcEndpoint",
         "Resource":[
            "arn:aws:ec2:*:*:vpc-endpoint/*",
            "arn:aws:ec2:*:*:vpc/*",
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:security-group/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateNetworkInterface"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:network-interface/*"
         ]
      }
   ]
}
```

Use a política a seguir se quiser criar e gerenciar seus ambientes do Amazon MWAA usando uma chave gerenciada pelo cliente para criptografia em repouso. Para usar uma chave gerenciada pelo cliente, o diretor do IAM deve ter permissão para acessar AWS KMS recursos usando a chave armazenada em sua conta.

Com usar uma chave gerenciada pelo cliente.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "kms:DescribeKey",
            "kms:ListGrants",
            "kms:CreateGrant",
            "kms:RevokeGrant",
            "kms:Decrypt",
            "kms:Encrypt",
            "kms:GenerateDataKey*",
            "kms:ReEncrypt*"
         ],
         "Resource":"arn:aws:kms:*:YOUR_ACCOUNT_ID:key/YOUR_KMS_ID"
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetEncryptionConfiguration"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Effect":"Allow",
         "Action":"ec2:CreateVpcEndpoint",
         "Resource":[
            "arn:aws:ec2:*:*:vpc-endpoint/*",
            "arn:aws:ec2:*:*:vpc/*",
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:security-group/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateNetworkInterface"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:network-interface/*"
         ]
      }
   ]
}
```

Política de acesso somente para leitura ao console: AmazonMWAA Access ReadOnly

Um usuário pode precisar acessar a política de permissões de `AmazonMWAAReadOnlyAccess` se precisar visualizar os recursos usados por um ambiente na página de detalhes do ambiente do console do Amazon MWAA. Ele não permite que um usuário crie novos ambientes, edite ambientes existentes ou permita que um usuário visualize a IU do Apache Airflow.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "airflow:ListEnvironments",
                "airflow:GetEnvironment",
                "airflow:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer

Um usuário pode precisar acessar a política de permissões de `AmazonMWAAWebServerAccess` se precisar acessar a IU do Apache Airflow. Ele não permite que o usuário visualize ambientes no console do Amazon MWAA nem use as APIs do Amazon MWAA para realizar nenhuma ação. Especifique o perfil `Admin`, `Op`, `User`, `Viewer` ou `Public` em `{airflow-role}` para personalizar o nível de acesso do usuário do token da web. Para obter mais informações, consulte Perfis padrão no Guia de referência do Apache Airflow.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
                "arn:aws:airflow:{your-region}:YOUR_ACCOUNT_ID:role/{your-environment-name}/{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> O Amazon MWAA fornece integração do IAM com os cinco perfis padrão de controle de acesso baseado em perfil (RBAC) do Apache Airflow. Para obter mais informações sobre

como trabalhar com perfis personalizados do Apache Airflow, consulte the section called "Tutorial: Restringir usuários a um subconjunto de DAGs".

Política de CLI do Apache Airflow: Acesso ao AmazonMWAA AirflowCli

Um usuário pode precisar acessar a política de permissões de `AmazonMWAAAirflowCliAccess` se precisar executar comandos de CLI do Apache Airflow (como `trigger_dag`). Ele não permite que o usuário visualize ambientes no console do Amazon MWAA nem use as APIs do Amazon MWAA para realizar nenhuma ação.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Como criar uma política JSON

Você pode criar a política JSON e anexar a política ao seu usuário, perfil ou grupo no console do IAM. As etapas a seguir descrevem como criar uma política JSON no IAM.

Para criar política de JSON

1. Abra a página de Políticas no console do IAM.
2. Escolha Criar política.
3. Escolha a guia JSON.
4. Adicione sua política JSON.
5. Escolha Revisar política.
6. Insira um valor no campo de texto para Nome e Descrição.

   Por exemplo, você pode nomear a política `AmazonMWAAReadOnlyAccess`.

7. Escolha Criar política.

Exemplo de caso de uso para anexar políticas a um grupo de desenvolvedores

Digamos que você esteja usando um grupo no IAM chamado `AirflowDevelopmentGroup` para aplicar permissões a todos os desenvolvedores da sua equipe de desenvolvimento do Apache Airflow. Esses usuários precisam acessar as políticas de permissão `AmazonMWAAFullConsoleAccess`, `AmazonMWAAAirflowCliAccess` e `AmazonMWAAWebServerAccess`. Esta seção descreve como criar um grupo no IAM, criar e anexar essas políticas, e associar o grupo a um usuário do IAM. As etapas pressupõem que você esteja usando uma chave pertencente àAWS.

Para criar a política do AmazonMWAA FullConsoleAccess

1. Baixe a política de acesso do AmazonMWAA FullConsoleAccess .
2. Abra a página de Políticas no console do IAM.
3. Escolha Criar política.
4. Escolha a guia JSON.
5. Cole a política JSON para `AmazonMWAAFullConsoleAccess`.
6. Substitua os seguintes valores:
   a. *`{your-account-id} — ID`* da sua AWS conta (como) `0123456789`
   b. *`{your-kms-id}`*: o identificador exclusivo de uma chave gerenciada pelo cliente, aplicável somente se você usar uma chave gerenciada pelo cliente para criptografia em repouso.
7. Escolha Revisar política.
8. Digite `AmazonMWAAFullConsoleAccess` em Nome.
9. Escolha Criar política.

Para criar a política do AmazonMWAA WebServerAccess

1. Baixe a política de acesso do AmazonMWAA WebServerAccess .
2. Abra a página de Políticas no console do IAM.
3. Escolha Criar política.

4. Escolha a guia JSON.
5. Cole a política JSON para `AmazonMWAAWebServerAccess`.
6. Substitua os seguintes valores:
    a. *`{your-region}`*: a região do seu ambiente do Amazon MWAA (como `us-east-1`)
    b. *`{your-account-id} – o ID`* da sua AWS conta (como) `0123456789`
    c. *`{your-environment-name}`*: o nome do seu ambiente do Amazon MWAA (como `MyAirflowEnvironment`)
    d. *`{airflow-role}`*: o Perfil padrão do Apache Airflow `Admin`
7. Escolha Revisar política.
8. Digite `AmazonMWAAWebServerAccess` em Nome.
9. Escolha Criar política.

Para criar a política do AmazonMWAA AirflowCliAccess

1. Baixe a política de acesso do AmazonMWAA AirflowCliAccess .
2. Abra a página de Políticas no console do IAM.
3. Escolha Criar política.
4. Escolha a guia JSON.
5. Cole a política JSON para `AmazonMWAAAirflowCliAccess`.
6. Escolha Revisar política.
7. Digite `AmazonMWAAAirflowCliAccess` em Nome.
8. Escolha Criar política.

Para criar o grupo

1. Abra a página Grupos no console do IAM.
2. Digite um nome de `AirflowDevelopmentGroup`.
3. Escolha Next Step.
4. Digite `AmazonMWAA` para filtrar os resultados em Filtro.
5. Selecione as três políticas que você criou.
6. Escolha Next Step.
7. Selecione Create Group.

Para associar a um usuário

1. Abra a página Usuários no console do IAM.
2. Escolha um usuário.
3. Selecione Grupos.
4. Escolha Adicionar usuário aos grupos.
5. Selecione o AirflowDevelopmentGrupo.
6. Selecione Add to Groups (Adicionar a grupos).

Próximas etapas

- Saiba como gerar um token para acessar a interface do Apache Airflow em Acessando o Apache Airflow.
- Saiba mais sobre a criação de políticas do IAM em Como criar políticas do IAM.

Perfil vinculado a serviços para o Amazon MWAA

O Amazon Managed Workflows para Apache Airflow usa funções vinculadas a AWS Identity and Access Management serviços (IAM). Um perfil vinculado a serviços é um tipo exclusivo de perfil do IAM vinculado diretamente ao Amazon MWAA. As funções vinculadas ao serviço são predefinidas pelo Amazon MWAA e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Um perfil vinculado a serviços facilita a configuração do Amazon MWAA porque você não precisa adicionar as permissões necessárias manualmente. O Amazon MWAA define as permissões desses perfis vinculados ao serviço e, a menos que definido em contrário, somente o Amazon MWAA pode assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon MWAA, pois você não pode remover inadvertidamente as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com perfis vinculados aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam

Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de perfil vinculado a serviços para o Amazon MWAA

O Amazon MWAA usa a função vinculada ao serviço chamada `AWSServiceRoleForAmazonMWAA` — A função vinculada ao serviço criada em sua conta concede ao Amazon MWAA acesso aos seguintes serviços: AWS

- Amazon CloudWatch Logs (CloudWatch Logs) — Para criar grupos de registros para registros do Apache Airflow.
- Amazon CloudWatch (CloudWatch) — Para publicar métricas relacionadas ao seu ambiente e seus componentes subjacentes em sua conta.
- Amazon Elastic Compute Cloud (Amazon EC2): para criar os seguintes recursos:
  - Um endpoint Amazon VPC em sua VPC para um cluster de banco de dados Amazon Aurora AWS PostgreSQL gerenciado a ser usado pelo Apache Airflow Scheduler and Worker.
  - Um endpoint da VPC do Amazon adicional para permitir o acesso de rede ao servidor Web se você escolher a opção de rede privada para seu servidor Web do Apache Airflow.
  - Interfaces de rede elásticas (ENIs) em sua Amazon VPC para permitir o acesso AWS à rede a recursos hospedados em sua Amazon VPC.

A seguinte política de confiança permite à entidade principal de serviço para assumir o perfil vinculado a serviços. A entidade principal de serviço do Amazon MWAA é `airflow.amazonaws.com`, conforme demonstrado pela política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "airflow.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

A política de permissões do perfil denominada `AmazonMWAAServiceRolePolicy` permite que o Amazon MWAA conclua as seguintes ações nos recursos especificados:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:airflow-*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpoint",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonMWAAManaged": false
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:ModifyVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "cloudwatch:namespace": [
                        "AWS/MWAA"
                    ]
                }
            }
        }
    ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Como criar um perfil vinculado a serviços para Amazon MWAA

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um novo ambiente do Amazon MWAA usando a AWS Management Console, a ou a AWS API AWS CLI, o Amazon MWAA cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria outro ambiente, o Amazon MWAA cria novamente um perfil vinculado a serviços para você.

Como editar um perfil vinculado a serviços do Amazon MWAA

O Amazon MWAA não permite que você edite a função vinculada ao AWSServiceRoleForAmazonMWAA serviço. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Como apagar um perfil vinculado a serviços do Amazon MWAA

Se você não precisar mais usar um recurso ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida.

Quando você exclui um ambiente do Amazon MWAA, o Amazon MWAA exclui todos os recursos associados que usa como parte do serviço. No entanto, você deve esperar até que o Amazon MWAA

conclua a exclusão de seu ambiente, antes de tentar excluir o perfil vinculado a serviços. Se você excluir o perfil vinculado a serviços antes que o Amazon MWAA exclua o ambiente, o Amazon MWAA talvez não consiga excluir todos os recursos associados ao ambiente.

Como excluir manualmente a função vinculada a serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForAmazonMWAA vinculada ao serviço. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte para os perfis vinculados a serviços do Amazon MWAA

O Amazon ECS é compatível com perfis vinculados a serviços em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon Managed Workflows for Apache Airflow.

Atualizações da política

Alteração	Descrição	Data
O Amazon MWAA atualiza a própria política de permissão de perfil vinculado a serviços	`AmazonMWAAServiceRolePolicy`: o Amazon MWAA atualiza a política de permissão de seu perfil vinculado a serviços para conceder permissão ao Amazon MWAA para publicar métricas adicionais relacionadas aos recursos subjacentes do serviço nas contas dos clientes. Essas novas métricas são publicadas sob a `AWS/MWAA`	18 de novembro de 2022
O Amazon MWAA passou a monitorar as alterações	A Amazon MWAA começou a monitorar as alterações em	18 de novembro de 2022

Alteração	Descrição	Data
	sua política de permissão de funções vinculadas a serviços AWS gerenciados.	

Perfil de execução do Amazon MWAA

Uma função de execução é uma função AWS Identity and Access Management (IAM) com uma política de permissões que concede permissão ao Amazon Managed Workflows for Apache Airflow para invocar os recursos de outros AWS serviços em seu nome. Isso pode incluir recursos como seu bucket do Amazon S3, chave AWS própria e CloudWatch registros. Os ambientes do Amazon MWAA precisam de um perfil de execução por ambiente. Esta página descreve como usar e configurar a função de execução do seu ambiente para permitir que o Amazon MWAA acesse outros AWS recursos usados pelo seu ambiente.

Sumário

Visão geral do perfil de execução

A permissão para o Amazon MWAA usar outros AWS serviços usados pelo seu ambiente é obtida da função de execução. Uma função de execução do Amazon MWAA precisa de permissão para os seguintes AWS serviços usados por um ambiente:

- Amazon CloudWatch (CloudWatch) — para enviar métricas e registros do Apache Airflow.
- Amazon Simple Storage Service (Amazon S3): para analisar o código DAG e os arquivos de suporte do seu ambiente (como `requirements.txt`).
- Amazon Simple Queue Service (Amazon SQS): para enfileirar as tarefas do Apache Airflow do seu ambiente em uma fila do Amazon SQS pertencente ao Amazon MWAA.
- AWS Key Management Service (AWS KMS) — para a criptografia de dados do seu ambiente (usando uma chave AWS própria ou sua chave gerenciada pelo cliente).

> **Note**
>
> Se você optou pelo Amazon MWAA para usar uma chave KMS AWS gerenciada para criptografar seus dados, então você deve definir permissões em uma política anexada à sua função de execução do Amazon MWAA que conceda acesso a chaves KMS arbitrárias armazenadas fora da sua conta via Amazon SQS. As duas condições a seguir são necessárias para que o perfil de execução do seu ambiente acesse chaves KMS arbitrárias:
>
> - Uma chave KMS em uma conta de terceiros precisa permitir esse acesso entre contas por meio da política de recursos deles.
> - Seu código DAG precisa acessar uma fila do Amazon SQS que começa com `airflow-celery-` na conta de terceiros e usa a mesma chave KMS para criptografia.
>
> Para mitigar os riscos associados ao acesso entre contas para recursos, recomendamos analisar o código inserido em seus DAGs para garantir que seus fluxos de trabalho não estejam acessando filas arbitrárias do Amazon SQS fora da sua conta. Além disso, você pode usar uma chave KMS gerenciada pelo cliente armazenada em sua própria conta para gerenciar a criptografia no Amazon MWAA. Isso limita o perfil de execução do seu ambiente para acessar somente a chave KMS em sua conta.
>
> Lembre-se de que depois de escolher uma opção de criptografia, você não poderá alterar sua seleção para um ambiente existente.

Um perfil de execução também precisa de permissão para as seguintes ações do IAM:

- `airflow:PublishMetrics`: para permitir que o Amazon MWAA monitore a integridade de um ambiente.

Permissões anexadas por padrão

Você pode usar as opções padrão no console do Amazon MWAA para criar um perfil de execução e uma chave pertencente àAWS, e então usar as etapas desta página para adicionar políticas de permissão ao seu perfil de execução.

- Quando você escolhe a opção Criar novo perfil no console, o Amazon MWAA atribui as permissões mínimas necessárias para um ambiente ao seu perfil de execução.
- Em alguns casos, o Amazon MWAA atribui as permissões máximas. Por exemplo, recomendamos escolher a opção no console do Amazon MWAA para criar um perfil de execução ao criar um ambiente. O Amazon MWAA adiciona automaticamente as políticas de permissões para todos os grupos de CloudWatch registros usando o padrão regex na função de execução como. `"arn:aws:logs:your-region:your-account-id:log-group:airflow-your-environment-name-*"`

Como adicionar permissão para usar outros AWS serviços

O Amazon MWAA não pode adicionar ou editar políticas de permissão em um perfil de execução existente após a criação de um ambiente. Você deve atualizar seu perfil de execução com políticas de permissão adicionais que sejam necessárias pelo seu ambiente. Por exemplo, se seu DAG exigir acesso ao AWS Glue, o Amazon MWAA não poderá detectar automaticamente essas permissões exigidas pelo seu ambiente nem adicionar as permissões à sua função de execução.

Você pode adicionar permissões a um perfil de execução de duas formas:

- Ao modificar a política JSON para seu perfil de execução em linha. Você pode usar os exemplos de documentos de política JSON nesta página para adicionar ou substituir a política JSON da seu perfil de execução no console do IAM.
- Ao criar uma política JSON para um AWS serviço e anexá-la à sua função de execução. Você pode usar as etapas desta página para associar um novo documento de política JSON para um AWS serviço à sua função de execução no console do IAM.

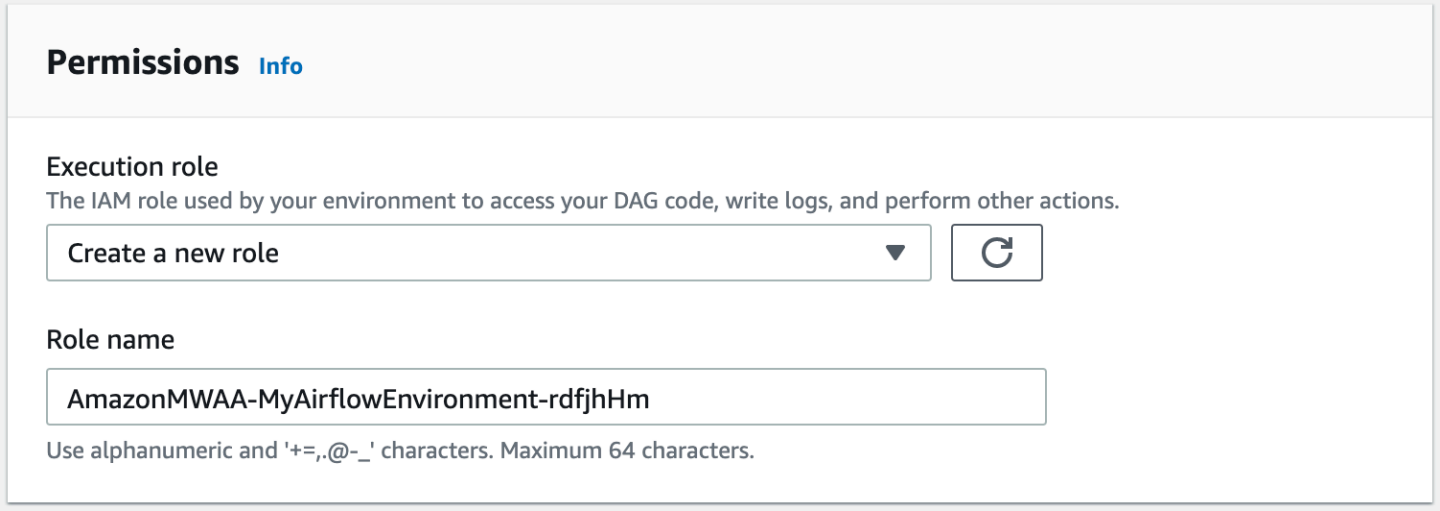
Supondo que o perfil de execução já esteja associada ao seu ambiente, o Amazon MWAA pode começar a usar as políticas de permissão adicionadas imediatamente. Isso também significa que, se você remover as permissões necessárias de um perfil de execução, seus DAGs poderão falhar.

Como associar um novo perfil de execução

Você pode alterar o perfil de execução do seu ambiente a qualquer momento. Se um novo perfil de execução ainda não estiver associado ao seu ambiente, use as etapas desta página para criar uma nova política de perfil de execução e associar o perfil ao seu ambiente.

Criar uma nova função

Por padrão, o Amazon MWAA cria uma chave pertencente àAWS para criptografia de dados e um perfil de execução em seu nome. Você pode escolher as opções padrão no console do Amazon MWAA ao criar um ambiente. A imagem a seguir mostra a opção padrão para criar um perfil de execução para um ambiente.



Visualizar e atualizar uma política de perfil de execução

Você pode visualizar o perfil de execução do seu ambiente no console do Amazon MWAA e atualizar a política JSON do perfil no console do IAM.

Para anexar uma política de perfil de execução

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha o perfil de execução no painel Permissões para abrir a página de permissões no IAM.

4. Escolha o nome do perfil de execução para abrir a política de permissões.
5. Escolha Editar política.
6. Selecione a guia JSON.
7. Atualize sua política JSON.
8. Escolha Revisar política.
9. Escolha Salvar alterações.

Anexe uma política JSON para usar outros serviços AWS

Você pode criar uma política JSON para um AWS serviço e anexá-la à sua função de execução. Por exemplo, você pode anexar a política JSON a seguir para conceder acesso de somente leitura a todos os recursos do AWS Secrets Manager.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "secretsmanager:GetResourcePolicy",
            "secretsmanager:GetSecretValue",
            "secretsmanager:DescribeSecret",
            "secretsmanager:ListSecretVersionIds"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Para anexar a política ao seu perfil de execução

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha sua função de execução no painel Permissões.
4. Escolha Anexar políticas.
5. Escolha Criar política.

6. Selecione JSON.
7. Cole a política de JSON.
8. Selecione Próximo: tags e Próximo: revisar.
9. Insira um nome descritivo (como `SecretsManagerReadPolicy`) e uma descrição para a política.
10. Escolha Criar política.

Concede acesso ao bucket do Amazon S3 com bloqueio de acesso público no nível da conta

Talvez você queira bloquear o acesso a todos os buckets em sua conta usando a operação `PutPublicAccessBlock` do Amazon S3. Quando você bloqueia o acesso a todos os buckets em sua conta, seu perfil de execução do ambiente deve incluir a ação `s3:GetAccountPublicAccessBlock` em uma política de permissão.

O exemplo a seguir demonstra a política que você deve anexar ao seu perfil de execução ao bloquear o acesso a todos os buckets do Amazon S3 em sua conta.

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": "s3:GetAccountPublicAccessBlock",
       "Resource": "*"
     }
  ]
}
```

Para obter mais informações sobre como restringir o acesso aos seus buckets do Amazon S3, consulte Bloquear o acesso público ao seu armazenamento do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Use conexões do Apache Airflow

Você também pode criar uma conexão do Apache Airflow e especificar seu perfil de execução e ARN dele no objeto de conexão do Apache Airflow. Para saber mais, consulte Como gerenciar conexões com o Apache Airflow.

Exemplos de políticas JSON para um perfil de execução

Os exemplos de políticas de permissão nesta seção mostram duas políticas que você pode usar para substituir a política de permissões usada para seu perfil de execução existente ou para criar um novo perfil de execução e usá-la em seu ambiente. Essas políticas contêm espaços reservados de ARN de recursos para grupos de log do Apache Airflow, um bucket Amazon S3 e um ambiente do Amazon MWAA.

Recomendamos copiar o exemplo de política, substituir os exemplos de ARNs ou espaços reservados e, em seguida, usar a política JSON para criar ou atualizar um perfil de execução. Por exemplo, substituindo `{your-region}` por `us-east-1`.

Exemplo de política para uma chave gerenciada pelo cliente

O exemplo a seguir mostra uma política de perfil de execução que você pode usar para uma chave gerenciada pelo cliente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{your-s3-bucket-name}",
                "arn:aws:s3:::{your-s3-bucket-name}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
```

```
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:{your-region}:{your-account-id}:log-group:airflow-{your-
environment-name}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
```

```
            "Resource": "arn:aws:sqs:{your-region}:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "Resource": "arn:aws:kms:{your-region}:{your-account-id}:key/{your-kms-cmk-
id}",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "sqs.{your-region}.amazonaws.com",
                        "s3.{your-region}.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Em seguida, você precisa permitir que o Amazon MWAA assuma esse perfil a fim de realizar ações em seu nome. Isso pode ser feito ao adicionar entidades principais de serviço `"airflow.amazonaws.com"` e `"airflow-env.amazonaws.com"` à lista de entidades confiáveis para esse perfil de execução usando o console do IAM ou colocando essas entidades principais de serviço no documento de política do perfil assumido para esse perfil de execução por meio do comando create-role de IAM com a AWS CLI. Um exemplo de documento de política do perfil assumido pode ser encontrado a seguir:

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
            "Service": ["airflow.amazonaws.com","airflow-env.amazonaws.com"]
        },
        "Action": "sts:AssumeRole"
      }
```

```
    ]
}
```

Em seguida, anexe a seguinte política JSON à sua chave gerenciada pelo cliente. Essa política usa o prefixo de chave de `kms:EncryptionContext`condição para permitir o acesso ao seu grupo de registros do Apache Airflow em Logs. CloudWatch

```
{
    "Sid": "Allow logs access",
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.{your-region}.amazonaws.com"
    },
    "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:{your-region}:{your-
account-id}:*"
        }
    }
}
```

Exemplo de política para uma AWS chave própria

O exemplo a seguir mostra uma política de perfil de execução que você pode usar para uma chave pertencente àAWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:{your-region}:{your-account-id}:environment/
{your-environment-name}"
```

```
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{your-s3-bucket-name}",
                "arn:aws:s3:::{your-s3-bucket-name}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:{your-region}:{your-account-id}:log-group:airflow-{your-
environment-name}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:{your-region}:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:{your-account-id}:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "sqs.{your-region}.amazonaws.com"
                    ]
                }
            }
        }
    ]
```

```
}
```

Próximas etapas

- Saiba mais sobre as permissões necessárias que você e seus usuários do Apache Airflow precisam para acessar seu ambiente em Como acessar um ambiente do Amazon MWAA.
- Saiba mais sobre o Como usar chaves gerenciadas pelo cliente para criptografia.
- Explore mais exemplos de políticas gerenciadas pelo cliente.

Prevenção contra o ataque do “substituto confuso” em todos os serviços

O problema “confused deputy” é um problema de segurança no qual uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Em AWS, a falsificação de identidade entre serviços pode resultar em um problema confuso de delegado. A imitação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado a usar suas permissões para atuar nos recursos de outro cliente indo contra permissão de acesso. Para evitar isso, o AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` no perfil de execução em seu ambiente para limitar as permissões que o Amazon MWAA concede a outro serviço para acessar o recurso. Use `aws:SourceArn` se quiser apenas um recurso associado a acessibilidade de serviço. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (`*`) para as partes desconhecidas do ARN. Por exemplo, `.arn:aws:airflow:*:123456789012:environment/*`

O valor de `aws:SourceArn` deve ser o ARN do seu ambiente do Amazon MWAA, para o qual você está criando um perfil de execução.

O exemplo a seguir mostra como é possível usar as chaves de contexto de condição globais `aws:SourceArn` e `aws:SourceAccount` no perfil de execução em seu ambiente para evitar o problema de “confused deputy”. Você pode usar a seguinte política de confiança ao criar um novo perfil de execução.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
            "Service": ["airflow.amazonaws.com","airflow-env.amazonaws.com"]
        },
        "Action": "sts:AssumeRole",
        "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:airflow:your-
region:123456789012:environment/your-environment-name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Modos de acesso do Apache Airflow

O console Amazon Managed Workflows for Apache Airflow contém opções integradas para configurar o roteamento privado ou público para o servidor web do Apache Airflow em seu ambiente. Este guia descreve os modos de acesso disponíveis para o servidor web Apache Airflow em seu ambiente Amazon Managed Workflows for Apache Airflow e os recursos adicionais que você precisará configurar em seu Amazon VPC se escolher a opção de rede privada.

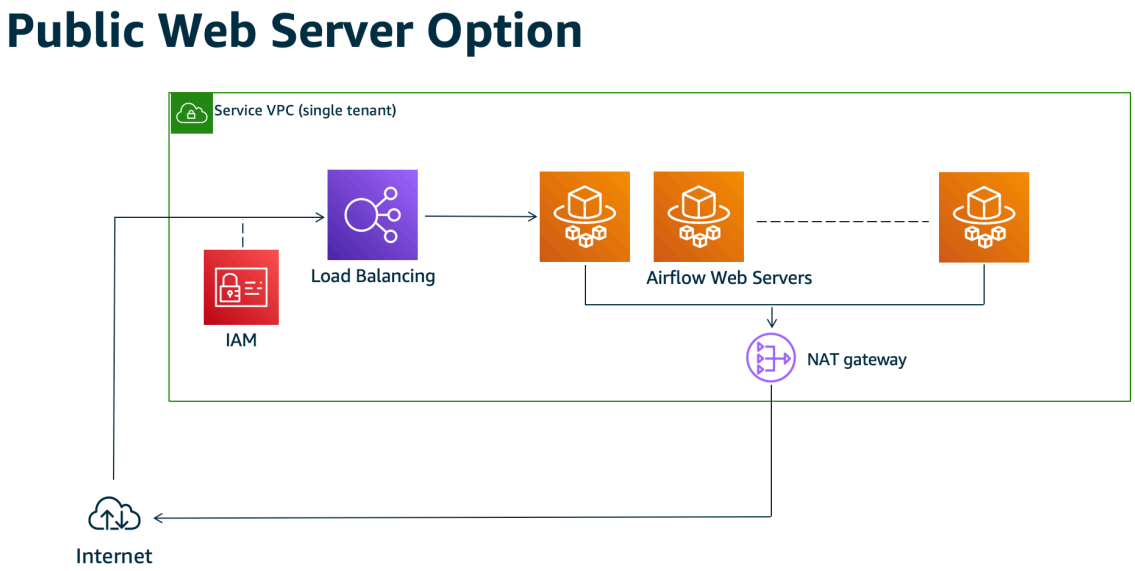
Sumário

Modos de acesso do Apache Airflow

Você pode escolher o roteamento privado ou público para o seu servidor Web Apache Airflow. Para ativar o roteamento privado, escolha Rede privada. Isso limita o acesso do usuário a um servidor Web Apache Airflow dentro de um Amazon VPC. Para habilitar o roteamento público, escolha Rede pública. Isso permite que os usuários acessem o servidor Web Apache Airflow pela Internet.

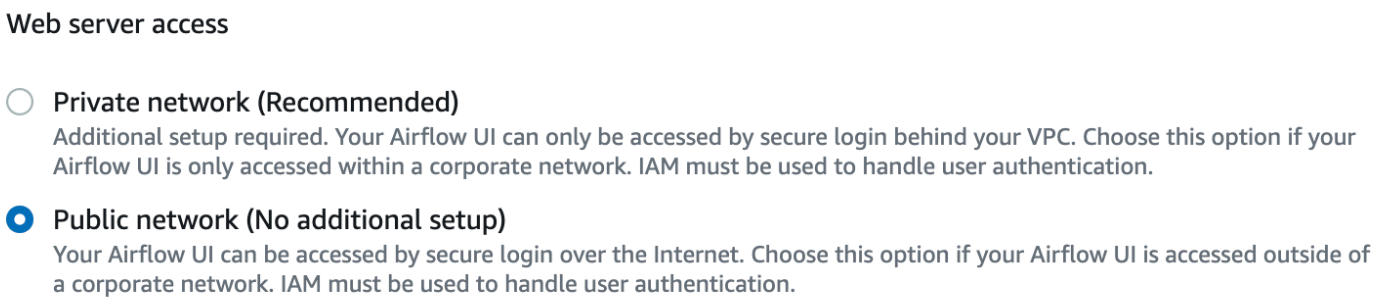
Rede pública

O diagrama de arquitetura a seguir mostra um ambiente do Amazon MWAA com um servidor web público.



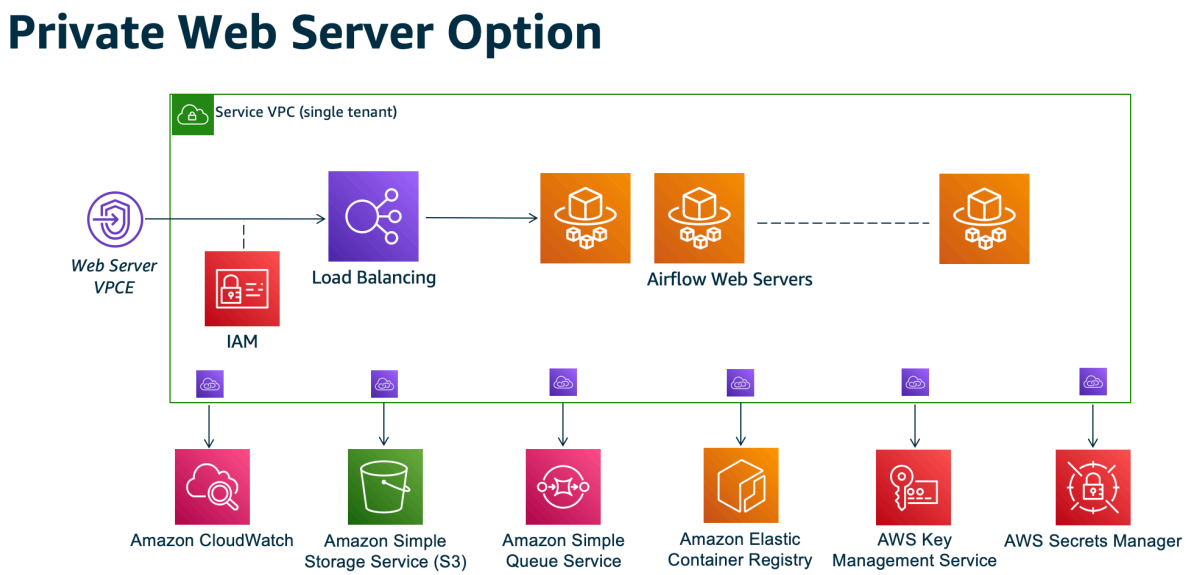
O modo de acesso à rede pública permite que a IU do Apache Airflow seja acessada pela Internet por usuários com acesso à Política do IAM do seu ambiente.

A imagem a seguir mostra onde encontrar a opção Rede pública no console do Amazon MWAA.



Rede privada

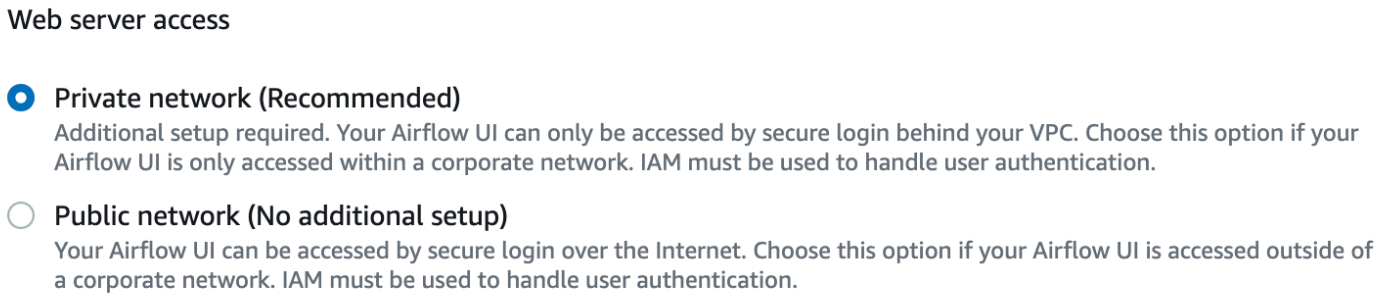
O diagrama de arquitetura a seguir mostra um ambiente do Amazon MWAA com um servidor web privado.



O modo de acesso à rede privada limita o acesso à interface do usuário do Apache Airflow aos usuários da Amazon VPC que receberam acesso à política do IAM do seu ambiente.

Ao criar um ambiente com acesso privado ao servidor web, você deve empacotar todas as suas dependências em um arquivo wheel do Python (`.whl`) e, em seguida, referenciar `.whl` em seu `requirements.txt`. Para obter instruções sobre como empacotar e instalar suas dependências usando o wheel, consulte Gerenciando dependências usando o Python wheel.

A imagem a seguir mostra onde encontrar a opção Rede privada no console do Amazon MWAA.



Visão geral dos modos de acesso

Esta seção descreve os endpoints da VPC (AWS PrivateLink) criados em seu Amazon VPC quando você escolhe o modo de acesso à rede pública ou à rede privada.

Modo de acesso à rede pública

Se você escolher o modo de acesso à rede pública para seu servidor Web Apache Airflow, o tráfego de rede será roteado publicamente pela Internet.

- O Amazon MWAA cria um endpoint de interface VPC para seu banco de dados de metadados Amazon Aurora PostgreSQL. O endpoint é criado nas zonas de disponibilidade mapeadas para suas sub-redes privadas e é independente de outras contas. AWS
- Em seguida, o Amazon MWAA vincula um endereço IP de suas sub-redes privadas aos endpoints da interface. Isso foi projetado para apoiar a prática recomendada de vincular um único IP de cada zona de disponibilidade do Amazon VPC.

Modo de acesso à rede privada

Se você escolheu o modo de acesso à rede privada para seu servidor Web Apache Airflow, o tráfego de rede será roteado de forma privada dentro do Amazon VPC.

- O Amazon MWAA cria um endpoint de interface VPC para seu servidor Web do Apache Airflow e uma interface endpoint para seu banco de dados de metadados Amazon Aurora PostgreSQL. Os endpoints são criados nas zonas de disponibilidade mapeadas para suas sub-redes privadas e são independentes de outras contas. AWS
- Em seguida, o Amazon MWAA vincula um endereço IP de suas sub-redes privadas aos endpoints da interface. Isso foi projetado para apoiar a prática recomendada de vincular um único IP de cada zona de disponibilidade do Amazon VPC.

Para saber mais, consulte the section called “Exemplos de casos de uso para um modo de acesso Amazon VPC e Apache Airflow”.

Configuração para modos de acesso público e privado

A seção a seguir descreve as configurações e as configurações adicionais necessárias de acordo com o modo de acesso escolhido do Apache Airflow para o seu ambiente.

Configuração de rede pública

Se você escolher a opção de rede pública para seu servidor Web do Apache Airflow, poderá começar a usar a IU do Apache Airflow depois de criar seu ambiente.

Você precisará seguir as etapas a seguir para configurar o acesso de seus usuários e a permissão para que seu ambiente use outros AWS serviços.

1. Adicione permissão. O Amazon MWAA precisa de permissão para usar outros AWS serviços. Quando você cria um ambiente, o Amazon MWAA cria uma função vinculada ao serviço que permite usar determinadas ações do IAM para o Amazon Elastic Container Registry (Amazon ECR), Logs e Amazon EC2 CloudWatch .

   Você pode adicionar permissão para usar ações adicionais para esses serviços ou para usar outros AWS serviços adicionando permissões à sua função de execução. Para saber mais, consulte Perfil de execução do Amazon MWAA.
2. Crie políticas de usuário. Pode ser necessário criar várias políticas do IAM para que os usuários configurem o acesso ao seu ambiente e à IU do Apache Airflow. Para saber mais, consulte Como acessar um ambiente do Amazon MWAA.

Configuração de rede privada

Se você escolher a opção de rede privada para seu servidor Web Apache Airflow, precisará configurar o acesso para seus usuários, a permissão para que seu ambiente use outros AWS serviços e criar um mecanismo para acessar os recursos em sua Amazon VPC a partir do seu computador.

1. Adicione permissão. O Amazon MWAA precisa de permissão para usar outros AWS serviços. Quando você cria um ambiente, o Amazon MWAA cria uma função vinculada ao serviço que permite usar determinadas ações do IAM para o Amazon Elastic Container Registry (Amazon ECR), Logs e Amazon EC2 CloudWatch .

Você pode adicionar permissão para usar ações adicionais para esses serviços ou para usar outros AWS serviços adicionando permissões à sua função de execução. Para saber mais, consulte Perfil de execução do Amazon MWAA.

2. Crie políticas de usuário. Pode ser necessário criar várias políticas do IAM para que os usuários configurem o acesso ao seu ambiente e à IU do Apache Airflow. Para saber mais, consulte Como acessar um ambiente do Amazon MWAA.
3. Ative o acesso à rede. Você precisará criar um mecanismo no seu Amazon VPC para se conectar ao endpoint da VPC (AWS PrivateLink) do seu servidor Web Apache Airflow. Por exemplo, criando um túnel VPN a partir do seu computador usando um AWS Client VPN.

Como acessar o endpoint da VPC para seu servidor Web Apache Airflow (acesso à rede privada)

Se você escolheu a opção de rede privada, precisará criar um mecanismo em seu Amazon VPC para acessar o endpoint da VPC (AWS PrivateLink) para seu servidor Web Apache Airflow. Recomendamos usar o mesmo Amazon VPC, grupo de segurança VPC e sub-redes privadas do seu ambiente do Amazon MWAA para esses recursos.

Para saber mais, consulte Gerenciamento de acesso para endpoint da VPC.

Acessando o Apache Airflow

O Amazon MWAA permite que você acesse seu ambiente Apache Airflow usando vários métodos: o console da interface de usuário (UI) do Apache Airflow, a CLI do Apache Airflow e a API REST do Apache Airflow. É possível usar o console do Amazon MWAA para visualizar e invocar um DAG na sua IU do Apache Airflow ou usar as APIs do Amazon MWAA para obter um token e invocar um DAG. Esta sessão descreve as permissões necessárias para acessar a IU do Apache Airflow, como gerar um token para fazer chamadas de API do Amazon MWAA diretamente em seu shell de comando e os comandos compatíveis na CLI do Apache Airflow.

Tópicos

Pré-requisitos

A seção a seguir descreve as etapas preliminares necessárias para usar os comandos e scripts desta seção.

Acesso

- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA em. Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer
- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA. Política completa de acesso à API e ao console: FullApi AmazonMWAA Access

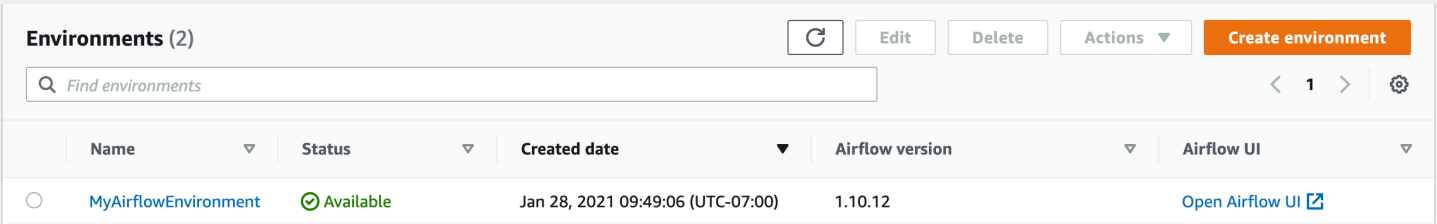
AWS CLI

O AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com AWS serviços usando comandos em seu shell de linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI — Instale a versão 2.
- AWS CLI — Configuração rápida com `aws configure`.

Abrir a IU do Airflow

A imagem a seguir mostra o link para sua IU do Apache Airflow no console Amazon MWAA.



Fazendo login no Apache Airflow

Você precisa de Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer permissões para sua AWS conta no AWS Identity and Access Management (IAM) para visualizar sua interface de usuário do Apache Airflow.

Para acessar sua IU do Apache Airflow

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Abrir a IU do Airflow.

Crie um token de acesso ao servidor web Apache Airflow

Você pode usar os comandos desta página para criar um token de acesso ao servidor web. Um token de acesso permite que você acesse seu ambiente Amazon MWAA. Por exemplo, é possível obter um token e, em seguida, implantar DAGs de forma programática usando as APIs do Amazon

MWAA. A seção a seguir inclui as etapas para criar um token de login na web do Apache Airflow usando o AWS CLI, um script bash, uma solicitação da API POST ou um script Python. O token retornado na resposta é válido por 60 segundos.

Sumário

Pré-requisitos

A seção a seguir descreve as etapas preliminares necessárias para usar os comandos e scripts desta página.

Acesso

- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA em. Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer
- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA. Política completa de acesso à API e ao console: FullApi AmazonMWAA Access

AWS CLI

O AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com AWS serviços usando comandos em seu shell de linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI — Instale a versão 2.
- AWS CLI — Configuração rápida com `aws configure`.

Usando o AWS CLI

O exemplo a seguir usa o create-web-login-tokencomando no AWS CLI para criar um token de login na web do Apache Airflow.

```
aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME
```

Como usar um script bash

O exemplo a seguir usa um script bash para chamar o create-web-login-tokencomando no AWS CLI para criar um token de login na web do Apache Airflow.

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `get-web-token.sh`.

   ```
   #!/bin/bash
   HOST=YOUR_HOST_NAME
   YOUR_URL=https://$HOST/aws_mwaa/aws-console-sso?login=true#
   WEB_TOKEN=$(aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME --query
    WebToken --output text)
   echo $YOUR_URL$WEB_TOKEN
   ```

2. Substitua os espaços reservados em *vermelho* por `YOUR_HOST_NAME` e `YOUR_ENVIRONMENT_NAME`. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

   ```
   123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
   ```

3. (opcional) os usuários do macOS e do Linux podem precisar executar o comando a seguir para garantir que o script seja executável.

   ```
   chmod +x get-web-token.sh
   ```

4. Execute o script a seguir para obter um token de login na web.

   ```
   ./get-web-token.sh
   ```

5. Você deve ver o seguinte em seu prompt de comando:

   ```
   https://123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com/
   aws_mwaa/aws-console-sso?login=true#{your-web-login-token}
   ```

Usar uma solicitação da API POST

O exemplo a seguir usa uma solicitação da API POST para criar um token de login na web do Apache Airflow.

1. Copie a seguinte URL e cole no campo URL do seu cliente da API REST.

```
https://YOUR_HOST_NAME/aws_mwaa/aws-console-sso?login=true#WebToken
```

2. Substitua os espaços reservados em *vermelho* por YOUR_HOST_NAME. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. Copie o seguinte JSON e cole no campo de corpo do seu cliente da API REST.

```
{
  "name": "YOUR_ENVIRONMENT_NAME"
}
```

4. Substitua os espaços reservados em *vermelho* por YOUR_ENVIRONMENT_NAME.
5. Adicione pares de chave-valor no campo de autorização. Por exemplo, se você estiver usando o Postman, escolha Assinatura da AWS e, em seguida, insira seu:
   - AWS_ACCESS_KEY_ID no AccessKey
   - AWS_SECRET_ACCESS_KEY no SecretKey
6. Você deverá ver a seguinte resposta:

```
{
  "webToken": "<Short-lived token generated for enabling access to the Apache
 Airflow Webserver UI>",
  "webServerHostname": "<Hostname for the WebServer of the environment>"
}
```

Como usar um script Python

O exemplo a seguir usa o método boto3 create_web_login_token em um script Python para criar um token da CLI do Apache Airflow e acionar um DAG. É possível executar esse script fora do Amazon

MWAA. Você só precisa instalar a biblioteca boto3. Talvez você queira criar um ambiente virtual para instalar a biblioteca. Ele pressupõe que você tenha configurado as credenciais de AWS autenticação para sua conta.

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `create-web-login-token.py`.

```
import boto3
mwaa = boto3.client('mwaa')
response = mwaa.create_web_login_token(
    Name="YOUR_ENVIRONMENT_NAME"
)
webServerHostName = response["WebServerHostname"]
webToken = response["WebToken"]
airflowUIUrl = 'https://{0}/aws_mwaa/aws-console-sso?
login=true#{1}'.format(webServerHostName, webToken)
print("Here is your Airflow UI URL: ")
print(airflowUIUrl)
```

2. Substitua os espaços reservados em *vermelho* por `YOUR_ENVIRONMENT_NAME`.
3. Execute o script a seguir para obter um token de login na web.

```
python3 create-web-login-token.py
```

Próximas etapas

- Explore a operação da API Amazon MWAA usada para criar um token de login na web em. CreateWebLoginToken

Como criar um token da CLI do Apache Airflow

Você pode usar os comandos nesta página para gerar um token da CLI e, em seguida, fazer chamadas de API do Amazon Managed Workflows for Apache Airflow diretamente no seu shell de comando. Por exemplo, você pode obter um token e, em seguida, implantar DAGs de forma programática usando as APIs do Amazon MWAA. A seção a seguir inclui as etapas para criar um token da CLI do Apache Airflow usando a AWS CLI, um script cURL, um script Python ou um script bash. O token retornado na resposta é válido por 60 segundos.

> **Note**
>
> O token AWS CLI é destinado a substituir as ações síncronas do shell, não os comandos assíncronos da API. Dessa forma, a simultaneidade disponível é limitada. Para garantir que o servidor web permaneça responsivo para os usuários, é recomendável não abrir uma nova solicitação de AWS CLI até que a anterior seja concluída com êxito.

Sumário

Pré-requisitos

A seção a seguir descreve as etapas preliminares necessárias para usar os comandos e scripts desta página.

Acesso

- Acesso à conta AWS em AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA em Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer.
- Acesso à conta AWS em AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA em Política completa de acesso à API e ao console: FullApi AmazonMWAA Access.

AWS CLI

A AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com os serviços da AWS usando comandos no shell da linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI: instalar a versão 2.
- AWS CLI: configuração rápida com `aws configure`.

Usar a AWS CLI

O exemplo a seguir usa o comando create-cli-token na AWS CLI para criar um token da CLI do Apache Airflow.

```
aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME
```

Como usar um script cURL

O exemplo a seguir usa um script cURL para chamar o comando create-web-login-token na AWS CLI para invocar a CLI do Apache Airflow por meio de um endpoint no servidor web Apache Airflow.

Apache Airflow v2

1. Copie a instrução cURL do seu arquivo de texto e cole-a no shell de comando.

   > **Note**
   >
   > Depois de copiá-la para a área de transferência, talvez seja necessário usar Editar > Colar a partir do seu menu shell.

   ```
   CLI_JSON=$(aws mwaa --region YOUR_REGION create-cli-token --
   name YOUR_ENVIRONMENT_NAME) \
     && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
     && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
     && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/
   cli" \
     --header "Authorization: Bearer $CLI_TOKEN" \
     --header "Content-Type: text/plain" \
   ```

```
  --data-raw "dags trigger YOUR_DAG_NAME") \
  && echo "Output:" \
  && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
  && echo "Errors:" \
  && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

2. Substitua os espaços reservados por YOUR_REGION com a região da AWS para seu ambiente, YOUR_DAG_NAME e YOUR_ENVIRONMENT_NAME. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. Você deve ver o seguinte em seu prompt de comando:

```
{
  "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
  "stdout":"<STDOUT of the CLI execution, base64 encoded>"
}
```

Apache Airflow v1

1. Copie a instrução cURL do seu arquivo de texto e cole-a no shell de comando.

> **Note**
>
> Depois de copiá-la para a área de transferência, talvez seja necessário usar Editar > Colar a partir do seu menu shell.

```
CLI_JSON=$(aws mwaa --region YOUR_REGION create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
  && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
  && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
  && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/
cli" \
  --header "Authorization: Bearer $CLI_TOKEN" \
  --header "Content-Type: text/plain" \
  --data-raw "trigger_dag YOUR_DAG_NAME") \
  && echo "Output:" \
  && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
```

```
  && echo "Errors:" \
  && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

2. Substitua os espaços reservados por YOUR_REGION com a região da AWS para seu ambiente, YOUR_DAG_NAME e YOUR_HOST_NAME. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. Você deve ver o seguinte em seu prompt de comando:

```
{
  "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
  "stdout":"<STDOUT of the CLI execution, base64 encoded>"
}
```

4. Substitua os espaços reservados por YOUR_ENVIRONMENT_NAME e YOUR_DAG_NAME.

Como usar um script bash

O exemplo a seguir usa um script bash para chamar o comando create-cli-token na AWS CLI para criar um token da CLI do Apache Airflow.

Apache Airflow v2

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `get-cli-token.sh`.

```
# brew install jq
  aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export CLI_TOKEN=$(jq
 -r .CliToken) && curl --request POST "https://YOUR_HOST_NAME/aws_mwaa/cli" \
      --header "Authorization: Bearer $CLI_TOKEN" \
      --header "Content-Type: text/plain" \
      --data-raw "dags trigger YOUR_DAG_NAME"
```

2. Substitua os espaços reservados em *vermelho* por YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME e YOUR_DAG_NAME. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. (opcional) os usuários do macOS e do Linux podem precisar executar o comando a seguir para garantir que o script seja executável.

```
chmod +x get-cli-token.sh
```

4. Execute o script a seguir para criar um token da CLI do Apache Airflow.

```
./get-cli-token.sh
```

Apache Airflow v1

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `get-cli-token.sh`.

```
# brew install jq
  aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export CLI_TOKEN=$(jq
 -r .CliToken) && curl --request POST "https://YOUR_HOST_NAME/aws_mwaa/cli" \
      --header "Authorization: Bearer $CLI_TOKEN" \
      --header "Content-Type: text/plain" \
      --data-raw "trigger_dag YOUR_DAG_NAME"
```

2. Substitua os espaços reservados em *vermelho* por YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME e YOUR_DAG_NAME. Por exemplo, um nome de host para uma rede pública pode ter a seguinte aparência (sem o https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. (opcional) os usuários do macOS e do Linux podem precisar executar o comando a seguir para garantir que o script seja executável.

```
chmod +x get-cli-token.sh
```

4. Execute o script a seguir para criar um token da CLI do Apache Airflow.

```
./get-cli-token.sh
```

Como usar um script Python

O exemplo a seguir usa o método boto3 create_cli_token em um script Python para criar um token da CLI do Apache Airflow e acionar um DAG. Você pode executar esse script fora do Amazon MWAA. A única coisa que você precisa fazer é instalar a biblioteca boto3. Talvez você queira criar um ambiente virtual para instalar a biblioteca. Ele pressupõe que você tenha configurado as credenciais de autenticação da AWS para sua conta.

Apache Airflow v2

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `create-cli-token.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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.
"""
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
```

```
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. Substitua os espaços reservados por `YOUR_ENVIRONMENT_NAME` e `YOUR_DAG_NAME`.
3. Execute o script a seguir para criar um token da CLI do Apache Airflow.

```
python3 create-cli-token.py
```

Apache Airflow v1

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `create-cli-token.py`.

```
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
```

```
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'trigger_dag'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. Substitua os espaços reservados por `YOUR_ENVIRONMENT_NAME` e `YOUR_DAG_NAME`.
3. Execute o script a seguir para criar um token da CLI do Apache Airflow.

```
python3 create-cli-token.py
```

Próximas etapas

- Explore a operação da API Amazon MWAA usada para criar um token da CLI em CreateCliToken (Criar token da CLI).

Usando a API REST do Apache Airflow

O Amazon Managed Workflows for Apache Airflow (Amazon MWAA) oferece suporte à interação com seus ambientes Apache Airflow diretamente usando a API REST do Apache Airflow para ambientes que executam o Apache Airflow v2.4.3 e superior. Isso permite acessar e gerenciar seus ambientes Amazon MWAA de forma programática, fornecendo uma forma padronizada de invocar fluxos de trabalho de orquestração de dados, gerenciar seus DAGs e monitorar o status de vários componentes do Apache Airflow, como o banco de dados de metadados, o acionador e o agendador.

Para oferecer suporte direto ao uso da API REST do Apache Airflow, o Amazon MWAA oferece a opção de escalar horizontalmente a capacidade do servidor web para lidar com o aumento da demanda, seja de solicitações da API REST, uso da interface de linha de comando (CLI) ou mais usuários simultâneos da interface de usuário (UI) do Apache Airflow. Para obter mais informações sobre como o Amazon MWAA escala servidores web, consulte. the section called “Configurando o escalonamento automático do servidor web”

Você pode usar a API REST do Apache Airflow para implementar os seguintes casos de uso em seus ambientes:

- Acesso programático — Agora você pode iniciar as execuções do Apache Airflow DAG, gerenciar conjuntos de dados e recuperar o status de vários componentes, como banco de dados de metadados, acionadores e agendadores, sem depender da interface de usuário ou da CLI do Apache Airflow.
- Integre-se com aplicativos e microsserviços externos — O suporte à API REST permite que você crie soluções personalizadas que integram seus ambientes Amazon MWAA com outros sistemas. Por exemplo, você pode iniciar fluxos de trabalho em resposta a eventos de sistemas externos, como trabalhos concluídos no banco de dados ou inscrições de novos usuários.
- Monitoramento centralizado — Você pode criar painéis de monitoramento que agregam o status de seus DAGs em vários ambientes Amazon MWAA, permitindo monitoramento e gerenciamento centralizados.

Os tópicos a seguir mostram como você obtém um token de acesso ao servidor web e, em seguida, usa esse token para fazer chamadas de API para a API REST do Apache Airflow. No exemplo a seguir, você chamará a API para iniciar uma nova execução do DAG.

Para obter mais informações sobre a API REST do Apache Airflow, consulte A referência da API REST do Apache Airflow.

Tópicos

Crie um token de sessão do servidor web

Para criar um token de acesso ao servidor web, use a função Python a seguir. Essa função primeiro chama a API Amazon MWAA para obter um token de login na web. O token de login da web, que expira após 60 segundos, é então trocado por um token de sessão da web, que permite acessar o servidor web e usar a API REST do Apache Airflow.

> **Note**
>
> O token da sessão expira após 12 horas.

```
def get_session_info(region, env_name):
    logging.basicConfig(level=logging.INFO)

    try:
        # Initialize MWAA client and request a web login token
        mwaa = boto3.client('mwaa', region_name=region)
        response = mwaa.create_web_login_token(Name=env_name)

        # Extract the web server hostname and login token
        web_server_host_name = response["WebServerHostname"]
        web_token = response["WebToken"]

        # Construct the URL needed for authentication
        login_url = f"https://{web_server_host_name}/aws_mwaa/login"
        login_payload = {"token": web_token}

        # Make a POST request to the MWAA login url using the login payload
        response = requests.post(
            login_url,
            data=login_payload,
            timeout=10
        )

        # Check if login was succesfull
```

```
        if response.status_code == 200:

            # Return the hostname and the session cookie
            return (
                web_server_host_name,
                response.cookies["session"]
            )
        else:
            # Log an error
            logging.error("Failed to log in: HTTP %d", response.status_code)
            return None
    except requests.RequestException as e:
         # Log any exceptions raised during the request to the MWAA login endpoint
        logging.error("Request failed: %s", str(e))
        return None
    except Exception as e:
        # Log any other unexpected exceptions
        logging.error("An unexpected error occurred: %s", str(e))
        return None
```

Chame a API REST do Apache Airflow

Depois que a autenticação for concluída, você terá as credenciais para começar a enviar solicitações aos endpoints da API. No exemplo abaixo, use o endpoint/dags/*dag_id*/dag.

```
def trigger_dag(region, env_name, dag_name):
    """
    Triggers a DAG in a specified MWAA environment using the Airflow REST API.

    Args:
    region (str): AWS region where the MWAA environment is hosted.
    env_name (str): Name of the MWAA environment.
    dag_name (str): Name of the DAG to trigger.
    """

    logging.info(f"Attempting to trigger DAG {dag_name} in environment {env_name} at
 region {region}")

    # Retrieve the web server hostname and session cookie for authentication
    try:
        web_server_host_name, session_cookie = get_session_info(region, env_name)
        if not session_cookie:
```

```
            logging.error("Authentication failed, no session cookie retrieved.")
            return
    except Exception as e:
        logging.error(f"Error retrieving session info: {str(e)}")
        return

    # Prepare headers and payload for the request
    cookies = {"session": session_cookie}
    json_body = {"conf": {}}

    # Construct the URL for triggering the DAG
    url = f"https://{web_server_host_name}/api/v1/dags/{dag_id}/dagRuns"

    # Send the POST request to trigger the DAG
    try:
        response = requests.post(url, cookies=cookies, json=json_body)
        # Check the response status code to determine if the DAG was triggered
 successfully
        if response.status_code == 200:
            logging.info("DAG triggered successfully.")
        else:
            logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
 {response.text}")
    except requests.RequestException as e:
        logging.error(f"Request to trigger DAG failed: {str(e)}")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 4:
        logging.error("Incorrect usage. Proper format: python script_name.py {region}
 {env_name} {dag_name}")
        sys.exit(1)

    region = sys.argv[1]
    env_name = sys.argv[2]
    dag_name = sys.argv[3]

    # Trigger the DAG with the provided arguments
    trigger_dag(region, env_name, dag_name)
```

Referência de comandos da CLI do Apache Airflow

Esta página descreve os comandos da CLI do Apache Airflow compatíveis e não compatíveis no Amazon Managed Workflows para Apache Airflow.

Sumário

Pré-requisitos

A seção a seguir descreve as etapas preliminares necessárias para usar os comandos e scripts desta página.

Acesso

- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA em. Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer
- AWS acesso à conta AWS Identity and Access Management (IAM) à política de permissões do Amazon MWAA. Política completa de acesso à API e ao console: FullApi AmazonMWAA Access

AWS CLI

O AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite que você interaja com AWS serviços usando comandos em seu shell de linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI — Instale a versão 2.
- AWS CLI — Configuração rápida com `aws configure`.

O que mudou na v2

- Novo: estrutura de comandos da CLI do Airflow. A CLI do Apache Airflow v2 é organizada para que os comandos relacionados sejam agrupados como subcomandos, o que significa que você precisa atualizar os scripts do Apache Airflow v1 se quiser atualizar para o Apache Airflow v2. Por exemplo, `unpause` no Apache Airflow v1 agora é `dags unpause` no Apache Airflow v2. Para saber mais, consulte Alterações na CLI do Airflow em 2 no Guia de referência do Apache Airflow.

Comandos CLI compatíveis

A seção a seguir lista os comandos da CLI do Apache Airflow disponíveis no Amazon MWAA.

Comandos compatíveis

Apache Airflow v2

Versões secundárias	Command
v2.0+	cheat-sheet
v2.0+	connections add
v2.0+	connections delete
v2.2+ (nota)	dags backfill
v2.0+	dags delete
v2.2+ (nota)	dags list

Versões secundárias	Command
v2.0+	dags list-jobs
v2.6+	punhetas list-import-errors
v2.2+ (nota)	dags list-runs
v2.2+ (nota)	dags next-execution
v2.0+	dags pause
v2.0+	dags report
v2.4+	dags reserialize
v2.0+	dags show
v2.0+	dags state
v2.0+	dags test
v2.0+	dags trigger
v2.0+	dags unpause
v2.4+	db clean
v2.0+	providers behaviours
v2.0+	providers get
v2.0+	providers hooks
v2.0+	providers links
v2.0+	providers list
v2.8+	notificações de provedores
v2.6+	providers secrets

Versões secundárias	Command
v2.7+	providers triggerer
v2.0+	providers widgets
v2.6+	roles add-perms
v2.6+	roles del-perms
v2.6+	funções criam
v2.0+	roles list
v2.0+	tasks clear
v2.0+	tasks failed-deps
v2.0+	tasks list
v2.0+	tasks render
v2.0+	tasks state
v2.0+	tarefas states-for-dag-run
v2.0+	tasks test
v2.0+	variables delete
v2.0+	variables get
v2.0+	variables set
v2.0+	variables list
v2.0+	version

Como usar comandos que analisam DAGs

Se seu ambiente estiver executando o Apache Airflow v1.10.12 ou v2.0.2, os comandos da CLI que analisam DAGs falharão se o DAG usar plug-ins que dependem de pacotes instalados por meio de: `requirements.txt`

Apache Airflow v2.0.2

- `dags backfill`
- `dags list`
- `dags list-runs`
- `dags next-execution`

Você pode usar estes comandos da CLI se seus DAGs não usarem plug-ins que dependam de pacotes instalados por meio de um `requirements.txt`.

Código de exemplo

A seção a seguir contém exemplos de maneiras diferentes de usar a CLI do Apache Airflow.

Definir, obter ou excluir uma variável do Apache Airflow v2

É possível usar o código de exemplo a seguir para definir, obter ou excluir uma variável no formato de `<script> <mwaa env name> get | set | delete <variable> <variable value> </variable> </variable>`.

```
[ $# -eq 0 ] && echo "Usage: $0 MWAA environment name " && exit

if [[ $2 == "" ]]; then
    dag="variables list"

elif  [ $2 == "get" ] ||  [ $2 == "delete" ] ||  [ $2 == "set" ]; then
    dag="variables $2 $3 $4 $5"

else
    echo "Not a valid command"
    exit 1
fi

CLI_JSON=$(aws mwaa --region $AWS_REGION create-cli-token --name $1) \
```

```
    && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
    && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
    && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/cli" \
    --header "Authorization: Bearer $CLI_TOKEN" \
    --header "Content-Type: text/plain" \
    --data-raw "$dag" ) \
    && echo "Output:" \
    && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
    && echo "Errors:" \
    && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

Adicionar uma configuração ao acionar um DAG

Você pode usar o código de exemplo a seguir com o Apache Airflow v1 e o Apache Airflow v2 para adicionar uma configuração ao acionar um DAG, como `airflow trigger_dag 'dag_name' –conf '{"key":"value"}'`.

```
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
key = "YOUR_KEY"
value = "YOUR_VALUE"
conf = "{\"" + key + "\":\"" + value + "\"}"

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
  Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = "trigger_dag {0} -c '{1}'".format(dag_name, conf)

mwaa_response = requests.post(
      mwaa_webserver_hostname,
      headers={
          'Authorization': mwaa_auth_token,
```

```
          'Content-Type': 'text/plain'
          },
      data=raw_data
      )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

Execute comandos CLI em um túnel SSH para um bastion host

O exemplo a seguir mostra como executar comandos da Airflow CLI usando um proxy de túnel SSH para um Linux Bastion Host.

Como usar cURL

1. ```
   ssh -D 8080 -f -C -q -N YOUR_USER@YOUR_BASTION_HOST
   ```
2. ```
   curl -x socks5h://0:8080 --request POST https://YOUR_HOST_NAME/aws_mwaa/cli --
   header YOUR_HEADERS --data-raw YOUR_CLI_COMMAND
   ```

Amostras GitHub e AWS tutoriais

- Trabalhando com parâmetros e variáveis do Apache Airflow v2.0.2 no Amazon Managed Workflows for Apache Airflow
- Interagindo com o Apache Airflow v1.10.12 no Amazon MWAA por meio da linha de comando
- Comandos interativos com o Apache Airflow v1.10.12 no Amazon MWAA e o Bash Operator no GitHub

Como gerenciar conexões com o Apache Airflow

Esta seção descreve as diferentes formas de configurar uma conexão do Apache Airflow para um ambiente do Amazon Managed Workflows for Apache Airflow.

Tópicos

Visão geral das variáveis e conexões do Apache Airflow

Em alguns casos, talvez você queira especificar conexões ou variáveis adicionais para um ambiente, como um perfil AWS, ou adicionar seu perfil de execução em um objeto de conexão no metastore do Apache Airflow e, em seguida, consultar a conexão de dentro de um DAG.

- Apache Airflow autogerenciado. Em uma instalação autogerenciada do Apache Airflow, você define as opções de configuração do Apache Airflow em `airflow.cfg`.

```
[secrets]
backend = airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
backend_kwargs = {"connections_prefix" : "airflow/connections", "variables_prefix" :
 "airflow/variables"}
```

- Apache Airflow no Amazon MWAA. No Amazon MWAA, você precisa adicionar essas definições de configuração como opções de configuração do Apache Airflow no console do Amazon MWAA. As opções de configuração do Apache Airflow são gravadas como variáveis de ambiente em seu ambiente e substituem todas as outras configurações existentes pela mesma configuração.

Pacotes do provedor Apache Airflow instalados em ambientes Amazon MWAA

O Amazon MWAA instala extras do provedor para os tipos de conexão Apache Airflow v2 e superiores quando você cria um novo ambiente. A instalação de pacotes do provedor permite que

você visualize um tipo de conexão na IU do Apache Airflow. Também significa que você não precisa especificar esses pacotes como uma dependência do Python em seu arquivo `requirements.txt`. Esta página lista os pacotes do provedor Apache Airflow instalados pelo Amazon MWAA para todos os ambientes do Apache Airflow v2.

> **Note**
>
> Para o Apache Airflow v2 e superior, o Amazon MWAA instala a Watchtower versão 2.0.1 após a execuçãopip3 `install -r requirements.txt`, para garantir que a compatibilidade com o registro não CloudWatch seja substituída por outras instalações da biblioteca Python.

Sumário

Pacotes de provedores para conexões do Apache Airflow v2.8.1

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.8.1, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

> **Note**
>
> Você pode especificar a versão mais recente compatível de `apache-airflow-providers-amazon` para atualizar este provedor. Para obter mais informações sobre como especificar versões mais novas, consulte the section called “Como especificar pacotes de fornecedores mais novos”.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon[aiobotocore] == 8.16.0
Conexão com o Postgres	apache-airflow-providers-postgres==5.10.0
Conexão FTP	apache-airflow-providers-ftp==3.7.0
Conexão Celery	apache-airflow-providers-celery==3.5.1
Conexão HTTP	apache-airflow-providers-http==4.8.0
Conexão IMAP	apache-airflow-providers-imap==3.5.0
SQL comum	apache-airflow-providers-common-sql==1.10.0
Conexão SQLite	apache-airflow-providers-sqlite==3.7.0

Pacotes de provedores para conexões Apache Airflow v2.7.2

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.7.2, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

> **Note**
>
> Você pode especificar a versão mais recente compatível de `apache-airflow-providers-amazon` para atualizar este provedor. Para obter mais informações sobre como especificar versões mais novas, consulte the section called “Como especificar pacotes de fornecedores mais novos”.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon[aiobotocore] == 8.7.1
Conexão com o Postgres	apache-airflow-providers-postgres==5.6.1

Connection type	Pacote
Conexão FTP	apache-airflow-providers-ftp==3.5.2
Conexão Celery	apache-airflow-providers-celery==3.3.4
Conexão HTTP	apache-airflow-providers-http==4.5.2
Conexão IMAP	apache-airflow-providers-imap==3.3.2
SQL comum	apache-airflow-providers-common-sql==1.7.2
Conexão SQLite	apache-airflow-providers-sqlite==3.4.3

Pacotes de provedores para conexões Apache Airflow v2.6.3

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.6.3, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

> **Note**
>
> Você pode especificar a versão mais recente compatível de `apache-airflow-providers-amazon` para atualizar este provedor. Para obter mais informações sobre como especificar versões mais novas, consulte the section called “Como especificar pacotes de fornecedores mais novos”.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon[aiobotocore]==8.2.0
Conexão com o Postgres	apache-airflow-providers-postgres==5.5.1
Conexão FTP	apache-airflow-providers-ftp==3.4.2
Conexão Celery	apache-airflow-providers-celery==3.2.1
Conexão HTTP	apache-airflow-providers-http==4.4.2

Connection type	Pacote
Conexão IMAP	apache-airflow-providers-imap==3.2.2
SQL comum	apache-airflow-providers-common-sql==1.5.2
Conexão SQLite	apache-airflow-providers-sqlite==3.4.2

Pacotes de provedores para conexões Apache Airflow v2.5.1

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.5.1, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

> **Note**
>
> Você pode especificar a versão mais recente compatível de `apache-airflow-providers-amazon` para atualizar este provedor. Para obter mais informações sobre como especificar versões mais novas, consulte the section called “Como especificar pacotes de fornecedores mais novos”.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon==7.1.0
Conexão com o Postgres	apache-airflow-providers-postgres==5.4.0
Conexão FTP	apache-airflow-providers-ftp==3.3.0
Conexão Celery	apache-airflow-providers-celery==3.1.0
Conexão HTTP	apache-airflow-providers-http==4.1.1
Conexão IMAP	apache-airflow-providers-imap==3.1.1
SQL comum	apache-airflow-providers-common-sql==1.3.3
Conexão SQLite	apache-airflow-providers-sqlite==3.3.1

Pacotes de provedores para conexões Apache Airflow v2.4.3

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.4.3, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon==6.0.0
Conexão com o Postgres	apache-airflow-providers-postgres==5.2.2
Conexão FTP	apache-airflow-providers-ftp==3.1.0
Conexão Celery	apache-airflow-providers-celery==3.0.0
Conexão HTTP	apache-airflow-providers-http==4.0.0
Conexão IMAP	apache-airflow-providers-imap==3.0.0
SQL comum	apache-airflow-providers-common-sql==1.2.0
Conexão SQLite	apache-airflow-providers-sqlite==3.2.1

Pacotes de provedores para conexões Apache Airflow v2.2.2

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.2.2, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

Connection type	Pacote
AWS Conexão	apache-airflow-providers-amazon==2.4.0
Conexão com o Postgres	apache-airflow-providers-postgres==2.3.0
Conexão FTP	apache-airflow-providers-ftp==2.0.1
Conexão Celery	apache-airflow-providers-celery==2.1.0
Conexão HTTP	apache-airflow-providers-http==2.0.1

Connection type	Pacote
Conexão IMAP	apache-airflow-providers-imap==2.0.1
Conexão SQLite	apache-airflow-providers-sqlite==2.0.1

Pacotes de provedores para conexões Apache Airflow v2.0.2

Quando você cria um ambiente Amazon MWAA no Apache Airflow v2.0.2, o Amazon MWAA instala os seguintes pacotes de provedores usados para conexões do Apache Airflow.

Connection type	Pacote
Conexão Tableau	apache-airflow-providers-tableau==1.0.0
Conexão Databricks	apache-airflow-providers-databricks==1.0.1
Conexão SSH	apache-airflow-providers-ssh==1.3.0
Conexão com o Postgres	apache-airflow-providers-postgres==1.0.2
Conexão Docker	apache-airflow-providers-docker==1.2.0
Conexão Oracle	apache-airflow-providers-oracle==1.1.0
Conexão Presto	apache-airflow-providers-presto==1.0.2
Conexão SFTP	apache-airflow-providers-sftp==1.2.0

Como especificar pacotes de fornecedores mais novos

A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.

Os arquivos de restrições do Apache Airflow especificam as versões do provedor disponíveis no momento de um lançamento do Apache Airflow. No entanto, em muitos casos os provedores mais

novos são compatíveis com essa versão do Apache Airflow. Como você deve usar restrições, para especificar uma versão mais recente de um pacote de provedor, você pode modificar o arquivo de restrições para uma versão específica do provedor:

1. Baixe o arquivo de restrições específicas da versão em https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt"
2. Modifique a versão `apache-airflow-providers-amazon` no arquivo de restrições para a versão que você deseja usar.
3. Salve o arquivo de restrições modificado na pasta DAGs do Amazon S3 do seu ambiente Amazon MWAA, por exemplo, como `constraints-3.11-updated.txt`
4. Especifique seus requisitos conforme mostrado a seguir.

```
--constraint "/usr/local/airflow/dags/constraints-3.11-updated.txt"

apache-airflow-providers-amazon==version-number
```

> **Note**
>
> Se você estiver usando um servidor web privado, recomendamos que você empacote as bibliotecas necessárias como arquivos WHL usando o executor local Amazon MWAA.

Visão geral dos tipos de conexão

O Apache Airflow armazena conexões, como uma string de conexão URI. É fornecido um modelo de conexões na IU do Apache Airflow para gerar a string de conexão URI, independentemente do tipo de conexão. Se um modelo de conexão não estiver disponível na IU do Apache Airflow, um modelo de conexão alternativo poderá ser usado para gerar essa string de conexão URI, como o uso do modelo de conexão HTTP. A principal diferença é o prefixo do URI, como `my-conn-type://`, que os provedores do Apache Airflow normalmente ignoram para uma conexão. Esta página descreve como usar modelos de conexão na IU do Apache Airflow de forma intercambiável para diferentes tipos de conexão.

> **Warning**
>
> Não substitua a conexão `aws_default` no Amazon MWAA. O Amazon MWAA usa essa conexão para realizar uma variedade de tarefas críticas, como coletar logs de tarefas.

Substituir essa conexão pode resultar em perda de dados e interrupções na disponibilidade do seu ambiente.

Tópicos

Exemplo de string de conexão URI

O exemplo a seguir mostra uma string do URI de conexão do tipo de conexão do MySQL.

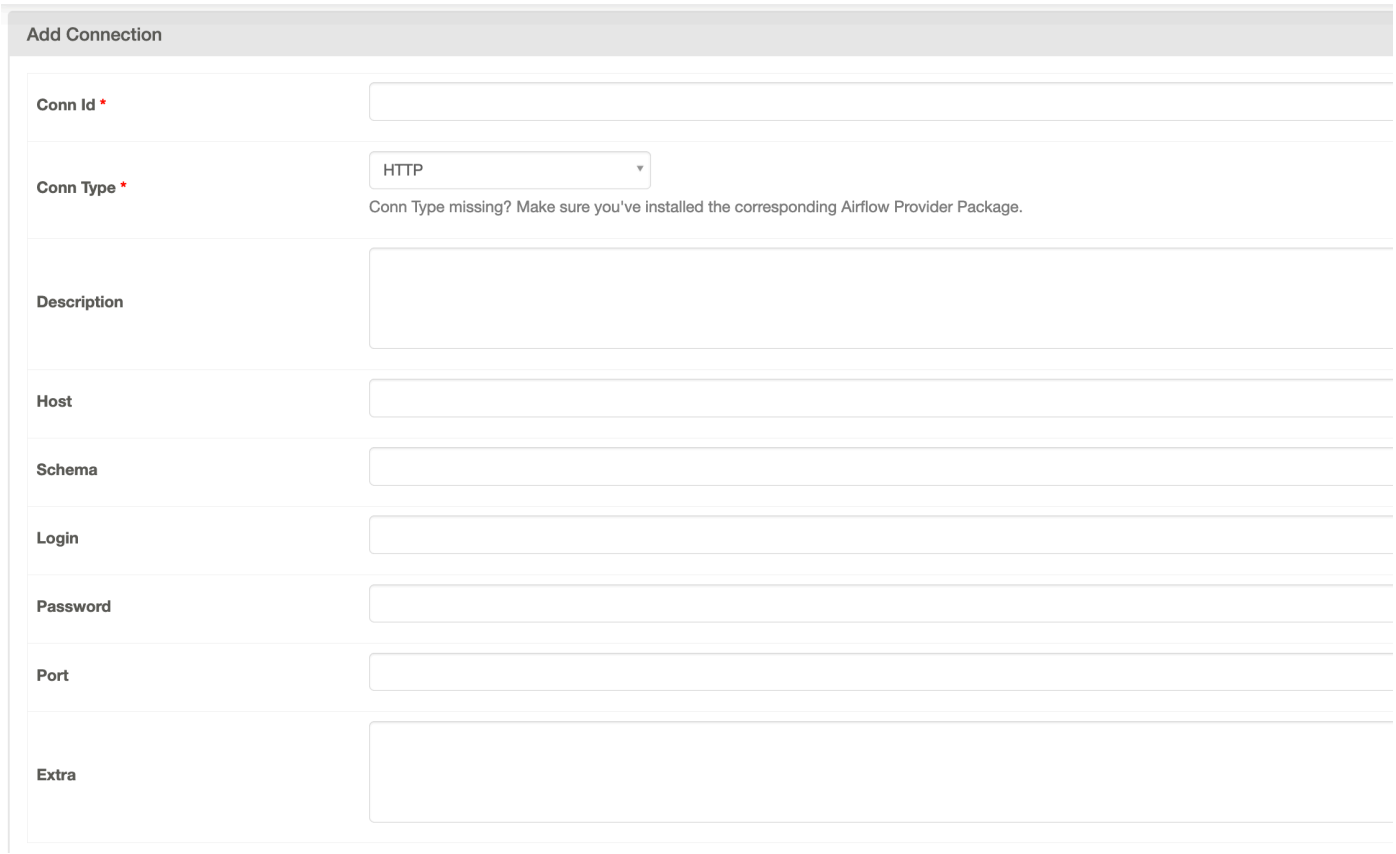
```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role%2FAmazonMWAA-
MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

Exemplo de modelo de conexão

O exemplo a seguir mostra o modelo de conexão HTTP na interface do Apache Airflow.

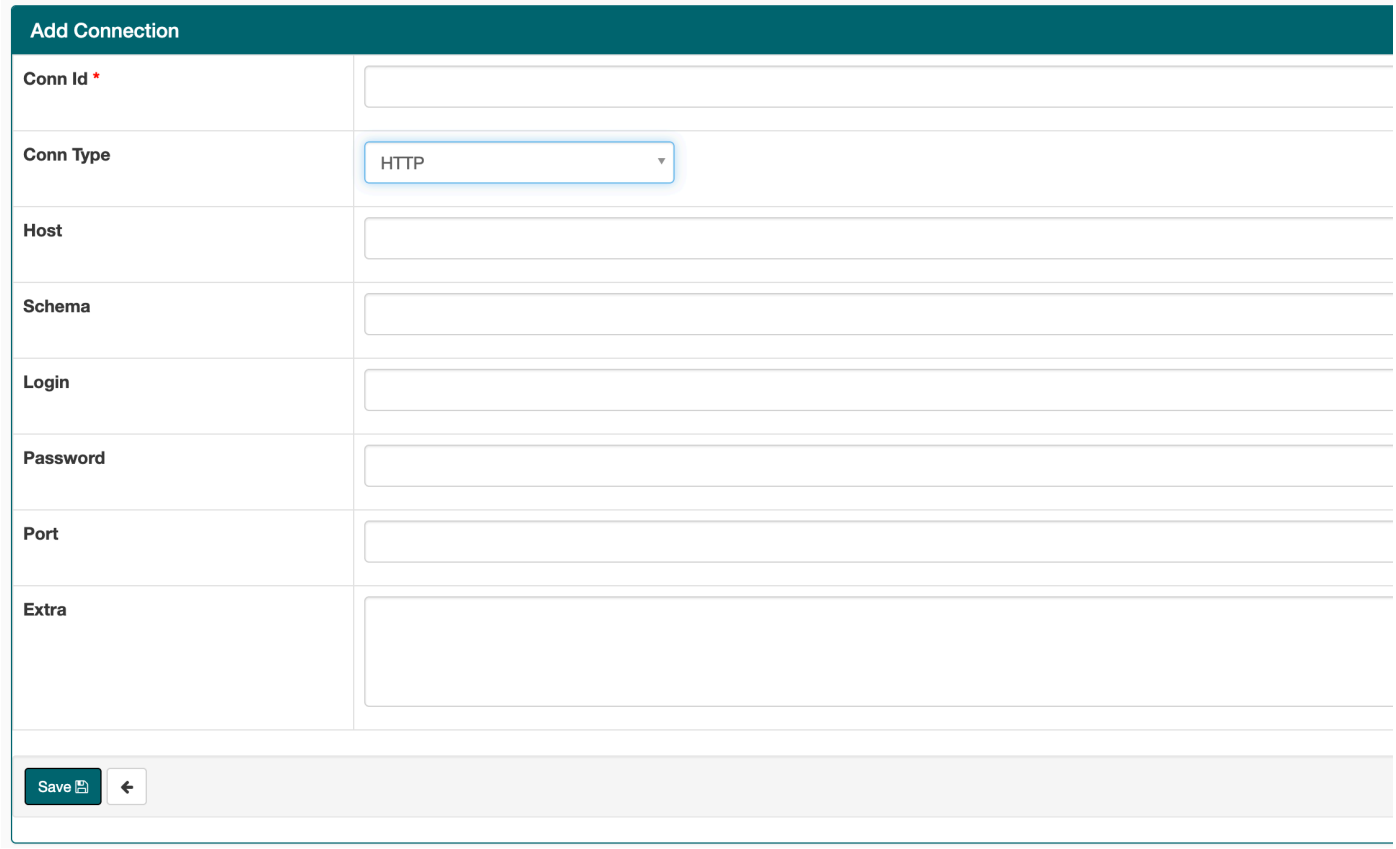
Apache Airflow v2

O exemplo a seguir mostra o modelo de conexão HTTP para o Apache Airflow v2 na IU do Apache Airflow.



Apache Airflow v1

O exemplo a seguir mostra o modelo de conexão HTTP para o Apache Airflow v1 na IU do Apache Airflow.



Exemplo de uso de um modelo de conexão HTTP para uma conexão Jdbc

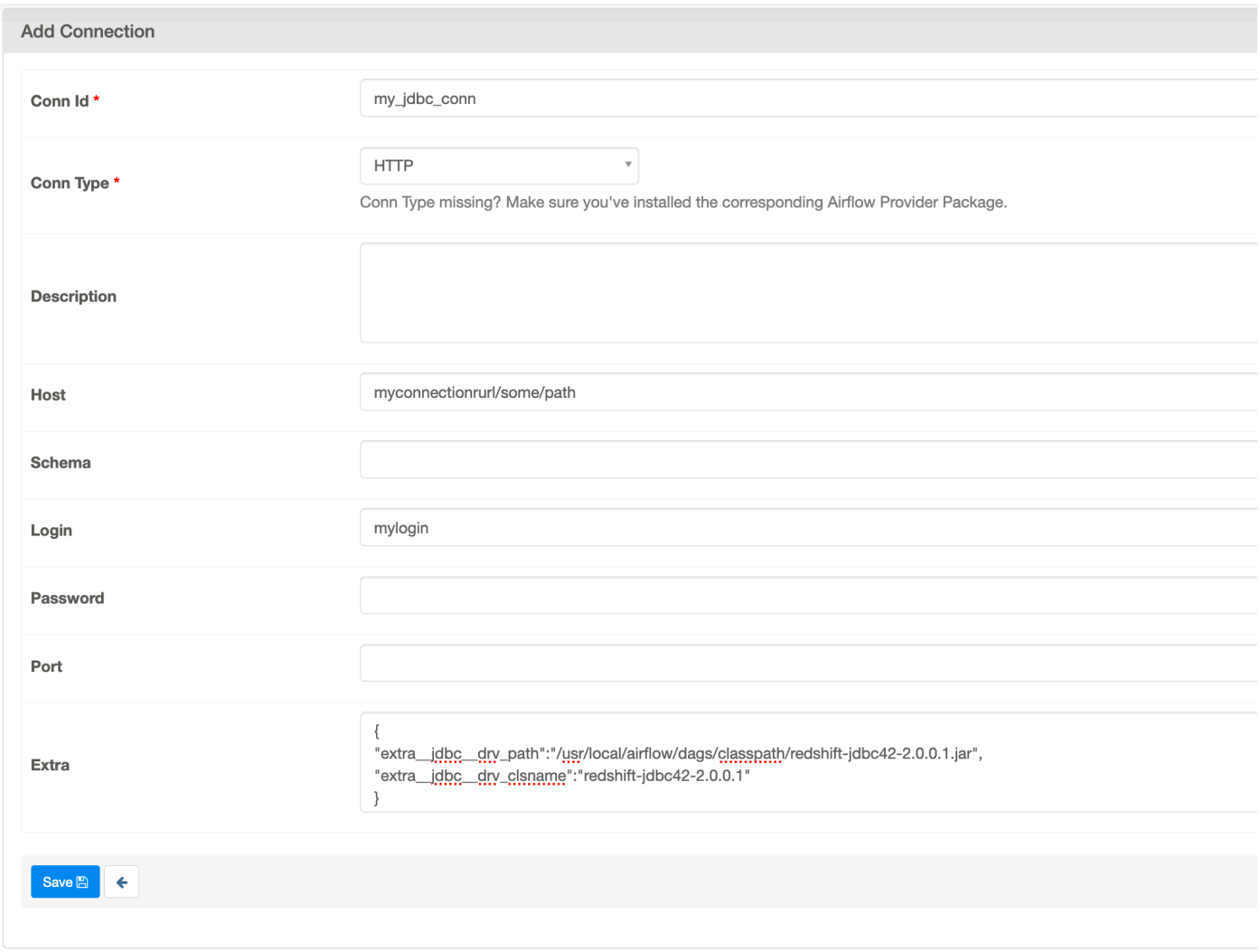
O exemplo a seguir mostra como usar o modelo de conexão HTTP para um tipo de conexão Jdbc no Apache Airflow v2.0.2 e os mesmos valores no modelo de conexão Jdbc para o Apache Airflow v1.10.12 na IU do Apache Airflow.

Apache Airflow v2

O exemplo a seguir mostra a string de conexão URI gerada pelo Apache Airflow para o exemplo desta seção.

```
http://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

O exemplo a seguir mostra como usar o modelo de conexão HTTP para uma conexão Jdbc para o Apache Airflow v2 na IU do Apache Airflow.

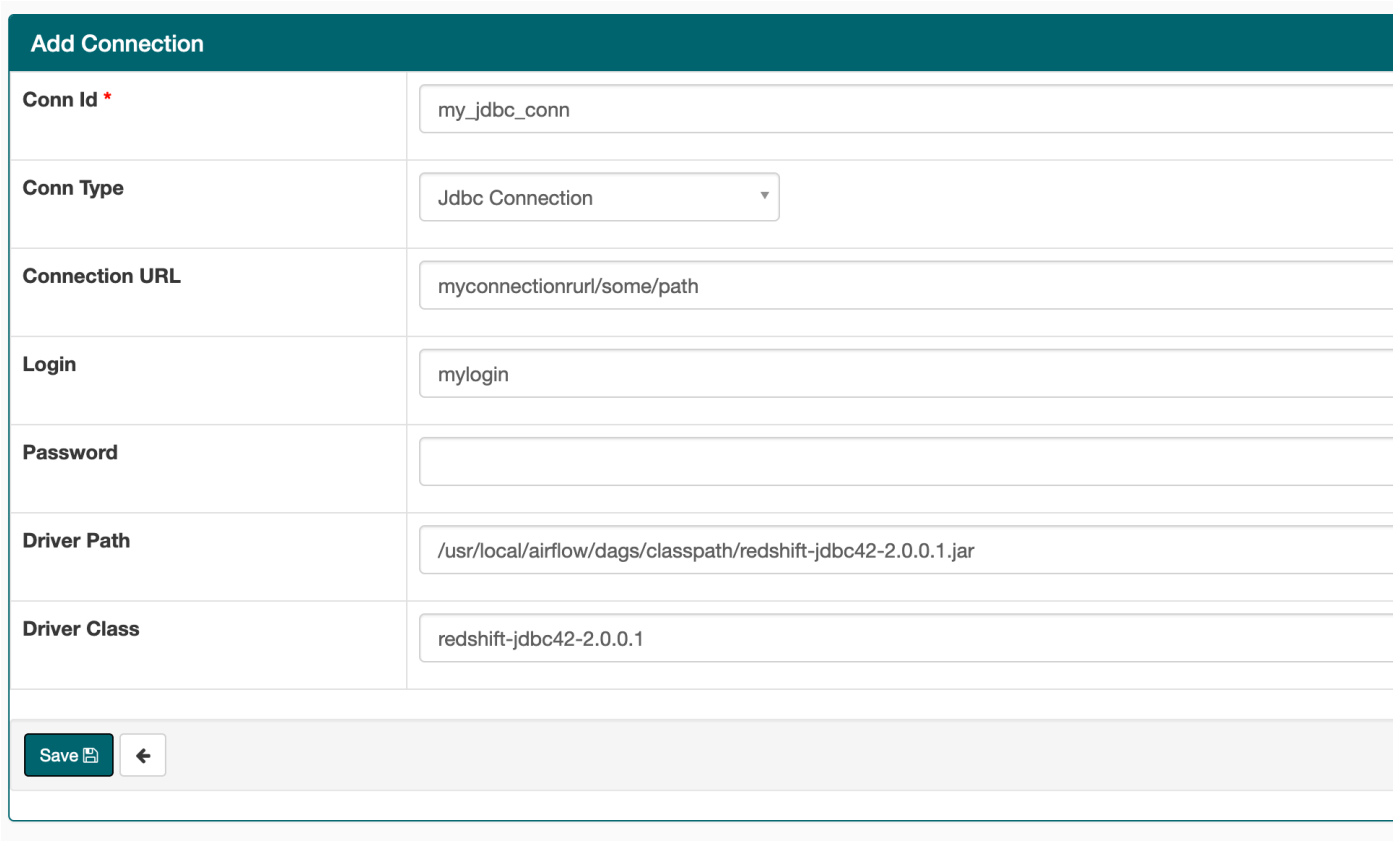


Apache Airflow v1

O exemplo a seguir mostra a string de conexão URI gerada pelo Apache Airflow para o exemplo desta seção.

```
jdbc://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

O exemplo a seguir mostra o modelo de conexão Jdbc para o Apache Airflow v1.10.12 na IU do Apache Airflow.



Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager

AWS Secrets Manager é um back-end alternativo compatível com o Apache Airflow em um ambiente Amazon Managed Workflows for Apache Airflow. Este guia mostra como usar para armazenar com segurança segredos AWS Secrets Manager para variáveis do Apache Airflow e uma conexão do Apache Airflow no Amazon Managed Workflows for Apache Airflow.

> **Note**
>
> - Você será cobrado pelos segredos que criar. Para obter informações sobre preços, consulte Preços do AWS.

Sumário

Etapa 1: forneça ao Amazon MWAA permissão para acessar as chaves secretas do Secrets Manager

O perfil de execução do seu ambiente Amazon MWAA precisa de acesso de leitura à chave secreta de entrada em AWS Secrets Manager. A política do IAM a seguir permite acesso de leitura e gravação usando a política AWS gerenciada SecretsManagerReadWrite.

Para anexar a política ao seu perfil de execução

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha sua função de execução no painel Permissões.
4. Escolha Anexar políticas.
5. Digite `SecretsManagerReadWrite` no campo de texto Políticas de filtro.
6. Escolha Anexar política.

Se você não quiser usar uma política de permissão AWS gerenciada, você pode atualizar diretamente a função de execução do seu ambiente para permitir qualquer nível de acesso aos recursos do Secrets Manager. Por exemplo, a declaração de política a seguir concede acesso de leitura a todos os segredos que você cria em uma AWS região específica no Secrets Manager.

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

Etapa 2: crie o back-end do Secrets Manager como uma opção de configuração do Apache Airflow

A seção a seguir descreve como criar uma opção de configuração do Apache Airflow no console Amazon MWAA para o back-end. AWS Secrets Manager Se você estiver usando uma definição de configuração com o mesmo nome em `airflow.cfg`, a configuração criada nas etapas a seguir terá precedência e substituirá as definições de configuração.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. Escolha Adicionar configuração personalizada no painel Opções de configuração do Airflow. Adicione os seguintes pares de chave-valor:

    a. **`secrets.backend`**: **`airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBacken`**

    b. **`secrets.backend_kwargs`**: **`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`** Configura o Apache Airflow para strings de conexão e variáveis nos caminhos `airflow/connections/*` e `airflow/variables/*`.

    É possível usar um padrão de pesquisa para reduzir o número de chamadas de API que o Amazon MWAA faz para o Secrets Manager em seu nome. Se você não especificar um

padrão de pesquisa, o Apache Airflow pesquisará todas as conexões e variáveis no back-end configurado. Ao especificar um padrão, você restringe os caminhos possíveis que o Apache Airflow procura. Isso reduz seus custos ao usar o Secrets Manager com o Amazon MWAA.

Para especificar um padrão de pesquisa, especifique os parâmetros `connections_lookup_pattern` e `variables_lookup_pattern`. Esses parâmetros aceitam uma RegEx string como entrada. Por exemplo, para procurar segredos que comecem com `test`, digite o seguinte para `secrets.backend_kwargs`:

```
{
  "connections_prefix": "airflow/connections",
  "connections_lookup_pattern": "^test",
  "variables_prefix" : "airflow/variables",
  "variables_lookup_pattern": "^test"
}
```

> **Note**
>
> Para usar `connections_lookup_pattern` e `variables_lookup_pattern`, você deve instalar a versão 7.3.0 ou superior de `apache-airflow-providers-amazon`. Para obter mais informações sobre como atualizar os pacotes do provedor para versões mais recentes, consulte the section called “Como especificar pacotes de fornecedores mais novos”.

6. Escolha Salvar.

Etapa três: gerar uma string de URI de AWS conexão do Apache Airflow

Para criar uma cadeia de caracteres de conexão, use a tecla “tab” no teclado para indentar os pares de valores-chave no objeto Connection. Também recomendamos criar uma variável para o objeto `extra` em sua sessão de shell. A seção a seguir mostra as etapas para gerar uma string de URI de conexão do Apache Airflow para um ambiente Amazon MWAA usando o Apache Airflow ou um script Python.

Apache Airflow CLI

A sessão de shell a seguir usa sua CLI local do Airflow para gerar uma string de conexão. Se você não tiver a CLI instalada, recomendamos usar o script Python.

1. Abra uma sessão de shell do Python:

```
python3
```

2. Digite o comando :

```
>>> import json
```

3. Digite o comando :

```
>>> from airflow.models.connection import Connection
```

4. Crie uma variável em sua sessão de shell para o objeto `extra`. Substitua os valores de amostra em *YOUR_EXECUTION_ROLE_ARN* pelo ARN do perfil de execução e pela região em *YOUR_REGION* (como `us-east-1`).

```
>>> extra=json.dumps({'role_arn': 'YOUR_EXECUTION_ROLE_ARN', 'region_name':
 'YOUR_REGION'})
```

5. Crie o objeto de conexão. Substitua o valor do exemplo em `myconn` pelo nome da conexão Apache Airflow.

```
>>> myconn = Connection(
```

6. Use a tecla “tab” do teclado para adicionar recuo a cada um dos seguintes pares de chave-valor em seu objeto de conexão. Substitua os valores da amostra em *vermelho*.

   a. Especifique o tipo de AWS conexão:

```
... conn_id='aws',
```

   b. Especifique a opção de banco de dados Apache Airflow:

```
... conn_type='mysql',
```

c. Especifique o URL da IU do Apache Airflow no Amazon MWAA:

```
... host='288888a0-50a0-888-9a88-1a111aaa0000.a1.us-
east-1.airflow.amazonaws.com/home',
```

d. Especifique o ID da chave de AWS acesso (nome de usuário) para fazer login no Amazon MWAA:

```
... login='YOUR_AWS_ACCESS_KEY_ID',
```

e. Especifique a chave de acesso AWS secreta (senha) para fazer login no Amazon MWAA:

```
... password='YOUR_AWS_SECRET_ACCESS_KEY',
```

f. Especifique a variável de sessão de shell `extra`:

```
... extra=extra
```

g. Feche o objeto de conexão.

```
... )
```

7. Imprima a string do URI de conexão:

```
>>> myconn.get_uri()
```

Você deve ver a string do URI de conexão na resposta:

```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
%2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

Python script

O script Python a seguir não requer a CLI do Apache Airflow.

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `mwaa_connection.py`.

```
import urllib.parse

conn_type = 'YOUR_DB_OPTION'
host = 'YOUR_MWAA_AIRFLOW_UI_URL'
port = 'YOUR_PORT'
login = 'YOUR_AWS_ACCESS_KEY_ID'
password = 'YOUR_AWS_SECRET_ACCESS_KEY'
role_arn = urllib.parse.quote_plus('YOUR_EXECUTION_ROLE_ARN')
region_name = 'YOUR_REGION'

conn_string = '{0}://{1}:{2}@{3}:{4}?
role_arn={5}&region_name={6}'.format(conn_type, login, password, host, port,
 role_arn, region_name)
print(conn_string)
```

2. Substitua os espaços reservados em *vermelho*.
3. Execute o script a seguir para gerar uma string de conexão.

```
python3 mwaa_connection.py
```

Etapa 4: adicione as variáveis no Secrets Manager

A seção a seguir descreve como criar o segredo para uma variável no Secrets Manager.

Para criar o segredo

1. Abra o console de AWS Secrets Manager.
2. Selecione Armazenar um novo segredo.
3. Selecione Outro tipo de segredo.
4. No painel Especificar os pares de chave/valor a serem armazenados nesse painel secreto, escolha Texto simples.
5. Adicione o valor da variável como Texto simples no formato a seguir.

```
"YOUR_VARIABLE_VALUE"
```

Por exemplo, para especificar um número inteiro:

```
14
```

Por exemplo, para especificar uma string:

```
"mystring"
```

6. Em Chave de criptografia, escolha uma opção de AWS KMS chave na lista suspensa.
7. Insira um nome no campo de texto para Nome secreto no formato a seguir.

```
airflow/variables/YOUR_VARIABLE_NAME
```

Por exemplo: .

```
airflow/variables/test-variable
```

8. Escolha Próximo
9. Na página Configurar segredo, no painel Nome e descrição do segredo, faça o seguinte.

   a. Em Nome secreto, forneça um nome para seu segredo.

   b. (Opcional) Em Descrição, insira uma descrição para o nome do segredo.

   Escolha Avançar.
10. Em Configurar rotação - opcional, deixe as opções padrão e escolha Avançar.
11. Repita essas etapas no Secrets Manager para quaisquer variáveis adicionais que você queira adicionar.
12. Na página Analisar, analise os detalhes do segredo e escolha Armazenar.

Etapa 5: adicione a conexão no Secrets Manager

A seção a seguir descreve como criar o segredo para o URI da string de conexão no Secrets Manager.

Para criar o segredo

1. Abra o console de AWS Secrets Manager.
2. Selecione Armazenar um novo segredo.

3. Selecione Outro tipo de segredo.
4. No painel Especificar os pares de chave/valor a serem armazenados nesse painel secreto, escolha Texto simples.
5. Adicione a string do URI de conexão como texto simples no formato a seguir.

```
YOUR_CONNECTION_URI_STRING
```

Por exemplo: .

```
mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
%2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1
```

> **Warning**
>
> O Apache Airflow analisa cada um dos valores na string de conexão. Você não deve usar aspas simples nem duplas, ou ele analisará a conexão como uma única string.

6. Em Chave de criptografia, escolha uma opção de AWS KMS chave na lista suspensa.
7. Insira um nome no campo de texto para Nome secreto no formato a seguir.

```
airflow/connections/YOUR_CONNECTION_NAME
```

Por exemplo: .

```
airflow/connections/myconn
```

8. Escolha Próximo
9. Na página Configurar segredo, no painel Nome e descrição do segredo, faça o seguinte.
   a. Em Nome secreto, forneça um nome para seu segredo.
   b. (Opcional) Em Descrição, insira uma descrição para o nome do segredo.

   Escolha Avançar.
10. Em Configurar rotação - opcional, deixe as opções padrão e escolha Avançar.

11. Repita essas etapas no Secrets Manager para quaisquer variáveis adicionais que você queira adicionar.
12. Na página Analisar, analise os detalhes do segredo e escolha Armazenar.

Código de exemplo

- Saiba como usar a chave secreta para a conexão do Apache Airflow (`myconn`) nesta página usando o código de exemplo em Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow.
- Saiba como usar a chave secreta para a variável do Apache Airflow (`test-variable`) nesta página usando o código de exemplo em Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow.

Recursos

- Para obter mais informações sobre como configurar segredos do Secrets Manager usando o console e o AWS CLI, consulte Criar um segredo no Guia do AWS Secrets Manager Usuário.
- Use um script Python para migrar um grande volume de variáveis e conexões do Apache Airflow para o Secrets Manager em Move your Apache Airflow connections and variables to AWS Secrets Manager (Mova suas conexões e variáveis do Apache Airflow para).

Próximas etapas

- Saiba como gerar um token para acessar a interface do Apache Airflow em Acessando o Apache Airflow.

Como gerenciar ambientes no Amazon MWAA

O console Amazon Managed Workflows for Apache Airflow contém opções integradas para configurar o acesso privado ou público à IU do Apache Airflow. Também contém opções integradas para configurar o tamanho do ambiente, quando escalar os operadores e opções de configuração do Apache Airflow que permitem que você substitua as configurações do Apache Airflow que normalmente só são acessíveis em `airflow.cfg`. Este guia descreve como usar essas configurações no console do Amazon MWAA.

Tópicos

Como configurar a classe de ambiente do Amazon MWAA

A classe de ambiente que você escolhe para seu ambiente Amazon MWAA determina o tamanho dos AWS Fargate contêineres AWS gerenciados nos quais o Celery Executor é executado e do banco de dados de metadados Amazon Aurora PostgreSQL gerenciado em que os AWS programadores do Apache Airflow criam instâncias de tarefas. Esta página descreve cada classe de ambiente do Amazon MWAA e as etapas para atualizar a classe de ambiente no console Amazon MWAA.

Seções

Funcionalidades do ambiente

A seção a seguir contém as tarefas simultâneas padrão do Apache Airflow, a Memória de Acesso Aleatório (RAM) e as unidades de processamento virtual centralizadas (vCPUs) para cada classe de

ambiente. As tarefas simultâneas listadas pressupõem que a simultaneidade de tarefas não exceda a capacidade de operadores do Apache Airflow no ambiente.

Na tabela a seguir, a capacidade do DAG se refere às definições do DAG, não às execuções, e pressupõe que seus DAGs sejam dinâmicos em um único arquivo Python e escritos com as melhores práticas do Apache Airflow.

As execuções de tarefas dependem de quantas são agendadas simultaneamente, pressupondo que o número de execuções do DAG definidas para iniciar ao mesmo tempo, não exceda o padrão `max_dagruns_per_loop_to_schedule`, bem como o tamanho e o número de operadores, conforme detalhado neste tópico.

mw1.small

- Capacidade de até 50 DAG
- 5 tarefas simultâneas (por padrão)
- 1 vCPUs
- RAM de 2 GB

mw1.medium

- Capacidade de até 200 DAG
- 10 tarefas simultâneas (por padrão)
- 2 vCPUs
- RAM de 4 GB

mw1.large

- Capacidade de até 1000 DAG
- 20 tarefas simultâneas (por padrão)
- 4 vCPUs
- RAM de 8 GB

mw1.xlarge

- Capacidade de até 2000 DAG

- 40 tarefas simultâneas (por padrão)
- 8 vCPUs
- 24 GB DE MEMÓRIA RAM

mw1.2xlarge

- Capacidade de até 4000 DAG
- 80 tarefas simultâneas (por padrão)
- 16 vCPUs
- 48 GB DE MEMÓRIA RAM

Você pode usar `celery.worker_autoscale` para aumentar as tarefas por operador. Para obter mais informações, consulte the section called “Exemplo de caso de uso de alto desempenho”.

Programadores do Apache Airflow

A seção a seguir contém as opções de programadores do Apache Airflow disponíveis no Amazon MWAA e como o número de programadores afeta o número de acionadores.

No Apache Airflow, um acionador gerencia tarefas que são adiadas até que certas condições, especificadas usando um acionador sejam atendidas. No Amazon MWAA, o acionador é executado com o programador na mesma tarefa do Fargate. Aumentar a contagem de programadores aumenta correspondentemente o número de acionadores disponíveis, otimizando a forma como o ambiente gerencia tarefas adiadas. Isso garante o manuseio eficiente das tarefas, programando-as prontamente para serem executadas quando as condições forem atendidas.

Apache Airflow v2

- v2: aceita de 2 a 5. Padronizado como 2.

Configurando a escalabilidade automática do funcionário do Amazon MWAA

O mecanismo de escalabilidade automática aumenta automaticamente o número de trabalhadores do Apache Airflow em resposta às tarefas em execução e em fila em seu ambiente Amazon

Managed Workflows for Apache Airflow e descarta trabalhadores extras quando não há mais tarefas na fila ou em execução. Esta página descreve como você pode configurar o auto scaling especificando o número máximo de trabalhadores do Apache Airflow que são executados em seu ambiente usando o console Amazon MWAA.

> **Note**
>
> O Amazon MWAA usa métricas do Apache Airflow para determinar quando operadores adicionais do Executor Celery são necessários e, conforme exigido, aumenta o número de operadores da Fargate até o valor especificado por `max-workers`. À medida que os trabalhadores adicionais concluem o trabalho e a carga de trabalho diminui, o Amazon MWAA os remove, reduzindo assim a escala para o valor definido por. `min-workers` Se os trabalhadores realizarem novas tarefas durante a redução da escala, o Amazon MWAA manterá o recurso Fargate e não removerá o funcionário. Para obter mais informações, consulte Como funciona o escalonamento automático do Amazon MWAA.

Seções

Como funciona o escalonamento de funcionários

O Amazon MWAA usa métricas `RunningTasks` e `QueuedTasks`, em que (tarefas em execução + tarefas em fila) / (tarefas por operador) = (operadores necessários). Se o número necessário de operadores for maior que o número atual, o Amazon MWAA adicionará contêineres de operadores Fargate a esse valor, até o valor máximo especificado em `max-workers`.

À medida que a carga de trabalho diminui `RunningTasks` e a soma `QueuedTasks` métrica diminui, a Amazon MWAA solicita à Fargate que reduza o número de trabalhadores para o meio ambiente. Todos os trabalhadores que ainda estão concluindo o trabalho permanecem protegidos durante a redução de escala até concluírem o trabalho. Dependendo da carga de trabalho, as tarefas podem ficar na fila enquanto os trabalhadores reduzem a escala.

Como usar o console do Amazon SNS

É possível escolher o número máximo de operadores que podem ser executados em seu ambiente simultaneamente no console do Amazon MWAA. Por padrão, é possível especificar um valor máximo de até 25.

Para configurar o número de operadores

1. Abra a página Environments (Ambientes) no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. No painel Classe de ambiente, insira um valor em Contagem máxima de operadores.
6. Escolha Salvar.

> **Note**
>
> Poderá levar alguns minutos para que as alterações sejam aplicadas no seu ambiente.

Exemplo de caso de uso de alto desempenho

A seção a seguir descreve os tipos de configurações que é possível usar para permitir alto desempenho e paralelismo em um ambiente.

Apache Airflow on-premises

Normalmente, em uma plataforma Apache Airflow local, você definiria as configurações de paralelismo de tarefas, escalonamento automático e simultaneidade em seu arquivo: `airflow.cfg`

- `core.parallelism`: o número máximo de instâncias de tarefas que podem ser executadas simultaneamente por agendador.
- `core.dag_concurrency`: a simultaneidade máxima para DAGs (não operadores).
- `celery.worker_autoscale`: o número máximo e mínimo de tarefas que podem ser executadas simultaneamente em qualquer operador.

Por exemplo, se `core.parallelism` estivesse definido como `100` e `core.dag_concurrency` estivesse definido como 7, você ainda só conseguiria executar um total de 14 tarefas simultaneamente se tivesse 2 DAGs. Dado isso, cada DAG está configurado para executar somente sete tarefas simultaneamente (em `core.dag_concurrency`), mesmo que o paralelismo geral esteja definido para `100` (em `core.parallelism`).

Um ambiente Amazon MWAA.

Em um ambiente Amazon MWAA, você pode definir essas configurações diretamente no console do Amazon MWAA usando Como usar opções de configuração do Apache Airflow no Amazon MWAAComo configurar a classe de ambiente do Amazon MWAA, e o mecanismo de escalabilidade automática de contagem máxima de trabalhadores. Embora não `core.dag_concurrency` esteja disponível na lista suspensa como uma opção de configuração do Apache Airflow no console Amazon MWAA, você pode adicioná-la como uma opção de configuração personalizada do Apache Airflow.

Digamos que, ao criar seu ambiente, você tenha escolhido as seguintes configurações:

1. A classe de ambiente mw1.small, que controla o número máximo de tarefas simultâneas que cada operador pode executar por padrão e a vCPU dos contêineres.
2. A configuração padrão de `10` operadores na Contagem máxima de operadores.
3. Uma opção de configuração do Apache Airflow para `celery.worker_autoscale` de `5,5` tarefas por operador.

Isso significa que é possível executar 50 tarefas simultâneas em seu ambiente. Todas as tarefas além de 50 serão colocadas na fila e aguardarão a conclusão das tarefas em execução.

Execute mais tarefas simultâneas. É possível modificar seu ambiente para executar mais tarefas simultaneamente usando as seguintes configurações:

1. Aumente o número máximo de tarefas simultâneas que cada operador pode executar por padrão e a vCPU dos contêineres escolhendo a classe de ambiente `mw1.medium` (10 tarefas simultâneas por padrão).
2. Acrescente `celery.worker_autoscale` como opção de configuração do Apache Airflow.
3. Aumente a contagem máxima de operadores. Neste exemplo, aumentar o número máximo de operadores de `10` para `20` dobraria o número de tarefas simultâneas que o ambiente pode executar.

Especifique o mínimo de operadores. Você também pode especificar o número mínimo e máximo de Apache Airflow Workers que são executados em seu ambiente usando o AWS Command Line Interface ()AWS CLI. Por exemplo: .

```
aws mwaa update-environment --max-workers 10 --min-workers 10 --
name YOUR_ENVIRONMENT_NAME
```

Para saber mais, consulte o comando update-environment em AWS CLI.

Tarefas de solução de problemas bloqueadas no estado de execução

Em casos raros, o Apache Airflow pode achar que ainda há tarefas em execução. Para resolver esse problema, você precisa limpar a tarefa perdida na sua IU do Apache Airflow. Para obter mais informações, consulte o tópico sobre solução de problemas Vejo minhas tarefas travadas ou não concluídas.

Próximas etapas

- Saiba mais sobre as melhores práticas que recomendamos para ajustar o desempenho do seu ambiente em Ajuste de desempenho para o Apache Airflow no Amazon MWAA.

Configurando a escalabilidade automática do servidor web Amazon MWAA

Para ambientes que executam o Apache Airflow Apache Airflow v2.2.2 e superior, o Amazon MWAA escala dinamicamente seus servidores web para lidar com cargas de trabalho flutuantes, o que, por sua vez, evita problemas de desempenho durante picos de carga. Ao escalar automaticamente o número de servidores web com base na utilização da CPU e na contagem ativa de conexões, o Amazon MWAA garante que seu ambiente Apache Airflow possa acomodar perfeitamente o aumento da demanda, seja de solicitações de API REST, uso de CLI ou mais usuários simultâneos da interface de usuário do Apache Airflow.

Seções

- Como funciona o escalonamento do servidor web
- Como usar o console do Amazon SNS

Como funciona o escalonamento do servidor web

O Amazon MWAA usa a métrica do contêiner e a métrica do balanceador de carga para determinar se a escalabilidade dos servidores web é necessária com base na quantidade de tráfego. `CPUUtilization` `ActiveConnectionCount` Se `CPUUtilization` for maior que 70 ou `ActiveConnectionCount` maior que 15, o Amazon MWAA adicionará mais contêineres de servidor web Fargate até o valor máximo especificado por. `MaxWebservers`

À medida que o tráfego diminui e os `ActiveConnectionCount` valores `CPUUtilization` e diminuem, o Amazon MWAA solicita à Fargate que reduza os contêineres do servidor web do ambiente até o valor mínimo definido por. `MinimumWebservers`

Como usar o console do Amazon SNS

Você pode escolher o número de servidores web que podem ser executados em seu ambiente simultaneamente no console do Amazon MWAA. Por padrão, o número mínimo de servidores da Web é dois e o número máximo de servidores da Web é cinco.

Para configurar o número de servidores web

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. No painel Classe de ambiente, insira um valor em Contagem máxima de servidores web.
6. Em seguida, insira um valor em Contagem mínima de servidores web.
7. Escolha Salvar.

> **Note**
>
> Poderá levar alguns minutos para que as alterações sejam aplicadas no seu ambiente.

Como usar opções de configuração do Apache Airflow no Amazon MWAA

As opções de configuração do Apache Airflow podem ser anexadas ao seu ambiente Amazon Managed Workflows for Apache Airflow como variáveis de ambiente. É possível escolher na lista suspensa sugerida ou especificar opções de configuração personalizadas para sua versão do Apache Airflow no console Amazon MWAA. Esta página descreve as opções de configuração do Apache Airflow disponíveis e como usar essas opções para substituir as configurações do Apache Airflow em seu ambiente.

Sumário

Pré-requisitos

Você precisará do seguinte antes de concluir as etapas nesta página.

- Permissões — Seu AWS administrador deve ter concedido acesso à política de controle de FullConsoleacesso do AmazonMWAA para seu ambiente. Além disso, seu ambiente Amazon MWAA deve ser autorizado pela sua função de execução para acessar os AWS recursos usados pelo seu ambiente.
- Acesso: se você precisar de acesso a repositórios públicos para instalar dependências diretamente no servidor web, seu ambiente deverá ser configurado com acesso ao servidor web de rede pública. Para ter mais informações, consulte the section called “Modos de acesso do Apache Airflow”.
- Configuração do Amazon S3: o bucket do Amazon S3 usado para armazenar seus DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` deve ser configurado com Acesso público bloqueado e Versionamento habilitado.

Como funciona

Quando você cria um ambiente, o Amazon MWAA anexa as definições de configuração que você especifica no console do Amazon MWAA nas opções de configuração do Airflow como variáveis de ambiente ao contêiner do seu ambiente. AWS Fargate Se você estiver usando uma configuração com o mesmo nome em `airflow.cfg`, as opções especificadas no console do Amazon MWAA substituirão os valores em `airflow.cfg`.

Embora não exponhamos o `airflow.cfg` na interface do usuário do Apache Airflow de um ambiente Amazon MWAA por padrão, você pode alterar as opções de configuração do Apache Airflow diretamente no console do Amazon MWAA, incluindo a configuração para expor as configurações. `webserver.expose_config`

Como usar opções de configuração para fazer upload de plug-ins no Apache Airflow v2

Por padrão, no Apache Airflow v2, os plug-ins são configurados para terem o upload sendo feito “lentamente” usando a configuração `core.lazy_load_plugins : True`. Se você estiver usando plug-ins personalizados no Apache Airflow v2, deverá adicionar `core.lazy_load_plugins : False` uma opção de configuração do Apache Airflow para fazer upload de plug-ins no início de cada processo do Airflow para substituir a configuração padrão.

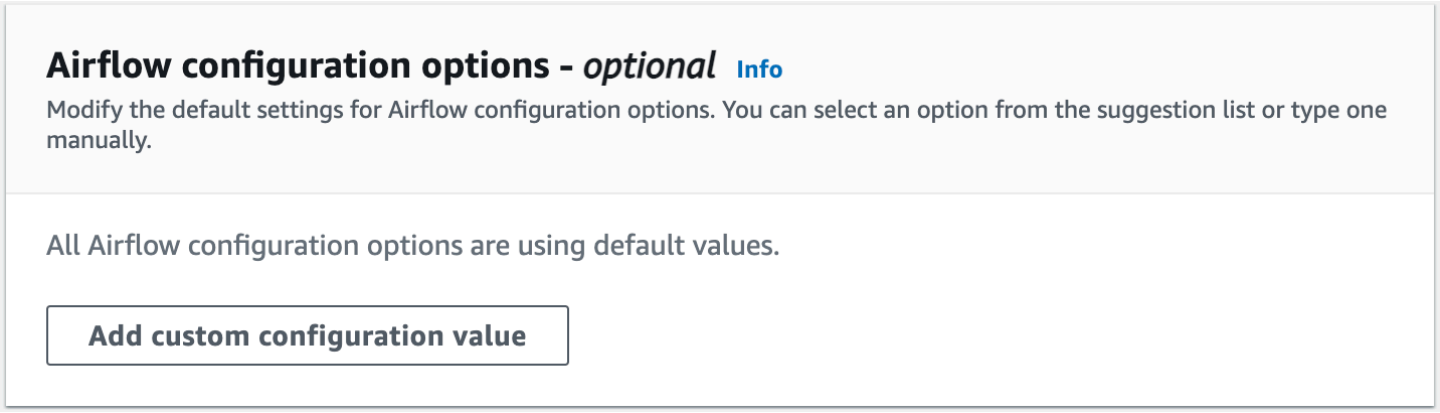
Visão geral das opções de configuração

Ao adicionar uma configuração no console do Amazon MWAA, o Amazon MWAA grava a configuração como uma variável de ambiente.

- Opções listadas. É possível escolher uma das configurações disponíveis para sua versão do Apache Airflow na lista suspensa. Por exemplo, `dag_concurrency` : `16`. A configuração é traduzida para o contêiner Fargate do seu ambiente como `AIRFLOW__CORE__DAG_CONCURRENCY : 16`
- Opções personalizadas. Também é possível especificar as opções de configuração do Airflow que não estão listadas para sua versão do Apache Airflow na lista suspensa. Por exemplo, `foo.user` : `YOUR_USER_NAME`. A configuração é traduzida para o contêiner Fargate do seu ambiente como `AIRFLOW__FOO__USER : YOUR_USER_NAME`

Opções de configuração do Apache Airflow

A imagem a seguir mostra onde é possível personalizar as opções de configuração do Apache Airflow no console do Amazon MWAA.



Referência do Apache Airflow

Para obter uma lista das opções de configuração que tem suporte pelo Apache Airflow, consulte Referência de configuração no Guia de referência do Apache Airflow. Para ver as opções da versão do Apache Airflow que você está executando no Amazon MWAA, selecione a versão na lista suspensa.

Como usar o console do Amazon SNS

O procedimento a seguir descreve as etapas para adicionar uma opção de configuração do Airflow ao seu ambiente.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. Escolha Adicionar configuração personalizada no painel Opções de configuração do Airflow.
6. Escolha uma configuração na lista suspensa e insira um valor, ou digite uma configuração personalizada e insira um valor.
7. Escolha Adicionar configuração personalizada para cada configuração que você deseja adicionar.
8. Escolha Salvar.

Referência da configuração

A seção a seguir contém a lista de configurações disponíveis do Apache Airflow na lista suspensa no console do Amazon MWAA.

Configurações de e-mail

A lista a seguir mostra as opções de configuração de notificação por e-mail do Airflow disponíveis no Amazon MWAA.

Recomendamos usar a porta 587 para tráfego SMTP. Por padrão, AWS bloqueia o tráfego SMTP de saída na porta 25 de todas as instâncias do Amazon EC2. Se precisar enviar tráfego de saída na porta 25, é possível solicitar que essa restrição seja removida.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2	email.email_backend	O utilitário do Apache Airflow usado para	airflow.utils.email.send_email_smtp

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
		notificações por e-mail em email_backend.	
v2	smtp.smtp_host	O nome do servidor de saída usado para o endereço de e-mail em smtp_host.	localhost
v2	smtp.smtp_starttls	O Transport Layer Security (TLS) é usado para criptografar o e-mail pela Internet em smtp_starttls.	Falso
v2	smtp.smtp_ssl	O Secure Sockets Layer (SSL) é usado para conectar o servidor e o e-mail de cliente em smtp_ssl.	Verdadeiro
v2	smtp.smtp_port	A porta Transmission Control Protocol (TCP) designada para o servidor em smtp_port.	587
v2	smtp.smtp_mail_from	O endereço de e-mail de saída em smtp_mail_from.	myemail@domain.com

Configurações da tarefa

A lista a seguir mostra as configurações disponíveis na lista suspensa para tarefas do Airflow no Amazon MWAA.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2	core.default_task_retries	O número de vezes para repetir uma tarefa do Apache Airflow em default_task_retries.	3
v2	core.parallelism	O número máximo de instâncias de tarefas que podem ser executadas simultaneamente em todo o ambiente em paralelo (paralelismo).	40

Configurações de scheduler

A lista a seguir mostra as configurações do scheduler Apache Airflow disponíveis na lista suspensa do Amazon MWAA.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2	scheduler.catchup_by_default	Solicita ao scheduler que crie uma	Falso

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
		execução do DAG para “acompanhar” o intervalo de tempo específico em catchup_by_default.	
v2	scheduler.scheduler_zombie_task_threshold	Informa ao agendador se a instância da tarefa deve ser marcada como falha e reprogramar a tarefa em scheduler_zombie_task_threshold.	300

Configurações do operador

A lista a seguir mostra as configurações de operadores do Airflow disponíveis na lista suspensa do Amazon MWAA.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2	celery.worker_autoscale	O número máximo e mínimo de tarefas que podem ser executadas simultaneamente em qualquer operador usando o Executor Celery em worker_au	16,12

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
		toscale. O valor deve ser separado por vírgula na seguinte ordem: `max_concurrency,min_concurrency` .	

Configurações do servidor da Web

A lista a seguir mostra as configurações de servidores Web do Airflow disponíveis na lista suspensa do Amazon MWAA.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2	webserver.default_ui_timezone	A configuração padrão de data e hora do IU do Apache Airflow em default_ui_timezone. **Note** Definir a opção `default_ui_timezone` não altera o fuso horário no qual seus DAGs estão	America/New_York

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
		programados para serem executados. Para alterar o fuso horário dos DAGs, é possível usar um plug-in personalizado. Para ter mais informações, consulte the section called “Como alterar o fuso horário de um DAG”.	

Configurações do acionador

A lista a seguir mostra as configurações do acionador do Apache Airflow disponíveis no Amazon MWAA.

Apache Airflow v2

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
v2.7	mwaa.triggerer_enabled	Usado para ativar e desativar o acionador no Amazon MWAA. Por padrão, esse	Verdadeiro

Versão Airflow	Opção de configuração do Airflow	Descrição	Valor de exemplo
		valor é definido como `True`. Se definido como `False`, o Amazon MWAA não iniciará nenhum processo acionador nos agendadores.	
v2.7	triggerer.default_capacity	Define o número de acionadores que cada acionador pode executar em paralelo. No Amazon MWAA, essa capacidade é definida por cada acionador e por cada agendador, pois os dois componentes são executados lado a lado. O padrão por agendador é definido como`60`,, `125` `250`500, e `1000` para instâncias pequenas, médias e grandes, xlarge e 2xlarge, respectivamente.	125

Exemplos e código de exemplo

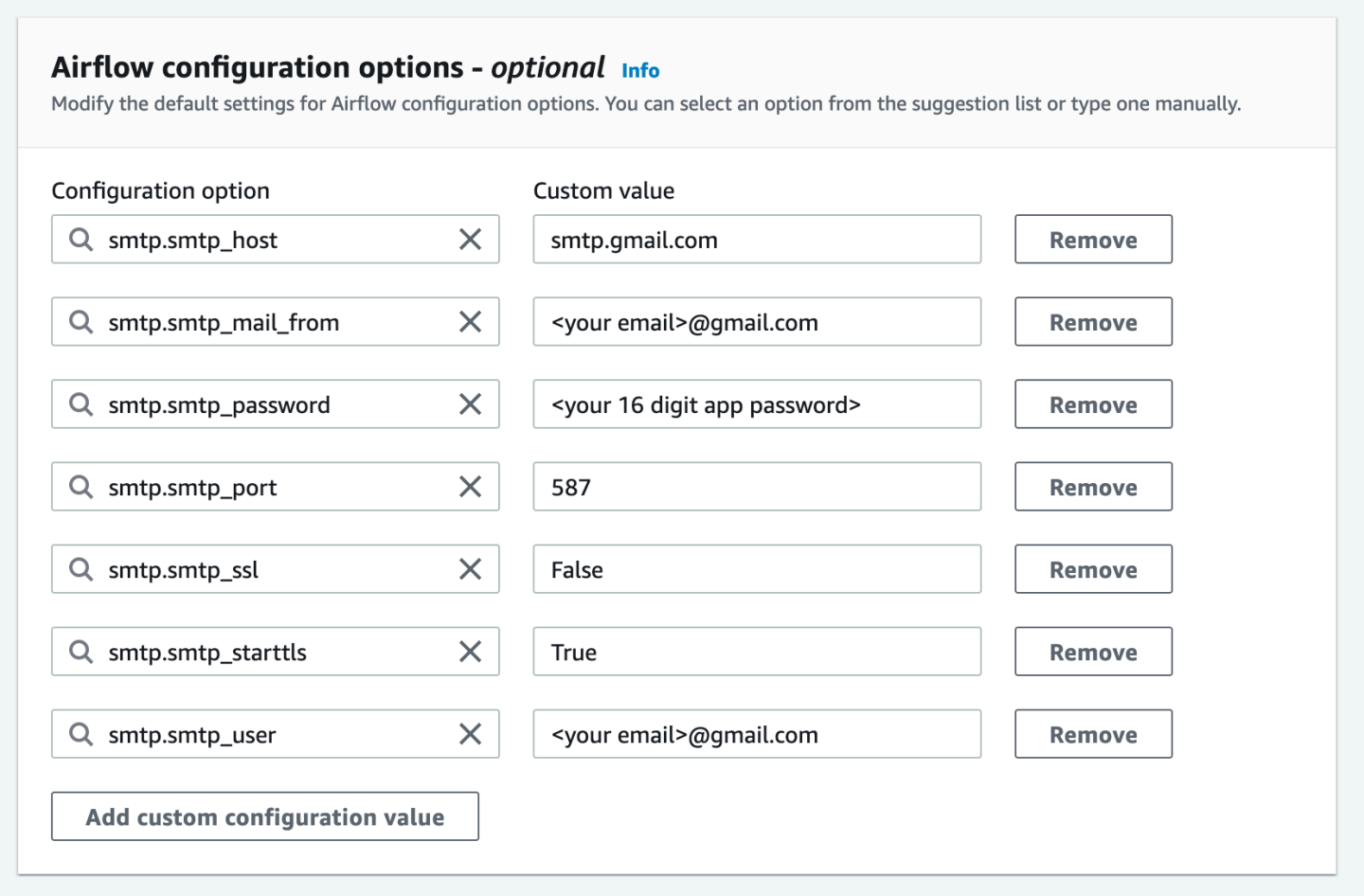
Exemplo de DAG

É possível usar o seguinte DAG para imprimir suas opções de configuração do Apache Airflow `email_backend`. Para executar em resposta aos eventos do Amazon MWAA, copie o código para a pasta DAGs do seu ambiente no seu bucket de armazenamento do Amazon S3.

```
from airflow.decorators import dag
from datetime import datetime

def print_var(**kwargs):
  email_backend = kwargs['conf'].get(section='email', key='email_backend')
  print("email_backend")
  return email_backend

@dag(
  dag_id="print_env_variable_example",
  schedule_interval=None,
  start_date=datetime(yyyy, m, d),
  catchup=False,
)
def print_variable_dag():
  email_backend_test = PythonOperator(
  task_id="email_backend_test",
  python_callable=print_var,
  provide_context=True
)

print_variable_test = print_variable_dag()
```

Exemplo de configurações de notificação por e-mail

As seguintes opções de configuração do Apache Airflow podem ser usadas para uma conta de e-mail do Gmail.com usando uma senha de aplicativo. Para obter mais informações, consulte Fazer login usando senhas de aplicativos no guia de referência da Ajuda do Gmail.



Próximas etapas

- Aprenda a fazer upload de sua pasta do DAG para seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Atualizando a versão do Apache Airflow

O Amazon MWAA é compatível com atualizações de versões anteriores. Isso significa que você pode atualizar seu ambiente de versão x**.4.**z para x**.5.**z. Para realizar uma atualização de versão principal, por exemplo, da versão **1.**y**.**z para **2.**y**.**z, você deve criar um novo ambiente e migrar seus recursos. Para obter mais informações sobre a atualização para uma nova versão principal do Apache Airflow, consulte Migração para um novo ambiente do Amazon MWAA no Guia de migração do Amazon MWAA.

Durante o processo de atualização, o Amazon MWAA captura um snapshot dos metadados do seu ambiente, atualiza os operadores, os programadores e o servidor web para a nova versão do Apache Airflow e, finalmente, restaura o banco de dados de metadados usando o snapshot.

> **Note**
>
> Você não pode fazer regredir a versão do Apache Airflow para seu ambiente.

Antes de atualizar, certifique-se de que seus DAGs e outros recursos de fluxo de trabalho são compatíveis com a nova versão do Apache Airflow para a qual você está fazendo a atualização. Se você usa um `requirements.txt` para gerenciar dependências, também deve garantir que as dependências especificadas em seus requisitos sejam compatíveis com a nova versão.

Tópicos

Atualize seus recursos de fluxo de trabalho

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

> **Warning**
>
> Especificar requisitos que são incompatíveis com sua versão de destino do Apache Airflow durante uma atualização pode resultar em um longo processo de reversão para a versão anterior do Apache Airflow com a versão de requisitos anterior.

Para migrar seus recursos de fluxo de trabalho

1. Crie uma bifurcação do repositório aws-mwaa-local-runner e clone uma cópia do executor local Amazon MWAA.
2. Acesse a ramificação do repositório aws-mwaa-local-runner que corresponda à versão para a qual você está atualizando.

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 repositório GitHub.
4. Para atualizar seu `requirements.txt`, siga as melhores práticas que recomendamos em Gerenciar dependências do Python, no Guia do usuário do Amazon MWAA.
5. (Opcional) Para acelerar o processo de atualização, limpe o banco de dados de metadados do ambiente. Ambientes com uma grande quantidade de metadados podem levar muito mais tempo para serem atualizados.
6. Depois de testar com sucesso seus recursos de fluxo de trabalho, copie seus DAGs, `requirements.txt` e plug-ins para o bucket DO Amazon S3 do seu ambiente.

Agora que você se preparou para editar o ambiente, especifique uma nova versão do Apache Airflow e inicie o procedimento de atualização.

Especifique a nova versão

Depois de concluir a atualização dos recursos do fluxo de trabalho para garantir a compatibilidade com a nova versão do Apache Airflow, faça o que se segue para editar os detalhes do ambiente e especificar a versão do Apache Airflow para a qual você deseja atualizar.

> **Note**
>
> Quando você executa uma atualização, todas as tarefas atualmente em execução no ambiente são encerradas durante o procedimento. O procedimento de atualização pode levar até duas horas, período durante o qual seu ambiente ficará indisponível.

Para especificar uma nova versão usando o console

1. Abra a página Ambientes no console do Amazon MWAA.
2. Na lista de Ambientes, escolha o ambiente que você deseja atualizar.
3. Na página do ambiente, escolha Editar para editar o ambiente.
4. Na seção Detalhes do ambiente, para a versão do Airflow, escolha o novo número da versão do Apache Airflow para o qual você deseja atualizar o ambiente na lista suspensa.
5. Escolha Avançar até que você esteja na página Revisar e salvar.

6. Na página Revisar e salvar, revise o resumo da política e depois escolha Salvar alterações para salvar seu trabalho.

Ao aplicar as alterações, seu ambiente inicia o procedimento de atualização. Durante esse período, o status do seu ambiente indica quais ações o Amazon MWAA está tomando e se o procedimento foi bem-sucedido.

Em um cenário de upgrade bem-sucedido, o status mostrará `UPDATING`, e depois `CREATING_SNAPSHOT` enquanto o Amazon MWAA captura um backup dos seus metadados. Por fim, o status retornará primeiro para `UPDATING` e depois para `AVAILABLE` quando o procedimento for concluído.

Se o ambiente falhar na atualização, o status do seu ambiente mostrará `ROLLING_BACK`. Se a reversão for bem-sucedida, o status mostrará primeiro `UPDATE_FAILED`, indicando que a atualização falhou, mas o ambiente está disponível. Se a reversão falhar, o status mostrará `UNAVAILABLE`, indicando que você não pode acessar o ambiente.

Como usar um script de startup com o Amazon MWAA

Um script de startup é um script de shell (`.sh`) que você hospeda no bucket do Amazon S3 do seu ambiente, semelhante aos seus DAGs, requisitos e plug-ins. O Amazon MWAA executa esse script durante o startup em cada componente individual do Apache Airflow (operador, programador e servidor web) antes de instalar os requisitos e inicializar o processo do Apache Airflow. Use um script de startup para:

- Instalar os runtimes: instale runtimes do Linux exigidos por seus fluxos de trabalho e conexões.
- Configurar variáveis de ambiente: defina variáveis de ambiente para cada componente do Apache Airflow. Substitua variáveis comuns como `PATH`, `PYTHONPATH` e `LD_LIBRARY_PATH`.
- Gerencie chaves e tokens: transmita tokens de acesso para repositórios personalizados para `requirements.txt` e configure chaves de segurança.

Os tópicos a seguir descrevem como configurar um script de startup para instalar runtimes do Linux, definir variáveis de ambiente e solucionar problemas relacionados usando o CloudWatch Logs.

Tópicos

- Configurar um script de startup

- Instale os runtimes do Linux usando um script de startup
- Definir variáveis de ambiente usando um script de startup

Configurar um script de startup

Para usar um script de startup com seu ambiente Amazon MWAA existente, faça upload de um arquivo `.sh` para o bucket Amazon S3 do seu ambiente. Em seguida, para associar o script ao ambiente, especifique o seguinte nos detalhes do seu ambiente:

- O caminho do URL do Amazon S3 para o script: o caminho relativo para o script hospedado em seu bucket, por exemplo, `s3://mwaa-environment/startup.sh`
- O ID da versão do Amazon S3 do script: a versão do script de startup de shell em seu bucket do Amazon S3. Você deve especificar o ID da versão que o Amazon S3 atribui ao arquivo toda vez que você atualiza o script. Os IDs de versão são strings opacas Unicode, com codificação UTF-8 e prontas para URL que não têm mais de 1.024 bytes de comprimento, por exemplo `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`.

Para concluir as etapas desta seção, use o seguinte exemplo de script. O script gera o valor atribuído a `MWAA_AIRFLOW_COMPONENT`. Essa variável de ambiente identifica cada componente do Apache Airflow no qual o script é executado.

Copie o código e salve-o localmente como `startup.sh`.

```
#!/bin/sh

echo "Printing Apache Airflow component"
echo $MWAA_AIRFLOW_COMPONENT
```

Em seguida, faça upload do script para o seu bucket do Amazon S3.

AWS Management Console

Para fazer o upload um script de shell (console)

1. Faça login no AWS Management Console e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na lista Buckets, escolha o nome do bucket associado ao ambiente.

3. Na guia Objects (Objetos), escolha Upload (Fazer upload).
4. Na página Upload, arraste e solte o script de shell que você criou.
5. Escolha Upload (Carregar).

O script aparece na lista de objetos. O Amazon S3 cria um novo ID de versão do arquivo. Se você atualizar o script e fizer o upload novamente usando o mesmo nome de arquivo, uma nova ID de versão será atribuída ao arquivo.

AWS CLI

Para criar e fazer o upload um script de shell (CLI)

1. Abra um novo prompt de comando e execute o comando `ls` do Amazon S3 para listar e identificar o bucket associado ao seu ambiente.

```
$ aws s3 ls
```

2. Navegue até a pasta em que você salvou o script de shell. Use `cp` em uma nova janela de prompt para fazer o upload do script em seu bucket. Substitua *`your-s3-bucket`* por suas informações.

```
$ aws s3 cp startup.sh s3://your-s3-bucket/startup.sh
```

Se tiver êxito, o Amazon S3 envia o caminho da URL para o objeto:

```
upload: ./startup.sh to s3://your-s3-bucket/startup.sh
```

3. Use o comando a seguir para recuperar o ID da versão mais recente do script.

```
$ aws s3api list-object-versions --bucket your-s3-bucket --prefix startup --
query 'Versions[?IsLatest].[VersionId]' --output text
```

```
BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Você especifica esse ID de versão ao associar o script a um ambiente.

Agora, associe o script ao seu ambiente.

AWS Management Console

Para associar o script a um ambiente (console)

1. Abra a página Ambientes no console do Amazon MWAA.
2. Selecione a linha do ambiente que você deseja atualizar e escolha Editar.
3. Na página Especificar detalhes, em Arquivo de script de inicialização - opcional, insira a URL do Amazon S3 para o script, por exemplo:. `s3://your-mwaa-bucket/startup-sh.`
4. Escolha a versão mais recente na lista suspensa ou navegue por S3 para encontrar o script.
5. Escolha Avançar para abrir a página Revisar e salvar.
6. Revise as alterações e escolha Salvar.

As atualizações do ambiente podem levar de 10 a 30 minutos. O Amazon MWAA executa o script de startup à medida que cada componente do seu ambiente é reiniciado.

AWS CLI

Para associar o script a um ambiente (CLI)

- Abra um prompt de comando e use `update-environment` para especificar a URL e o ID da versão do Amazon S3 para o script.

```
$ aws mwaa update-environment \
    --name your-mwaa-environment \
    --startup-script-s3-path startup.sh \
    --startup-script-s3-object-version BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Se for bem-sucedido, o Amazon MWAA retornará o nome do recurso da Amazon (ARN) do ambiente:

```
arn:aws::airflow:us-west-2:123456789012:environment/your-mwaa-environment
```

A atualização do ambiente pode levar de 10 a 30 minutos. O Amazon MWAA executa o script de startup à medida que cada componente do seu ambiente é reiniciado.

Por fim, recupere os eventos de log para verificar se o script está funcionando conforme o esperado. Quando você ativa o registro em log para cada componente do Apache Airflow, o Amazon MWAA

cria um novo grupo de logs e um novo fluxo de logs. Para obter mais informações, consulte Tipos de logs do Apache Airflow.

AWS Management Console

Para verificar o fluxo de logs do Apache Airflow (console)

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha seu ambiente.
3. No painel Monitoramento, escolha o grupo de logs para o qual você deseja visualizar os logs, por exemplo, grupo de logs do programador do Airflow.
4. No console do CloudWatch, na lista fluxo de logs, escolha um fluxo com o seguinte prefixo: `startup_script_exection_ip`.
5. No painel Eventos de logs, você verá a saída do comando imprimindo o valor de `MWAA_AIRFLOW_COMPONENT`. Por exemplo, para logs do programador, você fará o seguinte:

```
Printing Apache Airflow component
scheduler
Finished running startup script. Execution time: 0.004s.
Running verification
Verification completed
```

Você pode repetir as etapas anteriores para visualizar os logs do operador e do servidor web.

Instale os runtimes do Linux usando um script de startup

Use um script de startup para atualizar o sistema operacional de um componente do Apache Airflow e instale bibliotecas de runtime adicionais para usar com seus fluxos de trabalho. Por exemplo, o script a seguir executa `yum update` para atualizar o sistema operacional.

Ao executar `yum update` em um script de startup, você deve excluir o Python usando `--exclude=python*`, conforme mostrado no exemplo. Para que seu ambiente seja executado, o Amazon MWAA instala uma versão específica do Python compatível com seu ambiente. Portanto, não é possível atualizar a versão do Python do ambiente usando um script de startup.

```
#!/bin/sh
```

```
echo "Updating operating system"
sudo yum update -y --exclude=python*
```

Para instalar runtimes em um componente específico do Apache Airflow, use instruções condicionais `MWAA_AIRFLOW_COMPONENT`, `if` e `fi`. Este exemplo executa um único comando para instalar a biblioteca `libaio` no programador e no operador, mas não no servidor web.

> **Important**
>
> - Se você configurou um servidor web privado, deve usar a condição a seguir ou fornecer todos os arquivos de instalação localmente para evitar atingir os tempos limite da instalação.
> - Use `sudo` para executar operações que exigem privilégios administrativos.

```
#!/bin/sh

if [[ "${MWAA_AIRFLOW_COMPONENT}" != "webserver" ]]
then
     sudo yum -y install libaio
fi
```

Você pode usar um script de startup para verificar a versão do Python.

```
#!/bin/sh

export PYTHON_VERSION_CHECK=`python -c 'import sys; version=sys.version_info[:3];
 print("{0}.{1}.{2}".format(*version))'`
echo "Python version is $PYTHON_VERSION_CHECK"
```

O Amazon MWAA não oferece suporta à substituição da versão padrão do Python, pois isso pode levar a incompatibilidades com as bibliotecas instaladas do Apache Airflow.

Definir variáveis de ambiente usando um script de startup

Use scripts de startup para definir variáveis de ambiente e modificar as configurações do Apache Airflow. O exemplo a seguir define uma nova variável, `ENVIRONMENT_STAGE`. Você pode referenciar essa variável em um DAG ou em seus módulos personalizados.

```
#!/bin/sh

export ENVIRONMENT_STAGE="development"
echo "$ENVIRONMENT_STAGE"
```

Use scripts de startup para sobrescrever variáveis comuns do Apache Airflow ou do sistema. Por exemplo, você configura `LD_LIBRARY_PATH` para instruir o Python a procurar binários no caminho especificado. Isso permite que você forneça binários personalizados para seus fluxos de trabalho usando plug-ins:

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/local/airflow/plugins/your-custom-binary
```

Variáveis de ambiente reservadas

O Amazon MWAA reserva um conjunto de importantes variáveis de ambientes. Se você sobrescrever uma variável reservada, o Amazon MWAA a restaurará ao padrão. A seguir, listamos as variáveis reservadas:

- `MWAA__AIRFLOW__COMPONENT`: usada para identificar o componente Apache Airflow com um dos seguintes valores: `scheduler`, `worker` ou `webserver`.
- `AIRFLOW__WEBSERVER__SECRET_KEY`: a chave secreta usada para assinar com segurança os cookies de sessão no servidor web Apache Airflow.
- `AIRFLOW__CORE__FERNET_KEY`: a chave usada para criptografia e decodificação de dados confidenciais armazenados no banco de dados de metadados, por exemplo, senhas de conexão.
- `AIRFLOW_HOME`: o caminho para o diretório inicial do Apache Airflow, onde os arquivos de configuração e os arquivos DAG são armazenados localmente.
- `AIRFLOW__CELERY__BROKER_URL`: a URL do agente de mensagens usada para comunicação entre o agendador do Apache Airflow e os nó de processamento do Celery.
- `AIRFLOW__CELERY__RESULT_BACKEND`: a URL do banco de dados usada para armazenar os resultados das tarefas do Celery.
- `AIRFLOW__CORE__EXECUTOR`: a classe executora que o Apache Airflow deve usar. No Amazon MWAA, isso é um `CeleryExecutor`
- `AIRFLOW__CORE__LOAD_EXAMPLES`: usada para ativar ou desativar o carregamento de exemplos de DAGs.

- AIRFLOW__METRICS__METRICS_BLOCK_LIST: usada para gerenciar quais métricas do Apache Airflow são emitidas e capturadas pelo Amazon MWAA no CloudWatch.
- SQL_ALCHEMY_CONN: a string de conexão do banco de dados RDS for PostgreSQL usada para armazenar metadados do Apache Airflow no Amazon MWAA.
- AIRFLOW__CORE__SQL_ALCHEMY_CONN: usada para a mesma finalidade de SQL_ALCHEMY_CONN, mas seguindo a nova convenção de nomenclatura do Apache Airflow.
- AIRFLOW__CELERY__DEFAULT_QUEUE: a fila padrão para tarefas do Celery no Apache Airflow.
- AIRFLOW__OPERATORS__DEFAULT_QUEUE: a fila padrão para tarefas usando operadores específicos do Apache Airflow.
- AIRFLOW_VERSION: a versão Apache Airflow instalada no ambiente Amazon MWAA.
- AIRFLOW_CONN_AWS_DEFAULT: as credenciais AWS padrão usadas para integração com outros serviços da AWS.
- AWS_DEFAULT_REGION: define a Região padrão AWS usada com as credenciais padrão para integração com outros serviços na AWS.
- AWS_REGION: se definida, essa variável de ambiente substituirá os valores na variável de ambiente AWS_DEFAULT_REGION e na região da configuração de perfil.
- PYTHONUNBUFFERED: usada para enviar os fluxos `stdout` e `stderr` para logs de contêiner.
- AIRFLOW__METRICS__STATSD_ALLOW_LIST: usada para configurar uma lista de permissões de prefixos separados por vírgula para enviar as métricas que começam com os elementos da lista.
- AIRFLOW__METRICS__STATSD_ON: ativa o envio de métricas para StatsD.
- AIRFLOW__METRICS__STATSD_HOST: usada para se conectar ao daemon StatSD.
- AIRFLOW__METRICS__STATSD_PORT: usada para se conectar ao daemon StatSD.
- AIRFLOW__METRICS__STATSD_PREFIX: usada para se conectar ao daemon StatSD.
- AIRFLOW__CELERY__WORKER_AUTOSCALE: define a simultaneidade máxima e mínima.
- AIRFLOW__CORE__DAG_CONCURRENCY: define o número de instâncias de tarefas que podem ser executadas simultaneamente pelo programador em um DAG.
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG: define o número máximo de tarefas ativas por DAG.
- AIRFLOW__CORE__PARALLELISM: define o número máximo de instâncias de tarefas que podem ser executadas simultaneamente.
- AIRFLOW__SCHEDULER__PARSING_PROCESSES: define o número máximo de processos analisados pelo programador para agendar DAGs.

- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT: define o número de segundos que um operador espera para reconhecer a tarefa antes que a mensagem seja reenviada a outro operador.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION: define a Região AWS para o transporte subjacente Celery.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES: define a fila para o transporte subjacente do Celery.
- AIRFLOW_SCHEDULER_ALLOWED_RUN_ID_PATTERN: usado para verificar a validade de sua entrada para o parâmetro `run_id` ao acionar um DAG.
- AIRFLOW__WEBSERVER__BASE_URL: a URL do servidor web usada para fazer o host da IU do Apache Airflow.

Variáveis de ambiente não reservadas

Você pode usar um script de startup para substituir variáveis de ambiente não reservadas. A lista a seguir fornece algumas dessas variáveis comuns:

- PATH: especifica uma lista de diretórios em que o sistema operacional pesquisa arquivos e scripts executáveis. Quando um comando é executado na linha de comando, o sistema verifica os diretórios em PATH para encontrar e executar o comando. Ao criar operadores ou tarefas personalizadas no Apache Airflow, talvez seja necessário confiar em scripts ou executáveis externos. Se os diretórios que contêm esses arquivos não estiverem no especificado na variável PATH, as tarefas não serão executadas quando o sistema não conseguir localizá-las. Ao adicionar os diretórios apropriados para PATH, as tarefas do Apache Airflow podem encontrar e executar os executáveis necessários.
- PYTHONPATH: usado pelo interpretador do Python para determinar quais diretórios pesquisar módulos e pacotes importados. É uma lista de diretórios que você pode adicionar ao caminho de pesquisa padrão. Isso permite que o interpretador encontre e carregue bibliotecas Python não incluídas na biblioteca padrão ou instaladas nos diretórios do sistema. Use essa variável para adicionar seus módulos e pacotes Python personalizados e usá-los com seus DAGs.
- LD_LIBRARY_PATH: uma variável de ambiente usada pelo vinculador dinâmico e pelo carregador no Linux para encontrar e carregar bibliotecas compartilhadas. Especifica uma lista de diretórios contendo bibliotecas compartilhadas, que são pesquisadas antes dos diretórios padrão da biblioteca do sistema. Use essa variável para especificar seus binários personalizados.

- `CLASSPATH`: usada pelo Ambiente de Execução Java (JRE) e pelo Java Development Kit (JDK) para localizar e carregar classes, bibliotecas e recursos Java em runtime. É uma lista de diretórios, arquivos JAR e arquivos ZIP que contêm código Java compilado.

Como trabalhar com DAGs no Amazon MWAA

Para executar Gráficos acíclicos direcionados (Directed Acyclic Graphs, DAGs) em um ambiente Amazon Managed Workflows for Apache Airflow, copie seus arquivos para o bucket de armazenamento Amazon S3 conectado ao seu ambiente e, em seguida, informe ao Amazon MWAA onde seus DAGs e arquivos complementares estão localizados no console do Amazon MWAA. O Amazon MWAA se encarrega de sincronizar os DAGs entre operadores, agendadores e o servidor web. Este guia descreve como adicionar ou atualizar seus DAGs e instalar plug-ins personalizados e dependências do Python em um ambiente do Amazon MWAA.

Tópicos

Visão geral do bucket Amazon S3

Um bucket do Amazon S3 para um ambiente Amazon MWAA deve ter acesso público bloqueado. Por padrão, todos os recursos do Amazon S3, como buckets, objetos e sub-recursos relacionados (por exemplo, configuração de ciclo de vida), são privados.

- Somente o proprietário do recurso, a conta da AWS que criou o bucket, pode acessar esse recurso. O proprietário do recurso (por exemplo, seu administrador) pode conceder permissões de acesso a outros, criando uma política de controle de acesso.
- A política de acesso que você configura deve ter permissão para adicionar DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` ao seu bucket do Amazon S3. Para ver um exemplo de política que contém as permissões necessárias, consulte AmazonMWAAFullConsoleAccess.

Um bucket do Amazon S3 para um ambiente Amazon MWAA deve ter versionamento habilitado. Quando o controle de versionamento do bucket do Amazon S3 é ativado, sempre que uma nova versão for criada, uma nova cópia será criada.

- O versionamento está habilitado para os plug-ins personalizados em `plugins.zip` a e dependências do Python em um `requirements.txt` no seu bucket do Amazon S3.
- Você deve especificar a versão de um `plugins.zip` e um `requirements.txt` no console do Amazon MWAA sempre que esses arquivos forem atualizados em seu bucket do Amazon S3.

Como adicionar ou atualizar DAGs

Gráficos acíclicos direcionados (Directed Acyclic Graphs, DAGs) são definidos em um arquivo Python que define a estrutura do DAG como código. Você pode usar a AWS CLI, ou o console do Amazon S3, para fazer upload de DAGs para o ambiente. Esta página descreve as etapas para adicionar ou atualizar DAGs do Apache Airflow em seu ambiente Amazon Managed Workflows for Apache Airflow usando a pasta `dags` em seu bucket do Amazon S3.

Seções

Pré-requisitos

Você precisará do seguinte antes de concluir as etapas nesta página.

- Permissões: sua conta AWS deve ter o acesso concedido por seu administrador para a política de controle de acesso AmazonMWAAFullConsoleAccess para seu ambiente. Além disso, seu ambiente Amazon MWAA deve ser autorizado pela seu perfil de execução para acessar os recursos da AWS usados pelo seu ambiente.
- Acesso: se você precisar de acesso a repositórios públicos para instalar dependências diretamente no servidor web, seu ambiente deverá ser configurado com acesso ao servidor web de rede pública. Para obter mais informações, consulte the section called “Modos de acesso do Apache Airflow”.

- Configuração do Amazon S3: o bucket do Amazon S3 usado para armazenar seus DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` deve ser configurado com Acesso público bloqueado e Versionamento habilitado.

Como funciona

Um Gráfico acíclico direcionado (Directed Acyclic Graphs, DAGs) é definido em um único arquivo Python que define a estrutura do DAG como código. Ele consiste em:

- Uma definição de DAG.
- Operadores que descrevem como executar o DAG e as tarefas a serem executadas.
- Relações entre operadores que descrevem a ordem na qual as tarefas devem ser executadas.

Para executar uma plataforma Apache Airflow em um ambiente Amazon MWAA, você precisa copiar sua definição de DAG para a pasta `dags` em seu bucket de armazenamento. Por exemplo, a pasta DAG em seu bucket de armazenamento pode ter a seguinte aparência:

Example Pasta do DAG

```
dags/
# dag_def.py
```

O Amazon MWAA sincroniza automaticamente objetos novos e alterados do seu bucket do Amazon S3 para a pasta `/usr/local/airflow/dags` dos contêineres de scheduler e operador do Amazon MWAA a cada 30 segundos, preservando a hierarquia de arquivos da fonte do Amazon S3, independentemente do tipo de arquivo. O tempo que os novos DAGs levam para aparecer na sua IU do Apache Airflow é controlado por `scheduler.dag_dir_list_interval`. As alterações nos DAGs existentes serão coletadas no próximo ciclo de processamento do DAG.

> **Note**
>
> Você não precisa incluir o arquivo de configuração `airflow.cfg` na pasta do DAG. Você pode substituir as configurações padrão do Apache Airflow no console do Amazon MWAA. Para obter mais informações, consulte Como usar opções de configuração do Apache Airflow no Amazon MWAA.

O que mudou na v2

- Novo: operadores, hooks e executores. As instruções de importação em seus DAGs e os plug-ins personalizados que você especifica em `plugins.zip` no Amazon MWAA mudaram entre o Apache Airflow v1 e o Apache Airflow v2. Por exemplo, `from airflow.contrib.hooks.aws_hook import AwsHook` no Apache Airflow v1 mudou para `from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook` no Apache Airflow v2. Para saber mais, consulte Referência da API Python no Guia de referência do Apache Airflow.

Como testar DAGs usando o utilitário Amazon MWAA CLI

- O utilitário da interface de linha de comandos (CLI) replica localmente um ambiente do Amazon Managed Workflows for Apache Airflow.
- A CLI cria localmente uma imagem de contêiner Docker semelhante a uma imagem de produção do Amazon MWAA. Isso permite que você execute um ambiente Apache Airflow local para desenvolver e testar DAGs, plug-ins personalizados e dependências antes da implantação no Amazon MWAA.
- Para executar a CLI, consulte aws-mwaa-local-runner no GitHub.

Como fazer upload do código do DAG para o Amazon S3

Você pode usar o console do Amazon S3 ou a AWS Command Line Interface (AWS CLI) para fazer upload do código do DAG para o bucket do Amazon S3. As etapas a seguir pressupõem que você esteja fazendo upload do código (`.py`) em uma pasta chamada `dags` em seu bucket do Amazon S3.

Usar a AWS CLI

A AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com os serviços da AWS usando comandos no shell da linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI: instalar a versão 2.
- AWS CLI: configuração rápida com `aws configure`.

Para fazer o upload usando AWS CLI

1. Use o comando a seguir para listar todos os seus buckets do Amazon S3.

```
aws s3 ls
```

2. Use o seguinte comando para listar os arquivos e pastas no bucket do Amazon S3 para seu ambiente.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. Os comandos a seguir fazem upload de um arquivo `dag_def.py` para uma pasta `dags`.

```
aws s3 cp dag_def.py s3://YOUR_S3_BUCKET_NAME/dags/
```

   Se uma pasta chamada `dags` ainda não existir em seu bucket do Amazon S3, este comando cria a pasta `dags` e faz upload do arquivo denominado `dag_def.py` para a nova pasta.

Usar o console do Amazon S3

O console do Amazon S3 é uma interface de usuário baseada na Web que permite criar e gerenciar os recursos no bucket do Amazon S3. As etapas a seguir pressupõem que você tenha uma pasta DAGs chamada `dags`.

Fazer o upload usando o console do Amazon S3

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione o link do bucket do S3 no código do DAG no painel do S3 para abrir seu bucket de armazenamento no console do Amazon S3.
4. Escolha a pasta `dags`.
5. Escolha Upload (Carregar).
6. Escolha Adicionar arquivo.
7. Selecione a cópia local do seu `dag_def.py` e escolha Carregar.

Como especificar o caminho para sua pasta DAGs no console Amazon MWAA (pela primeira vez)

As etapas a seguir pressupõem que você está especificando o caminho para uma pasta do bucket do Amazon S3 chamada `dags`.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha o ambiente em que você deseja executar DAGs.
3. Escolha Editar.
4. No código DAG no painel Amazon S3, escolha Navegar S3 ao lado do campo pasta DAG.
5. Selecione sua pasta `dags`.
6. Selecione Choose (Escolher).
7. Selecione Avançar, Atualizar ambiente.

Como visualizar as alterações na IU do Apache Airflow

Fazendo login no Apache Airflow

Você precisa de permissões Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer para sua conta AWS no AWS Identity and Access Management (IAM) para visualizar sua IU do Apache Airflow.

Para acessar sua IU do Apache Airflow

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Abrir a IU do Airflow.

Próximas etapas

- Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.

Instalando plug-ins personalizados

O Amazon Managed Workflows for Apache Airflow oferece suporte ao gerenciador de plug-ins integrado do Apache Airflow, permitindo o uso de operadores, hooks, sensores ou interfaces personalizados do Apache Airflow. Esta página descreve as etapas para instalar plugins personalizados do Apache Airflow em seu ambiente do Amazon MWAA usando um arquivo `plugins.zip`.

Sumário

Pré-requisitos

Você precisará do seguinte antes de concluir as etapas nesta página.

- Permissões: sua conta AWS deve ter o acesso concedido por seu administrador para a política de controle de acesso AmazonMWAAFullConsoleAccess para seu ambiente. Além disso, seu ambiente Amazon MWAA deve ser autorizado pela seu perfil de execução para acessar os recursos da AWS usados pelo seu ambiente.
- Acesso: se você precisar de acesso a repositórios públicos para instalar dependências diretamente no servidor web, seu ambiente deverá ser configurado com acesso ao servidor web de rede pública. Para obter mais informações, consulte the section called “Modos de acesso do Apache Airflow”.
- Configuração do Amazon S3: o bucket do Amazon S3 usado para armazenar seus DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` deve ser configurado com Acesso público bloqueado e Versionamento habilitado.

Como funciona

Para executar plug-ins personalizados em seu ambiente, você deve fazer três coisas:

1. Crie um arquivo `plugins.zip` localmente.
2. Faça upload do arquivo `plugins.zip` local para seu bucket no Amazon S3.
3. Especifique a versão desse arquivo no campo Arquivo de plug-ins no console do Amazon MWAA.

> **Note**
>
> Se for a primeira vez que você faz o upload de um `plugins.zip` para o seu bucket do Amazon S3, também será preciso especificar o caminho para o arquivo no console do Amazon MWAA. Você só precisa concluir esta etapa uma vez.

O que mudou na v2

- Novo: operadores, hooks e executores. As instruções de importação em seus DAGs e os plug-ins personalizados que você especifica em `plugins.zip` no Amazon MWAA mudaram entre o Apache Airflow v1 e o Apache Airflow v2. Por exemplo, `from airflow.contrib.hooks.aws_hook import AwsHook` no Apache Airflow v1 mudou para `from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook` no

Apache Airflow v2. Para saber mais, consulte Referência da API Python no Guia de referência do Apache Airflow.

- Novo: Importações em plug-ins. A importação de operadores, sensores e hooks adicionados em plug-ins usando `airflow.{operators,sensors,hooks}.<plugin_name>` não é mais compatível. Essas extensões devem ser importadas como módulos Python regulares. Na versão 2 e versões posteriores, a abordagem recomendada é colocá-los no diretório DAGs e criar e usar um arquivo .airflowignore para excluí-los da análise como DAGs. Para saber mais, consulte Gerenciamento de módulos e Criação de um operador personalizado Guia de referência do Apache Airflow.

Visão geral dos plug-ins personalizados

O gerenciador de plug-ins embutido do Apache Airflow pode integrar atributos externos ao núcleo simplesmente soltando arquivos em uma pasta `$AIRFLOW_HOME/plugins`. Ele permite que você use operadores, hooks, sensores ou interfaces personalizados do Apache Airflow. A seção a seguir fornece um exemplo de estruturas de diretórios simples e aninhadas em um ambiente de desenvolvimento local e as instruções de importação resultantes, que determinam a estrutura de diretórios em um plugins.zip.

Diretório de plug-ins personalizados e limites de tamanho

O programador e os operadores do Apache Airflow procuram plug-ins personalizados durante a inicialização no contêiner Fargate gerenciado pela AWS para seu ambiente em `/usr/local/airflow/plugins/*`.

- Estrutura de diretório. A estrutura do diretório (em `/*`) é baseada no conteúdo do seu arquivo `plugins.zip`. Por exemplo, se seu `plugins.zip` contiver o diretório `operators` como um diretório de nível superior, o diretório será extraído para `/usr/local/airflow/plugins/operators` em seu ambiente.
- Limites de tamanho. Recomendamos um arquivo `plugins.zip` com menos de 1 GB. Quanto maior o tamanho de um arquivo `plugins.zip`, maior o tempo de inicialização em um ambiente. Embora o Amazon MWAA não limite explicitamente o tamanho de um arquivo `plugins.zip`, se as dependências não puderem ser instaladas em dez minutos, o serviço Fargate atingirá o tempo limite e tentará reverter o ambiente para um estado estável.

> **Note**
>
> Para ambientes que usam o Apache Airflow v1.10.12 ou o Apache Airflow v2.0.2, o Amazon MWAA limita o tráfego de saída no servidor web Apache Airflow e não permite que você instale plug-ins nem dependências do Python diretamente no servidor web. A partir do Apache Airflow v2.2.2, o Amazon MWAA pode instalar plug-ins e dependências diretamente no servidor web.

Exemplos de plug-ins personalizados

A seção a seguir usa um exemplo de código no Guia de referência do Apache Airflow para mostrar como estruturar seu ambiente de desenvolvimento local.

Exemplo de uso de uma estrutura de diretórios simples em plugins.zip

Apache Airflow v2

O exemplo a seguir mostra um arquivo `plugins.zip` com uma estrutura de diretório simples para o Apache Airflow v2.

Example diretório simples com PythonVirtualEnvOperator plugins.zip

O exemplo a seguir mostra a árvore de nível superior de um arquivo plugins.zip para o plug-in personalizado PythonVirtualEnvOperator em Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

O exemplo a seguir mostra o plug-in personalizado do PythonVirtualEnvOperator.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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 airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v1

O exemplo a seguir mostra um arquivo `plugins.zip` com uma estrutura de diretório simples para o Apache Airflow v1.

Example diretório simples com PythonVirtualEnvOperator plugins.zip

O exemplo a seguir mostra a árvore de nível superior de um arquivo plugins.zip para o plug-in personalizado PythonVirtualEnvOperator em Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

O exemplo a seguir mostra o plug-in personalizado do PythonVirtualEnvOperator.

```
from airflow.plugins_manager import AirflowPlugin
from airflow.operators.python_operator import PythonVirtualenvOperator

def _generate_virtualenv_cmd(self, tmp_dir):
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if self.system_site_packages:
        cmd.append('--system-site-packages')
    if self.python_version is not None:
        cmd.append('--python=python{}'.format(self.python_version))
    return cmd
PythonVirtualenvOperator._generate_virtualenv_cmd=_generate_virtualenv_cmd

class EnvVarPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Exemplo de uso de uma estrutura de diretórios aninhados em plugins.zip

Apache Airflow v2

O exemplo a seguir mostra um arquivo `plugins.zip` com diretórios separados para `hooks`, `operators` e um diretório `sensors` para o Apache Airflow v2.

Example plugins.zip

```
__init__.py
my_airflow_plugin.py
hooks/
|-- __init__.py
|-- my_airflow_hook.py
operators/
|-- __init__.py
|-- my_airflow_operator.py
|-- hello_operator.py
sensors/
|-- __init__.py
|-- my_airflow_sensor.py
```

O exemplo a seguir mostra as instruções de importação no DAG (pasta DAGs) que usa os plug-ins personalizados.

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_airflow_operator import MyOperator
from sensors.my_airflow_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
  task_id='taskB',
  my_field='some text'
 )

 hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



 sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
```

```
from operators.my_airflow_operator import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

Os exemplos a seguir mostram cada uma das instruções de importação necessárias nos arquivos de plug-in personalizados.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensors/my_airflow_sensor.py

```
from airflow.sensors.base import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example operators/my_airflow_operator.py

```
from airflow.operators.bash import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_airflow_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example operators/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

Siga as etapas em Teste de plug-ins personalizados usando o utilitário Amazon MWAA CLI e Criando um arquivo plugins.zip para compactar o conteúdo dentro em seu diretório `plugins`. Por exemplo, `cd plugins`.

Apache Airflow v1

O exemplo a seguir mostra um arquivo `plugins.zip` com diretórios separados para `hooks`, `operators` e um diretório `sensors` para o Apache Airflow v1.10.12.

Example plugins.zip

```
__init__.py
my_airflow_plugin.py
hooks/
    |-- __init__.py
    |-- my_airflow_hook.py
operators/
    |-- __init__.py
    |-- my_airflow_operator.py
    |-- hello_operator.py
sensors/
    |-- __init__.py
    |-- my_airflow_sensor.py
```

O exemplo a seguir mostra as instruções de importação no DAG (pasta DAGs) que usa os plug-ins personalizados.

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_operator import MyOperator
from sensors.my_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
```

```
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
  task_id='taskB',
  my_field='some text'
 )

 hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



 sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
from operators.my_airflow_operator import *
from utils.my_utils import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

Os exemplos a seguir mostram cada uma das instruções de importação necessárias nos arquivos de plug-in personalizados.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base_hook import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensors/my_airflow_sensor.py

```
from airflow.sensors.base_sensor_operator import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example operators/my_airflow_operator.py

```
from airflow.operators.bash_operator import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field
```

```
    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example operators/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

Siga as etapas em Teste de plug-ins personalizados usando o utilitário Amazon MWAA CLI e Criando um arquivo plugins.zip para compactar o conteúdo dentro em seu diretório `plugins`. Por exemplo, `cd plugins`.

Criação de um arquivo plugins.zip

As etapas a seguir descrevem as etapas que recomendamos para criar um arquivo plugins.zip localmente.

Etapa 1: testar plug-ins personalizados usando o utilitário Amazon MWAA CLI

- O utilitário da interface de linha de comandos (CLI) replica localmente um ambiente do Amazon Managed Workflows for Apache Airflow.
- A CLI cria localmente uma imagem de contêiner Docker semelhante a uma imagem de produção do Amazon MWAA. Isso permite que você execute um ambiente Apache Airflow local para

desenvolver e testar DAGs, plug-ins personalizados e dependências antes da implantação no Amazon MWAA.

- Para executar a CLI, consulte aws-mwaa-local-runner no GitHub.

Etapa 2: criar o arquivo plugins.zip

Você pode usar um utilitário de arquivamento ZIP incorporado ou qualquer outro utilitário ZIP (como 7zip) para criar um arquivo.zip.

> **Note**
>
> O utilitário zip integrado para o sistema operacional Windows pode adicionar subpastas quando você cria um arquivo .zip. Recomendamos verificar o conteúdo do arquivo plugins.zip antes de fazer o upload para o bucket do Amazon S3 para garantir que nenhum diretório adicional tenha sido adicionado.

1. Altere os diretórios para o diretório local de plug-ins do Airflow. Por exemplo:

```
myproject$ cd plugins
```

2. Execute o comando a seguir para garantir que o conteúdo tenha permissões executáveis (somente macOS e Linux).

```
plugins$ chmod -R 755 .
```

3. Compacte o conteúdo dentro de sua pasta `plugins`.

```
plugins$ zip -r plugins.zip .
```

Como fazer upload de **`plugins.zip`** para o Amazon S3

Você pode usar o console do Amazon S3 ou a AWS Command Line Interface (AWS CLI) para fazer upload do arquivo `plugins.zip` para o bucket do Amazon S3.

Usar a AWS CLI

A AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite interagir com os serviços da AWS usando comandos no shell da linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI: instalar a versão 2.
- AWS CLI: configuração rápida com `aws configure`.

Para fazer o upload usando AWS CLI

1. No prompt de comando, navegue até o diretório em que seu arquivo `plugins.zip` está armazenado. Por exemplo:

```
cd plugins
```

2. Use o comando a seguir para listar todos os seus buckets do Amazon S3.

```
aws s3 ls
```

3. Use o seguinte comando para listar os arquivos e pastas no bucket do Amazon S3 para seu ambiente.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

4. Use o seguinte comando para carregar o `plugins.zip` arquivo no bucket do Amazon S3 para seu ambiente.

```
aws s3 cp plugins.zip s3://YOUR_S3_BUCKET_NAME/plugins.zip
```

Usar o console do Amazon S3

O console do Amazon S3 é uma interface de usuário baseada na Web que permite criar e gerenciar os recursos no bucket do Amazon S3.

Fazer o upload usando o console do Amazon S3

1. Abra a página Ambientes no console do Amazon MWAA.

2. Escolha um ambiente.
3. Selecione o link do bucket do S3 no código do DAG no painel do S3 para abrir seu bucket de armazenamento no console do Amazon S3.
4. Escolha Upload (Carregar).
5. Escolha Adicionar arquivo.
6. Selecione a cópia local do seu `plugins.zip` e escolha Carregar.

Instalando plug-ins personalizados em seu ambiente

Esta seção descreve como instalar os plug-ins personalizados que você carregou no seu bucket do Amazon S3 especificando o caminho para o arquivo plugins.zip e especificando a versão do arquivo plugins.zip sempre que o arquivo zip for atualizado.

Como especificar o caminho para **`plugins.zip`** no console Amazon MWAA (pela primeira vez)

Se for a primeira vez que você faz o upload de um `plugins.zip` para o seu bucket do Amazon S3, também será preciso especificar o caminho para o arquivo no console do Amazon MWAA. Você só precisa concluir esta etapa uma vez.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Editar.
4. No código DAG no painel Amazon S3, escolha Browse S3 (Navegar S3) ao lado do campo Arquivos de plugin - opcional.
5. Selecione o `plugins.zip` arquivo no bucket do Amazon S3.
6. Selecione Choose (Escolher).
7. Selecione Avançar, Atualizar ambiente.

Como especificar a versão **`plugins.zip`** no console do Amazon MWAA

É necessário especificar a versão do seu arquivo `plugins.zip` no console do Amazon MWAA sempre que você fizer o upload de uma nova versão do seu `plugins.zip` no bucket do Amazon S3.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Editar.
4. No painel Código DAG no Amazon S3 , escolha uma versão do `plugins.zip` na lista suspensa.
5. Escolha Next (Próximo).

Exemplos de casos de uso para plugins.zip

- Saiba como criar um plug-in personalizado em Criação de um plug-in personalizado com o Apache Hive e o Hadoop.
- Saiba como criar um plug-in personalizado em Plugin personalizado para corrigir PythonVirtualEnvOperator .
- Saiba como criar um plug-in personalizado em Plugin personalizado com a Oracle.
- Saiba como criar um plug-in personalizado em the section called “Como alterar o fuso horário de um DAG”.

Próximas etapas

- Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.

Como instalar dependências do Python

Uma dependência do Python é qualquer pacote ou distribuição que não esteja incluído na instalação básica do Apache Airflow para sua versão do Apache Airflow em seu ambiente Amazon Managed Workflows for Apache Airflow. Esta página descreve as etapas para instalar dependências do Python do Apache Airflow em seu ambiente do Amazon MWAA usando o arquivo `requirements.txt` em seu bucket do Amazon S3.

Sumário

Pré-requisitos

Você precisará do seguinte antes de concluir as etapas nesta página.

- Permissões — Seu AWS administrador deve ter concedido acesso à política de controle de FullConsoleacesso do AmazonMWAA para seu ambiente. Além disso, seu ambiente Amazon MWAA deve ser autorizado pela sua função de execução para acessar os AWS recursos usados pelo seu ambiente.
- Acesso: se você precisar de acesso a repositórios públicos para instalar dependências diretamente no servidor web, seu ambiente deverá ser configurado com acesso ao servidor web de rede pública. Para ter mais informações, consulte the section called “Modos de acesso do Apache Airflow”.
- Configuração do Amazon S3: o bucket do Amazon S3 usado para armazenar seus DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` deve ser configurado com Acesso público bloqueado e Versionamento habilitado.

Como funciona

No Amazon MWAA, você instala todas as dependências do Python fazendo o upload de um arquivo `requirements.txt` no bucket do Amazon S3 e então especificando a versão do arquivo no

console do Amazon MWAA sempre que atualiza o arquivo. O Amazon MWAA executa `pip3 install -r requirements.txt` para instalar as dependências do Python no programador do Apache Airflow e em cada um dos operadores.

Para executar dependências do Python em seu ambiente, você deve fazer três coisas:

1. Crie um arquivo `requirements.txt` localmente.
2. Faça upload do `requirements.txt` local para seu bucket no Amazon S3.
3. Especifique a versão desse arquivo no campo Arquivo de requisitos no console do Amazon MWAA.

> **Note**
>
> Se for a primeira vez que você cria e faz o upload de um `requirements.txt` para o seu bucket do Amazon S3, também será preciso especificar o caminho para o arquivo no console do Amazon MWAA. Você só precisa concluir esta etapa uma vez.

Visão geral das dependências do Python

Você pode instalar extras do Apache Airflow e outras dependências do Python a partir do Python Package Index (PyPi.org), do Python wheels (`.whl`) ou das dependências do Python hospedadas em um repositório privado compatível com /PEP-503 em seu ambiente. PyPi

Limites de localização e tamanho das dependências do Python

O Apache Airflow Scheduler e os Workers procuram plug-ins personalizados durante a inicialização no contêiner AWS Fargate gerenciado para seu ambiente em. `/usr/local/airflow/plugins`

- Limites de tamanho. Recomendamos um arquivo `requirements.txt` que faça referência a bibliotecas cujo tamanho combinado seja menor que 1 GB. Quanto mais bibliotecas o Amazon MWAA precisar instalar, maior será o tempo de inicialização em um ambiente. Embora o Amazon MWAA não limite explicitamente o tamanho das bibliotecas instaladas, se as dependências não puderem ser instaladas em dez minutos, o serviço Fargate atingirá o tempo limite e tentará reverter o ambiente para um estado estável.

Como criar um arquivo requirements.txt

As etapas a seguir descrevem as etapas que recomendamos para criar um arquivo requirements.txt localmente.

Etapa 1: teste as dependências do Python usando o utilitário Amazon MWAA CLI

- O utilitário da interface de linha de comandos (CLI) replica localmente um ambiente do Amazon Managed Workflows for Apache Airflow.
- A CLI cria localmente uma imagem de contêiner Docker semelhante a uma imagem de produção do Amazon MWAA. Isso permite que você execute um ambiente Apache Airflow local para desenvolver e testar DAGs, plug-ins personalizados e dependências antes da implantação no Amazon MWAA.
- Para executar a CLI, consulte o aws-mwaa-local-runner em. GitHub

Etapa 2: criar o **`requirements.txt`**

A seção a seguir descreve como especificar dependências do Python do Python Package Index em um arquivo `requirements.txt`.

Apache Airflow v2

1. Testar localmente. Adicione bibliotecas adicionais de forma iterativa para encontrar a combinação certa de pacotes e suas versões, antes de criar um arquivo `requirements.txt`. Para executar o utilitário Amazon MWAA CLI, consulte o aws-mwaa-local-runner em. GitHub
2. Revise os extras do pacote Apache Airflow. Para ver uma lista dos pacotes instalados para o Apache Airflow v2 no Amazon MWAA, consulte Amazon MWAA local runner no site. `requirements.txt` GitHub
3. Adicione uma declaração de restrições. Adicione o arquivo de restrições do seu ambiente Apache Airflow v2 na parte superior do seu arquivo `requirements.txt`. Os arquivos de restrições do Apache Airflow especificam as versões do provedor disponíveis no momento de um lançamento do Apache Airflow.

   A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para

garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.

No exemplo a seguir, substitua *{environment-version}* pelo número da versão do seu ambiente e *{Python-version} pela versão do Python compatível com* seu ambiente.

Para obter informações sobre a versão do Python compatível com seu ambiente Apache Airflow, consulte Versões do Apache Airflow.

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

Se o arquivo de restrições determinar que o pacote `xyz==1.0` não é compatível com outros pacotes em seu ambiente, `pip3 install` não conseguirá impedir que bibliotecas incompatíveis sejam instaladas em seu ambiente. Se a instalação falhar em algum pacote, você poderá visualizar os registros de erros de cada componente do Apache Airflow (o agendador, o trabalhador e o servidor web) no fluxo de registros correspondente em Logs. CloudWatch Para obter mais informações sobre os tipos de log, consulte the section called "Como visualizar logs do Airflow".

4. Pacotes do Apache Airflow. Adicione os extras do pacote e a versão (==). Isso ajuda a evitar que pacotes com o mesmo nome, mas com versões diferentes, sejam instalados em seu ambiente.

```
apache-airflow[package-extra]==2.5.1
```

5. Bibliotecas Python. Adicione o nome do pacote e a versão (==) em seu arquivo `requirements.txt`. Isso ajuda a evitar que uma atualização futura de última hora do PyPidomínio.org seja aplicada automaticamente.

```
library == version
```

Example Boto3 e psycopg2-binary

Esse exemplo de código é fornecido para fins de demonstração. As bibliotecas boto e psycopg2-binary estão incluídas na instalação básica do Apache Airflow v2 e não precisam ser especificadas em um arquivo `requirements.txt`.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Se um pacote for especificado sem uma versão, o Amazon MWAA instalará a versão mais recente do pacote em .org. PyPi Esta versão pode entrar em conflito com outros pacotes em seu `requirements.txt`.

Apache Airflow v1

1. Testar localmente. Adicione bibliotecas adicionais de forma iterativa para encontrar a combinação certa de pacotes e suas versões, antes de criar um arquivo `requirements.txt`. Para executar o utilitário Amazon MWAA CLI, consulte o aws-mwaa-local-runner em. GitHub
2. Revise os extras do pacote Airflow. Revise a lista de pacotes disponíveis para o Apache Airflow v1.10.12 em https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt.
3. Adicione o arquivo de restrições. Adicione o arquivo de restrições do Apache Airflow v1.10.12 na parte superior do seu arquivo `requirements.txt`. Se o arquivo de restrições determinar que o pacote `xyz==1.0` não é compatível com outros pacotes em seu ambiente, `pip3 install` não conseguirá impedir que bibliotecas incompatíveis sejam instaladas em seu ambiente.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-1.10.12/constraints-3.7.txt"
   ```
4. Pacotes do Apache Airflow v1.10.12. Adicione os extras do pacote Airflow e a versão v1.10.12 do Apache Airflow (==). Isso ajuda a evitar que pacotes com o mesmo nome, mas com versões diferentes, sejam instalados em seu ambiente.

   ```
   apache-airflow[package]==1.10.12
   ```

   Example Secure Shell (SSH)

   O arquivo `requirements.txt` de exemplo a seguir, instala o SSH para o Apache Airflow v1.10.12.

```
apache-airflow[ssh]==1.10.12
```

5. Bibliotecas Python. Adicione o nome do pacote e a versão (==) em seu arquivo `requirements.txt`. Isso ajuda a evitar que uma atualização futura de última hora do PyPidomínio.org seja aplicada automaticamente.

```
library == version
```

Example Boto3

O arquivo `requirements.txt` de exemplo a seguir, instala a biblioteca Boto3 para o Apache Airflow v1.10.12.

```
boto3 == 1.17.4
```

Se um pacote for especificado sem uma versão, o Amazon MWAA instalará a versão mais recente do pacote em .org. PyPi Esta versão pode entrar em conflito com outros pacotes em seu `requirements.txt`.

Como fazer upload de **`requirements.txt`** para o Amazon S3

Você pode usar o console do Amazon S3 ou o AWS Command Line Interface (AWS CLI) para carregar um `requirements.txt` arquivo no seu bucket do Amazon S3.

Usando o AWS CLI

O AWS Command Line Interface (AWS CLI) é uma ferramenta de código aberto que permite que você interaja com AWS serviços usando comandos em seu shell de linha de comando. Para concluir as etapas nesta página, é necessário o seguinte:

- AWS CLI — Instale a versão 2.
- AWS CLI — Configuração rápida com `aws configure`.

Para fazer o upload usando o AWS CLI

1. Use o comando a seguir para listar todos os seus buckets do Amazon S3.

```
aws s3 ls
```

2. Use o seguinte comando para listar os arquivos e pastas no bucket do Amazon S3 para seu ambiente.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. O comando a seguir faz upload de um arquivo `requirements.txt` para um bucket do Amazon S3.

```
aws s3 cp requirements.txt s3://YOUR_S3_BUCKET_NAME/requirements.txt
```

Usar o console do Amazon S3

O console do Amazon S3 é uma interface de usuário baseada na Web que permite criar e gerenciar os recursos no bucket do Amazon S3.

Fazer o upload usando o console do Amazon S3

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione o link do bucket do S3 no código do DAG no painel do S3 para abrir seu bucket de armazenamento no console do Amazon S3.
4. Escolha Carregar.
5. Escolha Adicionar arquivo.
6. Selecione a cópia local do seu `requirements.txt` e escolha Carregar.

Como instalar dependências do Python em seu ambiente

Esta seção descreve como instalar as dependências que você fez upload no seu bucket do Amazon S3 especificando o caminho para o arquivo requirements.txt e especificando a versão do arquivo requirements.txt sempre que for atualizado.

Como especificar o caminho para `requirements.txt` no console Amazon MWAA (pela primeira vez)

Se for a primeira vez que você cria e faz o upload de um `requirements.txt` para o seu bucket do Amazon S3, também será preciso especificar o caminho para o arquivo no console do Amazon MWAA. Você só precisa concluir esta etapa uma vez.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. No código DAG no painel Amazon S3, escolha Procurar no S3 ao lado do campo Arquivo de requisitos: opcional.
5. Selecione o arquivo `requirements.txt` no bucket do Amazon S3.
6. Selecione Escolher.
7. Selecione Avançar, Atualizar ambiente.

É possível começar a usar os novos pacotes logo após a conclusão da atualização do ambiente.

Como especificar a versão `requirements.txt` no console do Amazon MWAA

É necessário especificar a versão do seu arquivo `requirements.txt` no console do Amazon MWAA sempre que você fizer o upload de uma nova versão do seu `requirements.txt` no bucket do Amazon S3.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. No painel Código DAG no Amazon S3 , escolha uma versão do `requirements.txt` na lista suspensa.
5. Selecione Avançar, Atualizar ambiente.

É possível começar a usar os novos pacotes logo após a conclusão da atualização do ambiente.

Como visualizar logs do seu **`requirements.txt`**

Você pode visualizar os registros do Apache Airflow para o Agendador ao agendar seus fluxos de trabalho e analisar sua pasta `dags`. As etapas a seguir descrevem como abrir o grupo de logs para o Scheduler no console Amazon MWAA e visualizar os registros do Apache Airflow no console Logs. CloudWatch

Para visualizar os logs de um **`requirements.txt`**

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha grupo de logs de agendador do Airflow no painel Monitoramento.
4. Escolha o log `requirements_install_ip` em Fluxos de logs.
5. Você deve ver a lista de pacotes que foram instalados no ambiente em `/usr/local/airflow/.local/bin`. Por exemplo: .

   ```
   Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
   Downloading https://files.pythonhosted.org/
   packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
   appdirs-1.4.4-py2.py3-none-any.whl
   Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
   ```

6. Analise a lista de pacotes e verifique se algum deles encontrou algum erro durante a instalação. Se algo der errado, é possível ver um erro semelhante ao seguinte:

   ```
   2021-03-05T14:34:42.731-07:00
   No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
   airflow/.local/bin (line 4))
   No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
   airflow/.local/bin (line 4))
   ```

Próximas etapas

- Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando o aws-mwaa-local-runner on. GitHub

Como excluir arquivos do Amazon S3

Esta página descreve como o versionamento funciona em um bucket do Amazon S3 para um ambiente Amazon Managed Workflows for Apache Airflow e as etapas para excluir um DAG, `plugins.zip` ou um arquivo `requirements.txt`.

Sumário

Pré-requisitos

Você precisará do seguinte antes de concluir as etapas nesta página.

- Permissões: sua conta AWS deve ter o acesso concedido por seu administrador para a política de controle de acesso AmazonMWAAFullConsoleAccess para seu ambiente. Além disso, seu ambiente Amazon MWAA deve ser autorizado pela seu perfil de execução para acessar os recursos da AWS usados pelo seu ambiente.
- Acesso: se você precisar de acesso a repositórios públicos para instalar dependências diretamente no servidor web, seu ambiente deverá ser configurado com acesso ao servidor web de rede pública. Para obter mais informações, consulte the section called “Modos de acesso do Apache Airflow”.
- Configuração do Amazon S3: o bucket do Amazon S3 usado para armazenar seus DAGs, plug-ins personalizados em `plugins.zip` e dependências do Python em `requirements.txt` deve ser configurado com Acesso público bloqueado e Versionamento habilitado.

Visão geral do versionamento

Os arquivos `requirements.txt` e `plugins.zip` em seu bucket do Amazon S3 são versionados. Quando o versionamento do bucket do Amazon S3 é habilitado para um objeto e um artefato (por exemplo, plugins.zip) é excluído de um bucket do Amazon S3, o arquivo não é totalmente excluído. Sempre que um artefato é excluído no Amazon S3, é criada uma nova cópia do arquivo que é um arquivo 404 (Objeto não encontrado) erro/0k que diz “Não estou aqui”. O Amazon S3 chama isso de marcador de exclusão. Um marcador de exclusão é uma versão “nula” do arquivo com um nome de chave (ou chave) e um ID de versão como qualquer outro objeto.

Recomendamos excluir versões de arquivos e marcadores periodicamente para reduzir os custos de armazenamento do seu bucket Amazon S3. Para excluir completamente as versões “não atuais” (anteriores) do arquivo, você deve excluir as versões dos arquivo(s) e, em seguida, o marcador de exclusão da versão.

Como funciona

O Amazon MWAA executa uma operação de sincronização em seu bucket Amazon S3 a cada trinta segundos. Isso faz com que todas as exclusões do DAG em um bucket do Amazon S3 sejam sincronizadas com a imagem do Airflow do seu contêiner Fargate.

Para arquivos `plugins.zip` e `requirements.txt`, as alterações ocorrem somente após uma atualização do ambiente, quando o Amazon MWAA cria uma nova imagem do Airflow do seu contêiner Fargate com os plug-ins personalizados e as dependências do Python. Se você excluir a versão atual de qualquer arquivo `requirements.txt` ou `plugins.zip` e, em seguida, atualizar seu ambiente sem fornecer uma nova versão para o arquivo excluído, a atualização falhará com uma mensagem de erro, como “Não é possível ler a versão `{version}` do arquivo `{file}`”.

Como excluir um DAG no Amazon S3

Um arquivo DAG (`.py`) não é versionado e pode ser excluído diretamente no console do Amazon S3. As etapas a seguir descrevem como excluir um DAG do bucket do Amazon S3.

Para excluir um DAG

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione o link do bucket do S3 no código do DAG no painel do S3 para abrir seu bucket de armazenamento no console do Amazon S3.

4. Escolha a pasta `dags`.
5. Selecione o DAG, Delete.
6. Em Excluir objetos?, digite `delete`.
7. Escolha Delete objects (Excluir objetos).

> **Note**
>
> O Apache Airflow preserva as execuções históricas do DAG. Depois que um DAG é executado no Apache Airflow, ele permanece na lista de DAGs do Airflow, independentemente do status do arquivo, até que você o exclua no Apache Airflow. Para excluir um DAG no Apache Airflow, escolha o botão vermelho “excluir” na coluna Links.

Como remover um requirements.txt ou plugins.zip “atual” de um ambiente

Atualmente, não há como remover um plugins.zip ou requirements.txt de um ambiente depois de adicionados, mas estamos trabalhando para resolver o problema. Nesse ínterim, uma solução alternativa é apontar para um texto vazio ou arquivo zip, respectivamente.

Como excluir uma versão “não atual” (anterior) do requirements.txt ou plugins.zip

Os arquivos `requirements.txt` e `plugins.zip` em seu bucket do Amazon S3 são versionados no Amazon MWAA. Se quiser excluir totalmente esses arquivos do bucket do Amazon S3, você deve recuperar a versão atual (121212) do objeto (por exemplo, plugins.zip), excluir a versão e, em seguida, remover o marcador de exclusão das versões do arquivo.

Você também pode excluir versões de arquivo “não atuais” (anteriores) no console do Amazon S3; no entanto, você ainda precisará excluir o marcador de exclusão usando uma das seguintes opções.

- Para recuperar a versão do objeto, consulte Como recuperar de versões de objetos de um bucket com versionamento ativado noGuia do Amazon S3.
- Para excluir a versão do objeto, consulte Como excluir de versões de objetos de um bucket com versionamento ativado noGuia do Amazon S3.
- Para remover um marcador de exclusão, consulte Gerenciando marcadores de exclusão no Guia do Amazon S3.

Como usar ciclos de vida para excluir versões “não atuais” (anteriores) e excluir marcadores automaticamente

Você pode configurar uma política de ciclo de vida para seu bucket do Amazon S3 para excluir versões “não atuais” (anteriores) dos arquivos plugins.zip e requirements.txt em seu bucket do Amazon S3 após um determinado número de dias ou para remover o marcador de exclusão de um objeto expirado.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Em Código DAG no Amazon S3, selecione seu bucket do Amazon S3.
4. Selecione Criar regra de ciclo de vida.

Exemplo de política de ciclo de vida para excluir versões “não atuais” do requirements.txt e excluir marcadores automaticamente

O exemplo a seguir mostra como criar uma regra de ciclo de vida que exclui permanentemente as versões “não atuais” de um arquivo requirements.txt e seus marcadores de exclusão após trinta dias.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Em Código DAG no Amazon S3, selecione seu bucket do Amazon S3.
4. Selecione Criar regra de ciclo de vida.
5. Em Nome da regra do ciclo de vida, digite `Delete previous requirements.txt versions and delete markers after thirty days`.
6. Em Prefixo, requisitos.
7. Em Ações da regra do ciclo de vida, selecione Excluir permanentemente versões anteriores dos objetos e Excluir marcadores de exclusão expirados ou uploads incompletos de várias partes.
8. Em Número de dias após os objetos se tornarem versões anteriores, digite `30`.
9. Em Marcadores de exclusão de objetos expirados, selecione Excluir marcadores de exclusão de objetos expirados, os objetos serão excluídos permanentemente após 30 dias.

Próximas etapas

- Saiba mais sobre os marcadores de exclusão do Amazon S3 em Gerenciando marcadores de exclusão.
- Saiba mais sobre os ciclos de vida do Amazon S3 em Como objetos expiram.

Redes

Este guia descreve a configuração de rede do Amazon VPC que você precisará para um ambiente do Amazon MWAA.

Seções

Sobre a rede do Amazon MWAA

Uma Amazon VPC é uma rede virtual vinculada à sua AWS conta. Oferece segurança na nuvem e a capacidade de escalar dinamicamente ao fornecer controle refinado sobre sua infraestrutura virtual e a segmentação do tráfego de rede. Esta página descreve a infraestrutura do Amazon VPC com roteamento público ou roteamento privado necessário para oferecer suporte a um ambiente Amazon Managed Workflows for Apache Airflow.

Sumário

Termos

Roteamento público

Uma rede do Amazon VPC que tem acesso à Internet.

Roteamento privado

Uma rede do Amazon VPC sem acesso à Internet.

O que é compatível

A tabela a seguir descreve os tipos de Amazon VPCs compatíveis com o Amazon MWAA.

Tipos de Amazon VPC	Compatível	
Um Amazon VPC de propriedade da conta que está tentando criar o ambiente.	Sim	
Uma Amazon VPC compartilhada em que várias AWS contas criam seus recursos. AWS	Sim	

Visão geral da infraestrutura da VPC

Ao criar um ambiente Amazon MWAA, o Amazon MWAA cria entre um a dois endpoints da VPC para seu ambiente com base no modo de acesso do Apache Airflow que você escolheu para seu ambiente. Esses endpoints aparecem como interfaces de rede elástica (ENIs) com IPs privados em seu Amazon VPC. Depois que esses endpoints são criados, qualquer tráfego destinado a esses IPs é roteado de forma privada ou pública para os AWS serviços correspondentes usados pelo seu ambiente.

A seção a seguir descreve a infraestrutura do Amazon VPC necessária para rotear o tráfego publicamente pela Internet ou de forma privada dentro do seu Amazon VPC.

Roteamento público pela Internet

Esta seção descreve a infraestrutura do Amazon VPC de um ambiente com roteamento público. Você precisará da seguinte infraestrutura de VPC:

- Um grupo de segurança da VPC. Um grupo de segurança VPC atua como um firewall virtual para sua instância para controlar o tráfego de entrada e saída.
  - Até cinco grupos de segurança podem ser especificados.
  - O grupo de segurança deve especificar uma regra de entrada autorreferenciada para si mesmo.
  - O grupo de segurança deve especificar uma regra de saída para todo o tráfego (`0.0.0.0/0`).
  - O grupo de segurança deve permitir todo o tráfego na regra de autorreferência. Por exemplo, (Recomendado) Exemplo de grupo de segurança autorreferenciado para todos os acessos .
  - Opcionalmente, o grupo de segurança pode restringir ainda mais o tráfego especificando o intervalo de portas para o intervalo de portas HTTPS `443` e um intervalo de portas TCP `5432`. Por exemplo, (Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 5432 e (Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 443.
- Duas sub-redes públicas. Sub-rede pública é uma sub-rede associada a uma tabela de rotas que contém uma rota para um gateway da Internet.
  - São necessárias duas sub-redes públicas. Isso permite que o Amazon MWAA crie uma nova imagem de contêiner para seu ambiente em sua outra zona de disponibilidade, se um contêiner falhar.
  - As duas sub-redes devem estar em zonas de disponibilidade diferentes. Por exemplo, `us-east-1a`, `us-east-1b`.
  - As sub-redes devem ser roteadas para um gateway NAT (ou instância NAT) com um endereço IP elástico (EIP).
  - Adicione uma rota à tabela de rotas da sub-rede que direciona o tráfego de entrada da internet para o gateway da Internet.
- Duas sub-redes privadas. Uma sub-rede privada é uma sub-rede que não é associada a uma tabela de rotas que contém uma rota para um gateway da Internet.
  - São necessárias duas sub-redes privadas. Isso permite que o Amazon MWAA crie uma nova imagem de contêiner para seu ambiente em sua outra zona de disponibilidade, se um contêiner falhar.

  - As duas sub-redes devem estar em zonas de disponibilidade diferentes. Por exemplo, `us-east-1a`, `us-east-1b`.
  - As sub-redes devem ter uma tabela de rotas para um dispositivo NAT (gateway ou instância).
  - A sub-rede pública deve ter uma rota para um gateway da Internet.
- Uma lista de controle de acesso (ACL) à rede. Listas de controle de acesso (ACL) à rede: as ACLs da rede permitem ou negam determinado tráfego de entrada e de saída no nível da sub-rede.
  - A NACL deve ter uma regra de entrada que permita todo o tráfego (`0.0.0.0/0`).
  - A NACL deve ter uma regra de saída que negue todo o tráfego (`0.0.0.0/0`).
  - Por exemplo, (Recomendado) Exemplos de ACLs.
- Dois gateways NAT (ou instâncias NAT). Um dispositivo NAT encaminha o tráfego das instâncias na sub-rede privada para a Internet ou outros AWS serviços e, em seguida, encaminha a resposta de volta para as instâncias.
  - O dispositivo NAT deve estar conectado a uma sub-rede pública. (Um dispositivo NAT por sub-rede pública.)
  - O dispositivo NAT deve ter um endereço IPv4 elástico (EIP) conectado a cada sub-rede pública.
- Um gateway da Internet. Um gateway de Internet conecta uma Amazon VPC à Internet e a outros AWS serviços.
  - Um gateway da Internet deve ser anexado ao Amazon VPC.

Roteamento privado sem acesso à Internet

Esta seção descreve a infraestrutura do Amazon VPC de um ambiente com roteamento privado. Você precisará da seguinte infraestrutura de VPC:

- Um grupo de segurança da VPC. Um grupo de segurança VPC atua como um firewall virtual para sua instância para controlar o tráfego de entrada e saída.
  - Até cinco grupos de segurança podem ser especificados.
  - O grupo de segurança deve especificar uma regra de entrada autorreferenciada para si mesmo.
  - O grupo de segurança deve especificar uma regra de saída para todo o tráfego (`0.0.0.0/0`).
  - O grupo de segurança deve permitir todo o tráfego na regra de autorreferência. Por exemplo, (Recomendado) Exemplo de grupo de segurança autorreferenciado para todos os acessos .
  - Opcionalmente, o grupo de segurança pode restringir ainda mais o tráfego especificando o intervalo de portas para o intervalo de portas HTTPS 443 e um intervalo de portas TCP 5432.

Por exemplo, (Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 5432 e (Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 443.

- Duas sub-redes privadas. Uma sub-rede privada é uma sub-rede que não é associada a uma tabela de rotas que contém uma rota para um gateway da Internet.
  - São necessárias duas sub-redes privadas. Isso permite que o Amazon MWAA crie uma nova imagem de contêiner para seu ambiente em sua outra zona de disponibilidade, se um contêiner falhar.
  - As duas sub-redes devem estar em zonas de disponibilidade diferentes. Por exemplo, `us-east-1a`, `us-east-1b`.
  - As sub-redes devem ter uma tabela de rotas para seus endpoints da VPC.
  - As sub-redes não devem ter uma tabela de rotas para um dispositivo NAT (gateway ou instância) e nemum gateway da Internet.
- Uma lista de controle de acesso (ACL) à rede. Listas de controle de acesso (ACL) à rede: as ACLs da rede permitem ou negam determinado tráfego de entrada e de saída no nível da sub-rede.
  - A NACL deve ter uma regra de entrada que permita todo o tráfego (`0.0.0.0/0`).
  - A NACL deve ter uma regra de saída que negue todo o tráfego (`0.0.0.0/0`).
  - Por exemplo, (Recomendado) Exemplos de ACLs.
- Uma tabela de rotas local. Uma tabela de rotas local é uma rota padrão para comunicação dentro da VPC.
  - A tabela de rotas local deve estar associada às suas sub-redes privadas.
  - A tabela de rotas local deve permitir que as instâncias em sua VPC se comuniquem com a rede. Por exemplo, se você estiver usando um AWS Client VPN para acessar o endpoint da interface VPC para seu servidor Web Apache Airflow, a tabela de rotas deve ser roteada para o endpoint da VPC.
- VPC endpoints para cada AWS serviço usado pelo seu ambiente e endpoints VPC Apache Airflow AWS na mesma região e Amazon VPC do seu ambiente Amazon MWAA.
  - Um VPC endpoint para cada AWS serviço usado pelo ambiente e VPC endpoints para o Apache Airflow. Por exemplo, (Obrigatório) Endpoints da VPC.
  - Os endpoints da VPC devem ter o DNS privado habilitado.
  - Os endpoints da VPC devem estar associados às duas sub-redes privadas do seu ambiente.
  - Os endpoints da VPC devem estar associados ao grupo de segurança do seu ambiente.

- A política de VPC endpoint para cada endpoint deve ser configurada para permitir o acesso aos AWS serviços usados pelo ambiente. Por exemplo, (Recomendado) Exemplo de política de endpoint da VPC para permitir todo o acesso.
- Uma política de endpoint da VPC para o Amazon S3 deve ser configurada para permitir o acesso ao bucket. Por exemplo, (Recomendado) Exemplo de política de endpoint do gateway Amazon S3 para permitir acesso ao bucket.

Exemplos de casos de uso para um modo de acesso Amazon VPC e Apache Airflow

Esta seção descreve os diferentes casos de uso para acesso à rede em seu Amazon VPC e o modo de acesso ao servidor Web Apache Airflow que você deve escolher no console do Amazon MWAA.

O acesso à Internet é permitido: nova rede Amazon VPC

Se o acesso à Internet em sua VPC for permitido pela sua organização e você quiser que os usuários acessem seu servidor Web Apache Airflow pela Internet:

1. Crie uma rede Amazon VPC com acesso à Internet.
2. Crie um ambiente com o modo de acesso à rede pública para seu servidor Web Apache Airflow.
3. O que recomendamos: recomendamos usar o modelo de AWS CloudFormation início rápido que cria a infraestrutura do Amazon VPC, um bucket do Amazon S3 e um ambiente do Amazon MWAA ao mesmo tempo. Para saber mais, consulte Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow.

Se o acesso à Internet em sua VPC for permitido pela sua organização e você quiser que os usuários tenham acesso limitado ao servidor Web Apache Airflow em seu VPC.

1. Crie uma rede Amazon VPC com acesso à Internet.
2. Crie um mecanismo para acessar o endpoint da interface VPC do seu servidor Web Apache Airflow a partir do seu computador.
3. Crie um ambiente com o modo de acesso à rede privada para seu servidor Web Apache Airflow.
4. O que recomendamos:

a. Recomendamos usar o console Amazon MWAA ou Opção um: criar a rede VPC no console Amazon MWAA o AWS CloudFormation modelo em. Opção dois: criar uma rede Amazon VPC com acesso à Internet

b. Recomendamos configurar o acesso usando um AWS Client VPN ao seu servidor Web Apache Airflow em. Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN

O acesso à Internet não é permitido: nova rede Amazon VPC

Se o acesso à Internet em sua VPC não for permitido pela sua organização:

1. Crie uma rede Amazon VPC sem acesso à Internet.
2. Crie um mecanismo para acessar o endpoint da interface VPC do seu servidor Web Apache Airflow a partir do seu computador.
3. Crie VPC endpoints para cada AWS serviço usado pelo seu ambiente.
4. Crie um ambiente com o modo de acesso à rede privada para seu servidor Web Apache Airflow.
5. O que recomendamos:

   a. Recomendamos usar o AWS CloudFormation modelo para criar uma Amazon VPC sem acesso à Internet e os endpoints de VPC para cada serviço AWS usado pela Amazon MWAA em. Opção três: criar uma rede Amazon VPC sem acesso à Internet

   b. Recomendamos configurar o acesso usando um AWS Client VPN ao seu servidor Web Apache Airflow em. Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN

O acesso à Internet não é permitido: rede Amazon VPC existente

Se o acesso à Internet em sua VPC não for permitido pela sua organização e você já tiver a rede Amazon VPC necessária sem acesso à Internet:

1. Crie VPC endpoints para cada AWS serviço usado pelo seu ambiente.
2. Crie endpoints da VPC para o Apache Airflow.
3. Crie um mecanismo para acessar o endpoint da interface VPC do seu servidor Web Apache Airflow a partir do seu computador.
4. Crie um ambiente com o modo de acesso à rede privada para seu servidor Web Apache Airflow.

5. O que recomendamos:

   a. Recomendamos criar e conectar os VPC endpoints necessários para AWS cada serviço usado pelo Amazon MWAA e os endpoints VPC necessários para o Apache Airflow in. Como criar endpoints de serviço de VPC necessários em um Amazon VPC com roteamento privado

   b. Recomendamos configurar o acesso usando um AWS Client VPN ao seu servidor Web Apache Airflow em. Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN

Segurança em sua VPC no Amazon MWAA

Esta página descreve os componentes do Amazon VPC usados para proteger seu ambiente Amazon Managed Workflows for Apache Airflow e as configurações necessárias para esses componentes.

Sumário

Termos

Roteamento público

Uma rede do Amazon VPC que tem acesso à Internet.

Roteamento privado

Uma rede do Amazon VPC sem acesso à Internet.

Visão geral de segurança

Grupos de segurança e listas de controle de acesso (ACLs) fornecem maneiras de controlar o tráfego de rede nas sub-redes e instâncias em seu Amazon VPC usando regras que você especifica.

- O tráfego de rede de e para uma sub-rede pode ser controlado por listas de controle de acesso (ACLs). Você só precisa de uma ACL, e a mesma ACL pode ser usada em vários ambientes.
- O tráfego de rede de e para uma instância pode ser controlado por um grupo de segurança do Amazon VPC. É possível usar de um a cinco grupos de segurança por ambiente.
- O tráfego de rede de e para uma instância também pode ser controlado pelas políticas de endpoint da VPC. Se o acesso à Internet em seu Amazon VPC não for permitido pela sua organização e você estiver usando uma rede Amazon VPC com roteamento privado, uma política de endpoint da VPC será necessária para os endpoints da VPC e os endpoints da VPC do Apache Airflow da AWS.

Lista de controle de acesso (ACLs) de rede

Uma lista de controle de acesso (ACL) à rede permite ou nega determinado tráfego de entrada e de saída no nível da sub-rede. Uma ACL não tem estado, o que significa que as regras de entrada e saída devem ser especificadas separadamente e explicitamente. É usado para especificar os tipos de tráfego de rede que são permitidos para entrar ou sair das instâncias em uma rede VPC.

Todo Amazon VPC tem uma ACL padrão que permite todo o tráfego de entrada e saída. É possível editar as regras de ACL padrão ou criar uma ACL personalizada e anexá-la às suas sub-redes. Uma sub-rede só pode ter uma ACL anexada a ela por vez, mas uma ACL pode ser anexada a várias sub-redes.

(Recomendado) Exemplos de ACLs

O exemplo a seguir mostra as regras de ACL de entrada e saída que podem ser usadas para um Amazon VPC com roteamento público ou roteamento privado.

Número da regra	Tipo	Protocolo	Intervalo de portas	Origem	Permissão/ Negação
100	Todo tráfego IPv4	Todos	Tudo	0.0.0.0/0	Permitir
*	Todo tráfego IPv4	Todos	Tudo	0.0.0.0/0	Deny

Grupos de segurança da VPC

Um grupo de segurança VPC atua como um firewall virtual que controla o tráfego em nível de instância. Um grupo de segurança tem estado, o que significa que, quando uma conexão de entrada é permitida, ele pode responder. É usado para especificar os tipos de tráfego de rede que são permitidos para entrar das instâncias em uma rede VPC.

Todo Amazon VPC tem um grupo de segurança padrão. Por padrão, ele não possui regras de entrada. Uma regra de saída que permite todo tráfego de saída. É possível editar as regras padrão do grupo de segurança ou criar um grupo de segurança personalizado e anexá-lo à seu Amazon VPC. No Amazon MWAA, você precisa configurar regras de entrada e saída para direcionar o tráfego em seus gateways NAT.

(Recomendado) Exemplo de grupo de segurança autorreferenciado para todos os acessos

O exemplo a seguir mostra as regras do grupo de segurança de entrada que permitem todo o tráfego de um Amazon VPC para um Amazon VPC com roteamento público ou roteamento privado. O grupo de segurança neste exemplo é uma regra autorreferenciada para si mesmo.

Tipo	Protocolo	Source type (Tipo de origem)	Origem		
Todo o tráfego	Tudo	Todos	sg-0909e8 e81919/- grupo my-		

Tipo	Protocolo	Source type (Tipo de origem)	Origem		
			mwaa-vpc-security		

O exemplo a seguir mostra as regras do grupo de segurança de saída.

Tipo	Protocolo	Source type (Tipo de origem)	Origem		
Todo o tráfego	Tudo	Tudo	0.0.0.0/0		

(Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 5432

O exemplo a seguir mostra as regras do grupo de segurança de entrada que permitem todo o tráfego HTTPS na porta 5432 para o banco de dados de metadados Amazon Aurora PostgreSQL (de propriedade da Amazon MWAA) para seu ambiente.

> **Note**
>
> Se você optar por restringir o tráfego usando essa regra, precisará adicionar outra regra para permitir o tráfego TCP na porta 443.

Tipo	Protocolo	Intervalo de portas	Tipo de origem	Origem	
TCP personalizado	TCP	5432	Personalizar	sg-0909e8e81919/-grupo my-	

Tipo	Protocolo	Intervalo de portas	Tipo de origem	Origem	
				mwaa-vpc-security	

(Opcional) Exemplo de grupo de segurança que restringe o acesso de entrada à porta 443

O exemplo a seguir mostra as regras do grupo de segurança de entrada que permitem todo o tráfego TCP na porta 443 para o servidor Web Apache Airflow.

Tipo	Protocolo	Intervalo de portas	Tipo de origem	Origem	
HTTPS	TCP	443	Personalizar	sg-0909e8e81919/-grupo my-mwaa-vpc-security	

Políticas de endpoint da VPC (somente roteamento privado)

Uma política de VPC endpoint (AWS PrivateLink) controla o acesso aos AWS serviços da sua sub-rede privada. Uma política de endpoint da VPC é uma política de recursos do IAM que você anexa ao gateway de sua VPC endpoint de interface. Esta seção descreve as permissões necessárias para as políticas de endpoint da VPC para cada endpoint da VPC.

Recomendamos usar uma política de endpoint de interface VPC para cada um dos endpoints de VPC que você criou, que permita acesso total a todos os AWS serviços, e usar sua função de execução exclusivamente para obter permissões. AWS

(Recomendado) Exemplo de política de endpoint da VPC para permitir todo o acesso

O exemplo a seguir mostra uma política de endpoint de interface da VPC para um Amazon VPC com roteamento privado.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

(Recomendado) Exemplo de política de endpoint do gateway Amazon S3 para permitir acesso ao bucket

O exemplo a seguir mostra uma política de endpoint de gateway de VPC que fornece acesso aos buckets do Amazon S3 exigidos para as operações do Amazon ECR para um Amazon VPC com roteamento privado. Isso é necessário para que sua imagem do Amazon ECR seja recuperada, além do bucket em que seus DAGs e arquivos complementares estão armazenados.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Gerenciando o acesso a endpoints Amazon VPC específicos de serviços no Amazon MWAA

Um VPC endpoint (AWS PrivateLink) permite que você conecte sua VPC de forma privada a serviços hospedados em AWS sem precisar de um gateway de Internet, um dispositivo NAT, VPN ou proxies de firewall. Esses endpoints são dispositivos virtuais horizontalmente escaláveis e altamente

disponíveis que permitem a comunicação entre instâncias em sua VPC e serviços. AWS Esta página descreve os endpoints da VPC criados pelo Amazon MWAA e como acessar o endpoint da VPC de seu servidor Web Apache Airflow, se você tiver escolhido o modo de acesso à rede privada no Amazon Managed Workflows for Apache Airflow.

Sumário

Definição de preço

- AWS PrivateLink Definição de preço

Visão geral do endpoint da VPC

Ao criar um ambiente Amazon MWAA, o Amazon MWAA cria entre um a dois endpoints da VPC para seu ambiente. Esses endpoints aparecem como interfaces de rede elástica (ENIs) com IPs privados em seu Amazon VPC. Depois que esses endpoints são criados, qualquer tráfego destinado a esses IPs é roteado de forma privada ou pública para os AWS serviços correspondentes usados pelo seu ambiente.

Modo de acesso à rede pública

Se você escolher o modo de acesso à rede pública para seu servidor Web Apache Airflow, o tráfego de rede será roteado publicamente pela Internet.

- O Amazon MWAA cria um endpoint de interface VPC para seu banco de dados de metadados Amazon Aurora PostgreSQL. O endpoint é criado nas zonas de disponibilidade mapeadas para suas sub-redes privadas e é independente de outras contas. AWS
- Em seguida, o Amazon MWAA vincula um endereço IP de suas sub-redes privadas aos endpoints da interface. Isso foi projetado para apoiar a prática recomendada de vincular um único IP de cada zona de disponibilidade do Amazon VPC.

Modo de acesso à rede privada

Se você escolheu o modo de acesso à rede privada para seu servidor Web Apache Airflow, o tráfego de rede será roteado de forma privada dentro do Amazon VPC.

- O Amazon MWAA cria um endpoint de interface VPC para seu servidor Web do Apache Airflow e uma interface endpoint para seu banco de dados de metadados Amazon Aurora PostgreSQL. Os endpoints são criados nas zonas de disponibilidade mapeadas para suas sub-redes privadas e são independentes de outras contas. AWS
- Em seguida, o Amazon MWAA vincula um endereço IP de suas sub-redes privadas aos endpoints da interface. Isso foi projetado para apoiar a prática recomendada de vincular um único IP de cada zona de disponibilidade do Amazon VPC.

Permissão para usar outros AWS serviços

Os endpoints da interface usam a função de execução do seu ambiente no AWS Identity and Access Management (IAM) para gerenciar a permissão para AWS os recursos usados pelo seu ambiente. À medida que mais AWS serviços são habilitados para um ambiente, cada serviço exigirá que você configure a permissão usando a função de execução do seu ambiente. Para adicionar permissões, consulte Perfil de execução do Amazon MWAA.

Caso tenha escolhido o modo de acesso à rede privada para seu servidor Web Apache Airflow, você também deve permitir permissão na política de endpoint da VPC para cada endpoint. Para saber mais, consulte the section called “Políticas de endpoint da VPC (somente roteamento privado)”.

Como visualizar endpoints da VPC

Esta seção descreve como visualizar os endpoints da VPC criados pelo Amazon MWAA e como identificar os endereços IP privados do seu endpoint da VPC do Apache Airflow.

Como visualizar endpoints da VPC no console da Amazon VPC

A seção a seguir mostra as etapas para visualizar um ou mais endpoints da VPC criados pelo Amazon MWAA e quaisquer endpoints da VPC que você possa ter criado se estiver usando roteamento privado para sua Amazon VPC.

Para visualizar um ou mais endpoints da VPC

1. Abra a página Endpoints no console da Amazon VPC.
2. Use o seletor de AWS região para selecionar sua região.
3. É possível ver um ou mais interfaces de endpoints da VPC criados pelo Amazon MWAA e quaisquer endpoints da VPC que você possa ter criado se estiver usando roteamento privado em sua Amazon VPC.

Para saber mais sobre os endpoints de serviço da VPC necessários para uma Amazon VPC com roteamento privado, consulte Como criar endpoints de serviço de VPC necessários em um Amazon VPC com roteamento privado.

Como identificar os endereços IP privados do seu servidor Web Apache Airflow e do seu endpoint da VPC

As etapas a seguir descrevem como recuperar o nome do host do seu servidor Web Apache Airflow e seu endpoint de interface VPC e seus endereços IP privados.

1. Use o comando a seguir AWS Command Line Interface (AWS CLI) para recuperar o nome do host do seu servidor Web Apache Airflow.

   ```
   aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query
    'Environment.WebserverUrl'
   ```

   Você deverá ver algo semelhante a seguinte resposta:

   ```
   "99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
   ```

2. Execute um comando dig no nome do host retornado na resposta do comando anterior. Por exemplo: .

```
dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-
west-2.airflow.amazonaws.com
```

Você deverá ver algo semelhante a seguinte resposta:

```
vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com.
```

3. Use o comando a seguir AWS Command Line Interface (AWS CLI) para recuperar o nome DNS do VPC endpoint retornado na resposta do comando anterior. Por exemplo: .

```
aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-
svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

Você deverá ver algo semelhante a seguinte resposta:

```
"DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com",
```

4. Execute um comando nslookup ou dig no nome do host do Apache Airflow e no nome DNS do endpoint da VPC para recuperar os endereços IP. Por exemplo: .

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

Você deverá ver algo semelhante a seguinte resposta:

```
192.0.5.1
  192.0.6.1
```

Como acessar o endpoint da VPC para seu servidor Web Apache Airflow (acesso à rede privada)

Caso tenha escolhido o modo de acesso à rede privada para seu servidor Web Apache Airflow, será preciso criar um mecanismo para acessar o endpoint da interface VPC para seu servidor Web

do Apache Airflow. Você deve usar o mesmo Amazon VPC, grupo de segurança VPC e sub-redes privadas do seu ambiente do Amazon MWAA para esses recursos.

Usando um AWS Client VPN

AWS Client VPN é um serviço VPN gerenciado baseado em cliente que permite que você acesse com segurança seus AWS recursos e recursos em sua rede local. É fornecida uma conexão TLS segura de qualquer local usando o cliente OpenVPN.

Recomendamos o seguinte tutorial do Amazon MWAA para configurar um Client VPN: Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN.

Como usar um Linux Bastion Host

Um bastion host é um servidor cujo objetivo é fornecer acesso a uma rede privada a partir de uma rede externa, como pela Internet a partir do seu computador. As instâncias Linux estão em uma sub-rede pública e são configuradas com um grupo de segurança que permite acesso SSH a partir do grupo de segurança anexado à instância subjacente do Amazon EC2 que executa o bastion host.

Recomendamos o seguinte tutorial do Amazon MWAA para configurar um Linux Bastion Host: Tutorial: Como configurar o acesso à rede privada usando um Linux Bastion Host.

Como usar um balanceador de carga (avançado)

A seção a seguir mostra as configurações que você precisará aplicar a um Application Load Balancer.

1. Grupos de destino. Você precisará usar grupos de destino que apontem para os endereços IP privados do seu servidor Web Apache Airflow e do seu endpoint de interface VPC. Recomendamos especificar endereços IP privados como destinos registrados, pois usar apenas um pode reduzir a disponibilidade. Para obter mais informações sobre como identificar os endereços IP privados, consulte the section called “Como identificar os endereços IP privados do seu servidor Web Apache Airflow e do seu endpoint da VPC”.
2. Códigos de status. Recomendamos o uso dos códigos de status `200` e `302` nas configurações do seu grupo de destino. Caso contrário, os destinos poderão ser sinalizados como não íntegros, se o endpoint da VPC do servidor Web do Apache Airflow responder com um erro `302 Redirect`.
3. Receptor HTTPS. Você precisará especificar a porta de destino para o servidor Web Apache Airflow. Por exemplo: .

Protocolo	Port
HTTPS	443

4. Novo domínio do ACM. Se você quiser associar um certificado SSL/TLS AWS Certificate Manager, precisará criar um novo domínio para o ouvinte HTTPS do seu balanceador de carga.
5. Região do certificado ACM. Se quiser associar um certificado SSL/TLS AWS Certificate Manager, você precisará fazer o upload para a mesma AWS região do seu ambiente. Por exemplo: .

    - Example região para fazer o upload do certificado

    ```
    aws acm import-certificate --certificate fileb://Certificate.pem --certificate-
    chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --
    region us-west-2
    ```

Como criar endpoints de serviço de VPC necessários em um Amazon VPC com roteamento privado

Uma rede Amazon VPC existente sem acesso à Internet precisa de endpoints de serviço VPC adicionais (AWS PrivateLink) para usar o Apache Airflow nos fluxos de trabalho gerenciados da Amazon para o Apache Airflow. Esta página descreve os endpoints de VPC necessários para os AWS serviços usados pelo Amazon MWAA, os endpoints de VPC necessários para o Apache Airflow e como criar e conectar os endpoints de VPC a um Amazon VPC existente com roteamento privado.

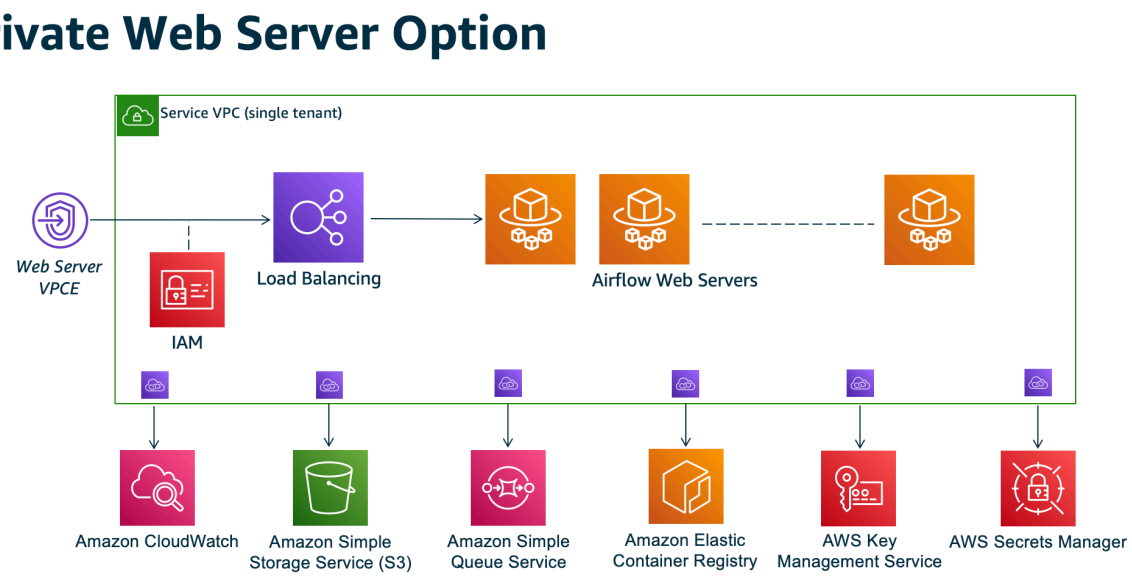
Sumário

- VPCs com DNS personalizado

Definição de preço

- AWS PrivateLink Definição de preço

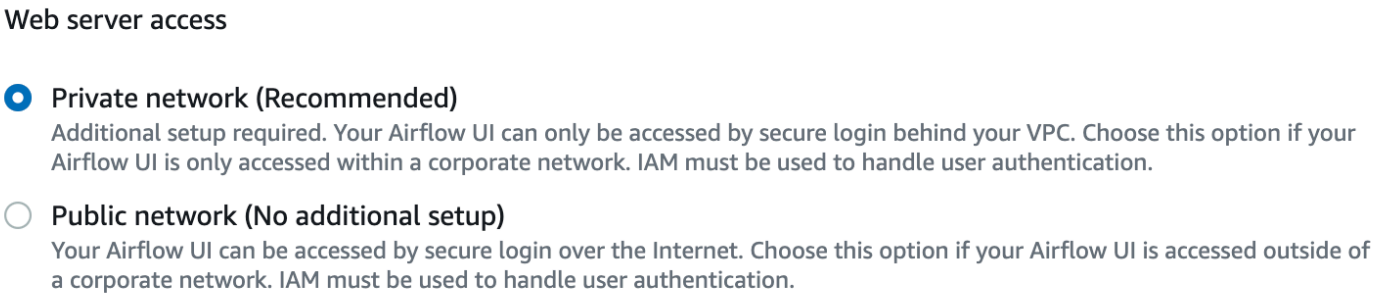
Rede privada e roteamento privado



O modo de acesso à rede privada limita o acesso à interface do usuário do Apache Airflow aos usuários da Amazon VPC que receberam acesso à política do IAM do seu ambiente.

Ao criar um ambiente com acesso privado ao servidor web, você deve empacotar todas as suas dependências em um arquivo wheel do Python (`.whl`) e, em seguida, referenciar `.whl` em seu `requirements.txt`. Para obter instruções sobre como empacotar e instalar suas dependências usando o wheel, consulte Gerenciando dependências usando o Python wheel.

A imagem a seguir mostra onde encontrar a opção Rede privada no console do Amazon MWAA.



- Roteamento privado. Um Amazon VPC sem acesso à Internet limita o tráfego de rede dentro da VPC. Esta página pressupõe que sua Amazon VPC não tenha acesso à Internet e exija endpoints VPC para AWS cada serviço usado por seu ambiente e endpoints VPC para Apache Airflow na mesma região e AWS Amazon VPC que seu ambiente Amazon MWAA.

(Obrigatório) Endpoints da VPC

A seção a seguir mostra os endpoint da VPC necessários para um Amazon VPC sem acesso à Internet. Ele lista os endpoints de VPC para cada AWS serviço usado pelo Amazon MWAA, incluindo os endpoints de VPC necessários para o Apache Airflow.

```
com.amazonaws.YOUR_REGION.s3
com.amazonaws.YOUR_REGION.monitoring
com.amazonaws.YOUR_REGION.ecr.dkr
com.amazonaws.YOUR_REGION.ecr.api
com.amazonaws.YOUR_REGION.logs
com.amazonaws.YOUR_REGION.sqs
com.amazonaws.YOUR_REGION.kms
com.amazonaws.YOUR_REGION.airflow.api
com.amazonaws.YOUR_REGION.airflow.env
com.amazonaws.YOUR_REGION.airflow.ops
```

Como conectar os endpoints da VPC necessários

Esta seção descreve as etapas para conectar os endpoints da VPC necessários para um Amazon VPC com roteamento privado.

VPC endpoints necessários para serviços AWS

A seção a seguir mostra as etapas para conectar os endpoints de VPC para os AWS serviços usados por um ambiente a uma Amazon VPC existente.

Para anexar endpoints da VPC às suas sub-redes privadas

1. Abra a página Endpoints no console da Amazon VPC.
2. Use o seletor de AWS região para selecionar sua região.
3. Criar o endpoint para o Amazon S3:
    a. Escolha Criar Endpoint.
    b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.s3**, digite: e pressione Enter no teclado.
    c. Recomendamos escolher o endpoint de serviço listado para o tipo de Gateway.

       Por exemplo, **com.amazonaws.us-west-2.s3 amazon Gateway**.
    d. Escolha a Amazon VPC do seu ambiente em VPC.
    e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que esse DNS privado esteja ativado selecionando Habilitar nome DNS.
    f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.
    g. Escolha Acesso total na Política.
    h. Escolha Criar endpoint.
4. Crie o primeiro endpoint para o Amazon ECR:
    a. Escolha Criar endpoint .
    b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.ecr.dkr**, digite: e pressione Enter no teclado.
    c. Selecione o endpoint do serviço.
    d. Escolha a Amazon VPC do seu ambiente em VPC.
    e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.
    f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.
    g. Escolha Acesso total na Política.

h. Escolha Criar endpoint.

5. Crie o segundo endpoint para o Amazon ECR:

   a. Escolha Criar Endpoint.

   b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**`.ecr.api`**, digite: e pressione Enter no teclado.

   c. Selecione o endpoint do serviço.

   d. Escolha a Amazon VPC do seu ambiente em VPC.

   e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

   f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

   g. Escolha Acesso total na Política.

   h. Escolha Criar endpoint.

6. Crie o endpoint para CloudWatch Logs:

   a. Escolha Criar Endpoint.

   b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**`.logs`**, digite: e pressione Enter no teclado.

   c. Selecione o endpoint do serviço.

   d. Escolha a Amazon VPC do seu ambiente em VPC.

   e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

   f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

   g. Escolha Acesso total na Política.

   h. Escolha Criar endpoint.

7. Crie o endpoint para CloudWatch monitoramento:

   a. Escolha Criar Endpoint.

   b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**`.monitoring`**, digite: e pressione Enter no teclado.

   c. Selecione o endpoint do serviço.

   d. Escolha a Amazon VPC do seu ambiente em VPC.

e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

g. Escolha Acesso total na Política.

h. Escolha Criar endpoint.

8. Crie o endpoint para Amazon SQS:

a. Escolha Criar Endpoint.

b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.sqs**, digite: e pressione Enter no teclado.

c. Selecione o endpoint do serviço.

d. Escolha a Amazon VPC do seu ambiente em VPC.

e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

g. Escolha Acesso total na Política.

h. Escolha Criar endpoint.

9. Crie o endpoint para AWS KMS:

a. Escolha Criar Endpoint.

b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.kms**, digite: e pressione Enter no teclado.

c. Selecione o endpoint do serviço.

d. Escolha a Amazon VPC do seu ambiente em VPC.

e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

g. Escolha Acesso total na Política.

h. Escolha Criar endpoint.

Endpoints da VPC necessários para o Apache Airflow

A seção a seguir mostra as etapas para conectar os endpoints da VPC do Apache Airflow a um Amazon VPC existente.

Para anexar endpoints da VPC às suas sub-redes privadas

1. Abra a página Endpoints no console da Amazon VPC.
2. Use o seletor de AWS região para selecionar sua região.
3. Crie o endpoint para a API do Apache Airflow:
    a. Escolha Criar Endpoint.
    b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.airflow.api**, digite: e pressione Enter no teclado.
    c. Selecione o endpoint do serviço.
    d. Escolha a Amazon VPC do seu ambiente em VPC.
    e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.
    f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.
    g. Escolha Acesso total na Política.
    h. Escolha Criar endpoint.
4. Crie o primeiro endpoint para o ambiente do Apache Airflow:
    a. Escolha Criar Endpoint.
    b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.airflow.env**, digite: e pressione Enter no teclado.
    c. Selecione o endpoint do serviço.
    d. Escolha a Amazon VPC do seu ambiente em VPC.
    e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.
    f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.
    g. Escolha Acesso total na Política.
    h. Escolha Criar endpoint.
5. Crie o segundo endpoint para as operações do Apache Airflow:

a. Escolha Criar Endpoint.

b. No campo de texto Filtrar por atributos ou pesquisar por palavra-chave**.airflow.ops**, digite: e pressione Enter no teclado.

c. Selecione o endpoint do serviço.

d. Escolha a Amazon VPC do seu ambiente em VPC.

e. Certifique-se de que suas duas sub-redes privadas em diferentes zonas de disponibilidade estejam selecionadas e que a opção Habilitar nome DNS esteja ativada.

f. Escolha o(s) grupo(s) de segurança do Amazon VPC do seu ambiente.

g. Escolha Acesso total na Política.

h. Escolha Criar endpoint.

(Opcional) Habilite endereços IP privados para seu endpoint de interface VPC do Amazon S3

Os endpoints da interface Amazon S3 não oferecem suporte a DNS privado. As solicitações do endpoint S3 ainda são resolvidas para um endereço IP público. Para transformar o endereço do S3 em um endereço IP privado, você precisa adicionar uma zona hospedada privada no Route 53 para o endpoint regional do S3.

Como usar Route 53

Esta seção descreve as etapas para habilitar endereços IP privados para um endpoint da interface S3 usando o Route 53.

1. Crie uma zona hospedada privada para seu endpoint de interface VPC do Amazon S3 (como s3.eu-west-1.amazonaws.com) e associe-a ao seu Amazon VPC.
2. Crie um registro ALIAS A para seu endpoint da interface VPC do Amazon S3 (como s3.eu-west-1.amazonaws.com) que resolva o nome DNS do seu VPC Interface Endpoint.
3. Crie um registro curinga ALIAS A para seu endpoint de interface Amazon S3 (como *.s3.eu-west-1.amazonaws.com) que é resolvido para o nome DNS do VPC Interface Endpoint.

VPCs com DNS personalizado

Se seu Amazon VPC usa roteamento de DNS personalizado, você precisa fazer as alterações em seu resolvedor de DNS (não no Route 53, normalmente uma instância EC2 executando um servidor DNS) ao criar um registro CNAME. Por exemplo: .

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value:  *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

Gerenciando seus próprios endpoints Amazon VPC no Amazon MWAA

O Amazon MWAA usa endpoints Amazon VPC para se integrar a vários AWS serviços necessários para configurar um ambiente Apache Airflow. O gerenciamento de seus próprios endpoints tem dois casos de uso principais:

1. Isso significa que você pode criar ambientes Apache Airflow em um Amazon VPC compartilhado ao usar um AWS Organizationspara gerenciar várias AWS contas e compartilhar recursos.
2. Ele permite que você use políticas de acesso mais restritivas ao restringir suas permissões aos recursos específicos que usam seus endpoints.

Se você optar por gerenciar seus próprios VPC endpoints, será responsável por criar seus próprios endpoints para o ambiente RDS para o banco de dados PostgreSQL e para o servidor web do ambiente.

Para obter mais informações sobre como o Amazon MWAA implanta o Apache Airflow na nuvem, consulte o diagrama de arquitetura do Amazon MWAA.

Criação de um ambiente em uma Amazon VPC compartilhada

Se você usa AWS Organizationspara gerenciar várias AWS contas que compartilham recursos, você pode usar VPC endpoints gerenciados pelo cliente com o Amazon MWAA para compartilhar recursos ambientais com outra conta em sua organização.

Quando você configura o acesso compartilhado à VPC, a conta que possui a Amazon VPC principal (proprietário) compartilha as duas sub-redes privadas exigidas pela Amazon MWAA com outras

contas (participantes) que pertencem à mesma organização. As contas de participantes que compartilham essas sub-redes podem visualizar, criar, modificar e excluir ambientes na Amazon VPC compartilhada.

Suponha que você tenha uma conta`Owner`,, que atua como a `Root` conta na organização e possui os recursos da Amazon VPC, e uma conta de participante`Participant`, um membro da mesma organização. Ao `Participant` criar um novo Amazon MWAA na Amazon VPC com o qual ele compartilha, o `Owner` Amazon MWAA primeiro criará os recursos de VPC do serviço e, em seguida, entrará em um estado por até 72 horas. `PENDING`

Depois que o status do ambiente muda de `CREATING` para`PENDING`, um principal agindo em nome de `Owner` cria os endpoints necessários. Para fazer isso, o Amazon MWAA lista o endpoint do banco de dados e do servidor web no console do Amazon MWAA. Você também pode chamar a ação da `GetEnvironment`API para obter os endpoints do serviço.

> **Note**
>
> Se a Amazon VPC que você usa para compartilhar recursos for uma Amazon VPC privada, você ainda deverá concluir as etapas descritas em. the section called “Como gerenciar o acesso às endpoints da VPC” O tópico aborda a configuração de um conjunto diferente de endpoints do Amazon VPC relacionados a outros AWS serviços que AWS se integram, como Amazon ECR, Amazon ECS e Amazon SQS. Esses serviços são essenciais para operar e gerenciar seu ambiente Apache Airflow na nuvem.

Pré-requisitos

Antes de criar um ambiente Amazon MWAA em uma VPC compartilhada, você precisa dos seguintes recursos:

- Uma AWS conta, `Owner` para ser usada como a conta proprietária da Amazon VPC.
- Uma unidade AWS Organizationsorganizacional, `MyOrganization` criada como raiz.
- Uma segunda AWS conta,`Participant`, abaixo `MyOrganization` para servir à conta do participante que cria o novo ambiente.

Além disso, recomendamos que você se familiarize com as responsabilidades e permissões dos proprietários e participantes ao compartilhar recursos na Amazon VPC.

Crie a Amazon VPC

Primeiro, crie uma nova Amazon VPC que as contas do proprietário e do participante compartilharão:

1. Entre no console usando e0`wner`, em seguida, abra o AWS CloudFormation console. Use o modelo a seguir para criar uma pilha. Essa pilha provisiona vários recursos de rede, incluindo uma Amazon VPC e as sub-redes que as duas contas compartilharão nesse cenário.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: >-
  This template deploys a VPC, with a pair of public and private subnets spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each
  AZ), and default routes for them in the private subnets.
Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: mwaa-
  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16
  PublicSubnet1CIDR:
    Description: >-
      Please enter the IP range (CIDR notation) for the public subnet in the
      first Availability Zone
    Type: String
    Default: 10.192.10.0/24
  PublicSubnet2CIDR:
    Description: >-
      Please enter the IP range (CIDR notation) for the public subnet in the
      second Availability Zone
    Type: String
    Default: 10.192.11.0/24
  PrivateSubnet1CIDR:
    Description: >-
      Please enter the IP range (CIDR notation) for the private subnet in the
      first Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: >-
```

```
      Please enter the IP range (CIDR notation) for the private subnet in the
      second Availability Zone
    Type: String
    Default: 10.192.21.0/24
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName
  InternetGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Public Subnet (AZ1)'
  PublicSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
```

```
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Public Subnet (AZ2)'
  PrivateSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Subnet (AZ1)'
  PrivateSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Subnet (AZ2)'
  NatGateway1EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway2EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway1:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1
```

```
  NatGateway2:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2
  PublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Public Routes'
  DefaultPublicRoute:
    Type: 'AWS::EC2::Route'
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
  PrivateRouteTable1:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Routes (AZ1)'
  DefaultPrivateRoute1:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1
  PrivateSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
```

```
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1
  PrivateRouteTable2:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Routes (AZ2)'
  DefaultPrivateRoute2:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2
  PrivateSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2
  SecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupName: mwaa-security-group
      GroupDescription: Security group with a self-referencing inbound rule.
      VpcId: !Ref VPC
  SecurityGroupIngress:
    Type: 'AWS::EC2::SecurityGroupIngress'
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: '-1'
      SourceSecurityGroupId: !Ref SecurityGroup
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join
      - ','
      - - !Ref PublicSubnet1
        - !Ref PublicSubnet2
  PrivateSubnets:
```

```
    Description: A list of the private subnets
    Value: !Join
      - ','
      - - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1
  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2
  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1
  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2
  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress
```

2. Depois que os novos recursos da Amazon VPC tiverem sido provisionados, navegue até o AWS Resource Access Manager console e escolha Criar compartilhamento de recursos.
3. Escolha as sub-redes que você criou na primeira etapa na lista de sub-redes disponíveis com as quais você pode compartilhar. `Participant`

Criar o ambiente do

Conclua as etapas a seguir para criar um ambiente Amazon MWAA com endpoints Amazon VPC gerenciados pelo cliente.

1. Faça login usando `Participant` e abra o console do Amazon MWAA. Conclua a primeira etapa: especifique os detalhes para especificar um bucket do Amazon S3, uma pasta do DAG e dependências para seu novo ambiente. Para obter mais informações, consulte Introdução.
2. Na página Definir configurações avançadas, em Rede, escolha as sub-redes da Amazon VPC compartilhada.
3. Em Gerenciamento de endpoints, escolha CLIENTE na lista suspensa.
4. Mantenha o padrão para as opções restantes na página e, em seguida, escolha Criar ambiente na página Revisar e criar.

O ambiente começa em um `CREATING` estado e depois muda para`PENDING`. Quando o ambiente estiver`PENDING`, anote o nome do serviço do endpoint do banco de dados e o nome do serviço do endpoint do servidor Web (se você configurar um servidor web privado) usando o console.

Quando você cria um novo ambiente usando o console Amazon MWAA. O Amazon MWAA cria um novo grupo de segurança com as regras de entrada e saída necessárias. Anote o ID do security group.

Na próxima seção, `0wner` usaremos os endpoints de serviço e o ID do grupo de segurança para criar novos endpoints da Amazon VPC na Amazon VPC compartilhada.

Crie os endpoints da Amazon VPC

Conclua as etapas a seguir para criar os endpoints da Amazon VPC necessários para seu ambiente.

1. Faça login no site AWS Management Console usando `0wner` o https://console.aws.amazon.com/vpc/ aberto.
2. Escolha Grupos de segurança no painel de navegação esquerdo e, em seguida, crie um novo grupo de segurança na Amazon VPC compartilhada usando as seguintes regras de entrada e saída:

	Tipo	Protocolo	Tipo de origem	Origem
Entrada	Todo o tráfego	Tudo	Todos	Seu grupo de segurança ambiental
Saída	Todo o tráfego	Tudo	Todos	`0.0.0.0/0`

> **Warning**
>
> A `0wner` conta deve configurar um grupo de segurança na `0wner` conta para permitir o tráfego do novo ambiente para a Amazon VPC compartilhada. Você pode fazer isso criando um novo grupo de `0wner` segurança ou editando um existente.

3. Escolha Endpoints e, em seguida, crie novos endpoints para o banco de dados do ambiente e o servidor web (se estiver no modo privado) usando os nomes dos serviços de endpoint das

etapas anteriores. Escolha a Amazon VPC compartilhada, as sub-redes que você usou para o ambiente e o grupo de segurança do ambiente.

Se for bem-sucedido, o ambiente mudará de `PENDING` trás para frente `CREATING` e, finalmente, para`AVAILABLE`. Quando estiver`AVAILABLE`, você poderá entrar no console do Apache Airflow.

Solução de problemas compartilhada do Amazon VPC

Use a referência a seguir para resolver problemas que você encontra ao criar ambientes em uma Amazon VPC compartilhada.

Ambiente no **`PENDING`** status **`CREATE_FAILED`** posterior

- Verifique se `Owner` está compartilhando as sub-redes com `Participant` o uso. AWS Resource Access Manager
- Verifique se os endpoints da Amazon VPC para o banco de dados e o servidor web foram criados nas mesmas sub-redes associadas ao ambiente.
- Verifique se o grupo de segurança usado com seus endpoints permite o tráfego dos grupos de segurança usados para o ambiente. A `Owner` conta cria regras que fazem referência ao grupo de segurança `Participant` como*`account-number`*/*`security-group-id`*:.

Tipo	Protocolo	Tipo de origem	Origem
Todo o tráfego	Tudo	Todos	*123456789 012 /sg-0909e 8e81919*

Para obter mais informações, consulte Responsabilidades e permissões para proprietários e participantes

Ambiente preso no **`PENDING`** status

Verifique o status de cada VPC endpoint para garantir que esteja. `Available` Se você configurar um ambiente com um servidor Web privado, também deverá criar um endpoint para o servidor Web. Se o ambiente estiver preso`PENDING`, isso pode indicar que o endpoint privado do servidor web está ausente.

The Vpc Endpoint Service '*vpce-service-name*' does not existErro recebido

Se você ver o erro a seguir, verifique se a conta que está criando os endpoints está na `0wner` conta que possui a VPC compartilhada:

```
ClientError: An error occurred (InvalidServiceName) when calling the
 CreateVpcEndpoint operation:

The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Tutoriais para Amazon Managed Workflows for Apache Airflow

Este guia inclui step-by-step tutoriais sobre como usar e configurar um ambiente Amazon Managed Workflows para Apache Airflow.

Tópicos

Tutorial: Como configurar o acesso à rede privada usando um AWS Client VPN

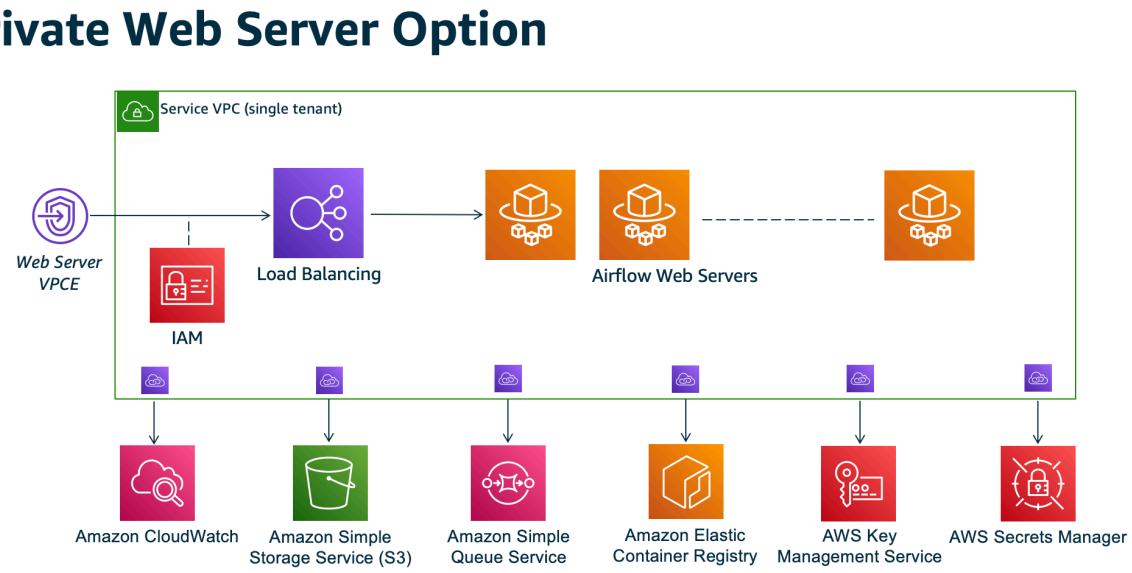
Este tutorial mostra as etapas para criar um túnel VPN do seu computador para o servidor web Apache Airflow para seu ambiente Amazon Managed Workflows for Apache Airflow. Para se conectar à Internet por meio de um túnel VPN, primeiro você precisará criar um endpoint AWS Client VPN. Depois de configurado, um endpoint do cliente VPN atua como um servidor VPN, permitindo uma conexão segura do seu computador com os recursos em sua VPC. Em seguida, você se conectará à VPN do cliente a partir do seu computador usando AWS Client VPN para Desktop.

Seções

- Etapa 6: adicione uma regra de entrada de autorização à sua VPN do cliente
- Etapa 7: baixe o arquivo de configuração do endpoint do cliente VPN
- Etapa 8: conecte-se à AWS Client VPN
- Próximas etapas

Rede privada

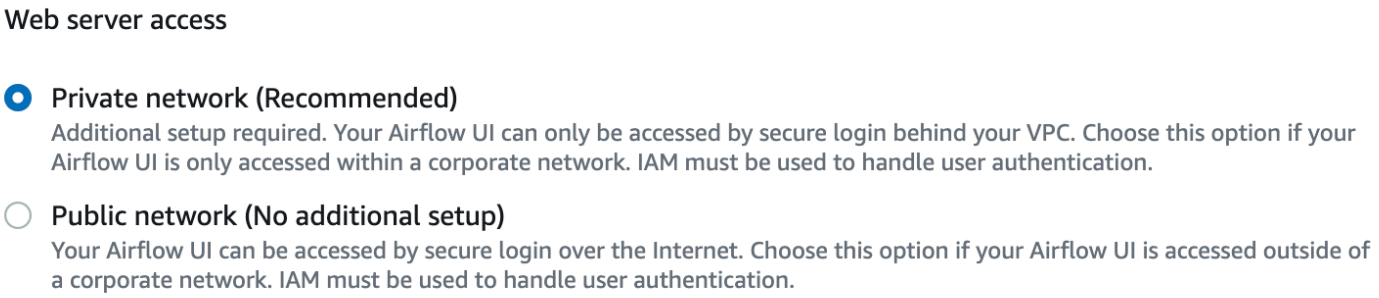
Este tutorial pressupõe que você tenha escolhido o modo de acesso à rede privada para seu servidor Web Apache Airflow.



O modo de acesso à rede privada limita o acesso à interface do usuário do Apache Airflow aos usuários da Amazon VPC que receberam acesso à política do IAM do seu ambiente.

Ao criar um ambiente com acesso privado ao servidor web, você deve empacotar todas as suas dependências em um arquivo wheel do Python (`.whl`) e, em seguida, referenciar `.whl` em seu `requirements.txt`. Para obter instruções sobre como empacotar e instalar suas dependências usando o wheel, consulte Gerenciando dependências usando o Python wheel.

A imagem a seguir mostra onde encontrar a opção Rede privada no console do Amazon MWAA.



Casos de uso

Você pode usar este tutorial antes ou depois de criar um ambiente Amazon MWAA. Você deve usar o mesmo Amazon VPC, grupos de segurança VPC e sub-redes privadas do seu ambiente. Se você usar este tutorial após criar um ambiente Amazon MWAA, depois de concluir as etapas, poderá retornar ao console do Amazon MWAA e alterar o modo de acesso do servidor Web do Apache Airflow para Rede privada.

Antes de começar

1. Verifique as permissões do usuário. Certifique-se de que sua conta no AWS Identity and Access Management (IAM) tenha permissões suficientes para criar e gerenciar recursos de VPC.
2. Use sua VPC do Amazon MWAA. Este tutorial pressupõe que você esteja associando a VPN do cliente a uma VPC existente. O Amazon VPC deve estar na mesma região AWS de um ambiente Amazon MWAA e ter duas sub-redes privadas. Se você não criou um Amazon VPC, use o modelo AWS CloudFormation em Opção três: criar uma rede Amazon VPC sem acesso à Internet.

Objetivos

Neste tutorial, você irá:

1. Criar um endpoint AWS Client VPN usando um modelo AWS CloudFormation para um Amazon VPC existente.
2. Gerir os certificados e as chaves de servidor e cliente e, em seguida, fazer upload do certificado e da chave de servidor para AWS Certificate Manager na mesma região AWS de um ambiente do Amazon MWAA.

3. Baixe e modifique um arquivo de configuração de endpoint do cliente VPN para sua VPN do cliente e use o arquivo para criar um perfil de VPN para se conectar usando a VPN do cliente para Desktop.

(Opcional) Etapa 1: identifique sua VPC, regras CIDR e segurança(s) da VPC

A seção a seguir descreve como encontrar IDs para seu Amazon VPC, grupo de segurança da VPC e uma forma de identificar as regras CIDR necessárias para criar sua VPN do cliente nas etapas subsequentes.

Identifique suas regras CIDR

A seção a seguir mostra como identificar as regras CIDR, que você precisará para criar sua VPN do cliente.

Para identificar o CIDR da sua VPN do cliente

1. Abra a página Suas Amazon VPCs no console do Amazon VPC.
2. Use o seletor de região na barra de navegação para escolher a região AWS como um ambiente Amazon MWAA.
3. Escolha sua Amazon VPC.
4. Supondo que os CIDRs de suas sub-redes privadas são:

   - Sub-rede privada 1: 10.192.10.0/24
   - Sub-rede privada 2: 10.192.11.0/24

   Se o CIDR para seu Amazon VPC for 10.192.0.0/16, o CIDR IPv4 do cliente que você especificaria para sua VPN do cliente seria 10.192.0.0/22.
5. Salve este valor de CIDR e o valor do seu ID de VPC para as etapas subsequentes.

Identifique sua VPC e o(s) grupo(s) de segurança

A seção a seguir mostra como encontrar o ID do seu Amazon VPC e o(s) grupo(s) de segurança, que você precisará para criar sua VPN do cliente.

> **Note**
>
> Você pode estar usando mais de um grupo de segurança. Você precisará especificar todos os grupos de segurança da sua VPC nas etapas subsequentes.

Para identificar o(s) grupo(s) de segurança

1. Abra a página Grupo de segurança no console do Amazon VPC.
2. Use o seletor de região na barra de navegação para escolher a região AWS.
3. Procure o Amazon VPC no VPC ID e identifique os grupos de segurança associados à VPC.
4. Salve o ID de seu(s) grupo(s) de segurança e da VPC para as etapas subsequentes.

Etapa 2: crie os certificados de servidor e de cliente

Um endpoint do cliente VPN é compatível apenas com tamanhos de chave RSA de 1024 bits e 2048 bits. O procedimento a seguir mostra como usar o OpenVPN easy-rsa para gerar os certificados e as chaves de servidor e cliente e faça o upload dos certificados para ACM usando a AWS Command Line Interface (AWS CLI).

Criar os certificados de cliente

1. Siga estas etapas rápidas para criar e fazer upload dos certificados no ACM por meio da AWS CLI em Autenticação e autorização do cliente: autenticação mútua.
2. Nestas etapas, você deve especificar a mesma região AWS de um ambiente Amazon MWAA no comando AWS CLI ao fazer upload de seus certificados de servidor e cliente. Veja a seguir alguns exemplos de como especificar a região nestes comandos:

   a. Example região para certificado de servidor

   ```
   aws acm import-certificate --certificate fileb://server.crt --private-key
    fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
   ```

   b. Example região para certificado de cliente

   ```
   aws acm import-certificate --certificate fileb://client1.domain.tld.crt
    --private-key fileb://client1.domain.tld.key --certificate-chain fileb://
   ca.crt --region us-west-2
   ```

c. Após essas etapas, salve o valor retornado na resposta AWS CLI para os ARNs do certificado do servidor e do certificado do cliente. Você especificará esses ARNs em seu modelo AWS CloudFormation para criar a VPN do cliente.

3. Nestas etapas, um certificado de cliente e uma chave privada são salvos no seu computador. Veja a seguir um exemplo de onde encontrar essas credenciais:

   a. Example No macOS

      No macOS, o conteúdo é salvo em `/Users/youruser/custom_folder`. Se você listar todos os conteúdos (`ls -a`) deste diretório, deverá ver algo semelhante ao seguinte:

      ```
      .
      ..
      ca.crt
      client1.domain.tld.crt
      client1.domain.tld.key
      server.crt
      server.key
      ```

   b. Após estas etapas, salve o conteúdo ou anote a localização do certificado do cliente em `client1.domain.tld.crt` e a chave privada em `client1.domain.tld.key`. Você adicionará estes valores ao arquivo de configuração de sua VPN do cliente.

Etapa 3: salve o modelo AWS CloudFormation localmente

A seção a seguir contém o modelo AWS CloudFormation para criar a VPN do cliente. Você deve especificar o mesmo Amazon VPC, grupo(s) de segurança VPC e sub-redes privadas do seu ambiente Amazon MWAA.

- Copie o conteúdo do modelo a seguir e salve localmente como `mwaa_vpn_client.yaml`. Também é possível baixar o modelo.

  Substitua os seguintes valores:

  - **`YOUR_CLIENT_ROOT_CERTIFICATE_ARN`** - o ARN do seu certificado client1.domain.tld em `ClientRootCertificateChainArn`.
  - **`YOUR_SERVER_CERTIFICATE_ARN`** - o ARN do seu certificado server em `ServerCertificateArn`.

- A regra CIDR IPv4 do cliente em `ClientCidrBlock`. É fornecida uma regra CIDR de `10.192.0.0/22`.
- Seu ID da Amazon VPC está em `VpcId`. É fornecida uma VPC de `vpc-010101010101`.
- Seus IDs de grupo de segurança da VPC em `SecurityGroupIds`. É fornecido um grupo de segurança de `sg-0101010101`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: This template deploys a VPN Client Endpoint.
Resources:
  ClientVpnEndpoint:
    Type: 'AWS::EC2::ClientVpnEndpoint'
    Properties:
      AuthenticationOptions:
        - Type: "certificate-authentication"
          MutualAuthentication:
            ClientRootCertificateChainArn: "YOUR_CLIENT_ROOT_CERTIFICATE_ARN"
      ClientCidrBlock: 10.192.0.0/22
      ClientConnectOptions:
        Enabled: false
      ConnectionLogOptions:
        Enabled: false
      Description: "MWAA Client VPN"
      DnsServers: []
      SecurityGroupIds:
        - sg-0101010101
      SelfServicePortal: ''
      ServerCertificateArn: "YOUR_SERVER_CERTIFICATE_ARN"
      SplitTunnel: true
      TagSpecifications:
        - ResourceType: "client-vpn-endpoint"
          Tags:
          - Key: Name
            Value: MWAA-Client-VPN
      TransportProtocol: udp
      VpcId: vpc-010101010101
      VpnPort: 443
```

> **Note**
>
> Se você estiver usando mais de um grupo de segurança para seu ambiente, poderá especificar vários grupos de segurança no seguinte formato:
>
> ```
> SecurityGroupIds:
>       - sg-0112233445566778b
>       - sg-0223344556677889f
> ```

Etapa 4: crie a pilha AWS CloudFormation VPN do cliente

Para criar o AWS Client VPN

1. Abra o console do AWS CloudFormation.
2. Escolha O modelo está pronto e Fazer upload de um arquivo de modelo.
3. Selecione Escolher arquivo e selecione seu arquivo `mwaa_vpn_client.yaml`.
4.
5. Escolha Avançar, Avançar.
6. Selecione a confirmação e então escolha Criar pilha.

Etapa 5: associe sub-redes à sua VPN do cliente

Para associar sub-redes privadas à AWS Client VPN

1. Abra Console do Amazon VPC.
2. Escolha a página Endpoint do cliente VPN.
3. Selecione sua VPN do cliente e, em seguida, escolha a guia Associações, Associar.
4. Na lista suspensa, escolha:
   - Seu Amazon VPC em VPC.
   - Uma de suas sub-redes privadas em Escolha uma sub-rede para associar.
5. Escolha Associar.

> **Note**
>
> A VPC e a sub-rede levam vários minutos para serem associadas à VPN do cliente.

Etapa 6: adicione uma regra de entrada de autorização à sua VPN do cliente

Você precisa adicionar uma regra de entrada de autorização usando a regra CIDR da sua VPC à sua VPN do cliente. Se você quiser autorizar usuários ou grupos específicos do seu grupo do Active Directory ou do Provedor de Identidades (IdP) baseado em SAML, consulte as regras de autorização no Guia da VPN do cliente.

Para adicionar o CIDR à AWS Client VPN

1. Abra Console do Amazon VPC.
2. Escolha a página Endpoint do cliente VPN.
3. Selecione sua VPN do cliente e, em seguida, escolha a guia Autorização, Autorizar entrada.
4. Especifique o seguinte:
   - A regra CIDR do Amazon VPC na Rede de destino para habilitação. Por exemplo:

     ```
     10.192.0.0/16
     ```

   - Para Conceder acesso a, escolha Permitir acesso a todos os usuários.
   - Para Descrição, insira um nome descritivo.
5. Escolha Adicionar regra de autorização.

> **Note**
>
> Dependendo dos componentes de rede do seu Amazon VPC, você também pode precisar dessa regra de entrada de autorização na sua lista de controle de acesso à rede (NACL).

Etapa 7: baixe o arquivo de configuração do endpoint do cliente VPN

Para fazer download do arquivo de configuração

1. Siga estas etapas rápidas para baixar o arquivo de configuração do Client VPN em Baixar o arquivo de configuração do endpoint do Client VPN.
2. Nestas etapas, você deverá acrescentar uma string ao nome DNS do endpoint do cliente VPN. Veja um exemplo abaixo:

   - Example nome DNS do endpoint

     Se o nome DNS do endpoint do cliente VPN tiver a seguinte aparência:

     ```
     remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
     ```

     Você pode adicionar uma string para identificar seu endpoint do cliente VPN da seguinte forma:

     ```
     remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-
     west-1.amazonaws.com 443
     ```

3. Nestas etapas, você deve adicionar o conteúdo do certificado do cliente entre um novo conjunto de etiquetas `<cert></cert>` e o conteúdo da chave privada entre um novo conjunto de etiquetas `<key></key>`. Veja um exemplo abaixo:

   a. Abra um prompt de comando e altere os diretórios para o local do certificado do cliente e da chave privada.

   b. Example client1.domain.tld.crt do macOS

      Para mostrar o conteúdo do arquivo `client1.domain.tld.crt` no macOS, você pode usar `cat client1.domain.tld.crt`.

      Copie o valor do terminal e cole em `downloaded-client-config.ovpn`, desta forma:

      ```
      ZZZ1111dddaBBB
      -----END CERTIFICATE-----
      </ca>
      <cert>
      -----BEGIN CERTIFICATE-----
      YOUR client1.domain.tld.crt
      ```

```
-----END CERTIFICATE-----
</cert>
```

c. Example client1.domain.tld.key do macOS

Para mostrar o conteúdo de `client1.domain.tld.key`, você pode usar `cat client1.domain.tld.key`.

Copie o valor do terminal e cole em `downloaded-client-config.ovpn`, desta forma:

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.key
-----END CERTIFICATE-----
</key>
```

Etapa 8: conecte-se à AWS Client VPN

O cliente para AWS Client VPN é fornecido gratuitamente. Você pode conectar seu computador diretamente à AWS Client VPN para uma experiência completa de VPN.

Para se conectar à VPN do cliente

1. Baixe e instale AWS Client VPN for Desktop.
2. Abra a AWS Client VPN.
3. Escolha Arquivo, Perfis gerenciados no menu do cliente VPN.
4. Escolha Adicionar perfil e, em seguida, escolha `downloaded-client-config.ovpn`.
5. Insira um nome descritivo em Nome de exibição.
6. Escolha Adicionar perfil e Concluído.
7. Selecione Connect (Conectar).

Depois de se conectar à VPN do cliente, você precisará se desconectar de outras VPNs para visualizar qualquer um dos recursos em seu Amazon VPC.

> **Note**
>
> Talvez seja necessário sair do cliente e começar de novo antes de conseguir se conectar.

Próximas etapas

- Saiba como criar um ambiente do Amazon MWAA em Comece a usar o Amazon Managed Workflows for Apache Airflow. Você deve criar um ambiente na mesma região AWS como uma VPN do cliente e usar a mesma VPC, sub-redes privadas e grupo de segurança como uma VPN do cliente.

Tutorial: Como configurar o acesso à rede privada usando um Linux Bastion Host

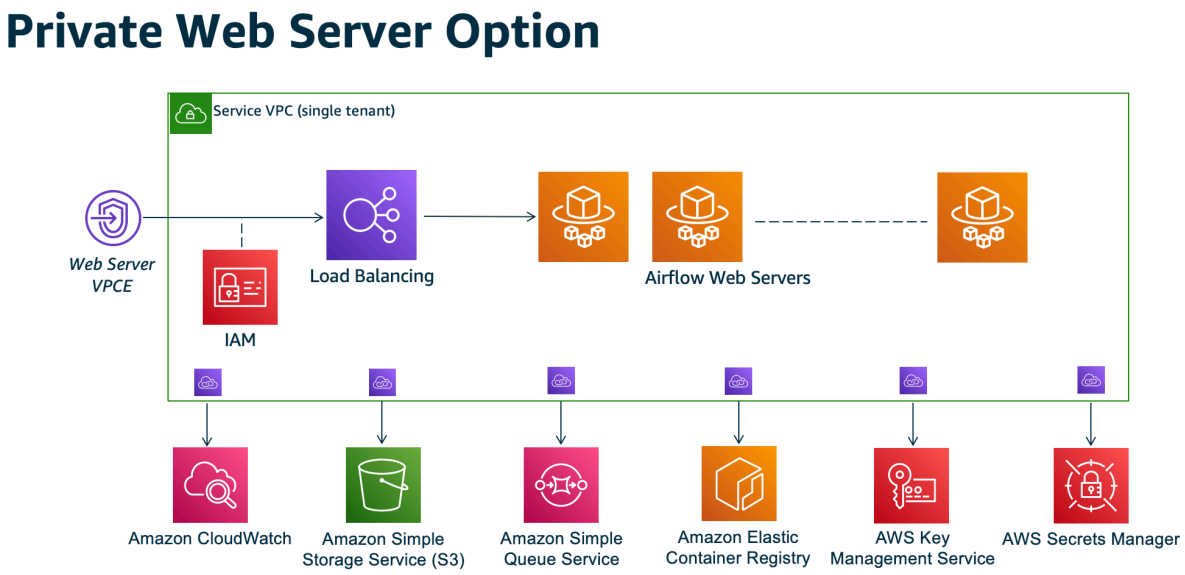
Este tutorial mostra as etapas para criar um túnel SSH do seu computador para o servidor web Apache Airflow para seu ambiente Amazon Managed Workflows for Apache Airflow. Pressupõe-se que você já tenha criado um ambiente Amazon MWAA. Depois de configurado, um Linux Bastion Host atua como um servidor jump, permitindo uma conexão segura do seu computador com os recursos em sua VPC. Em seguida, você usará um complemento de gerenciamento de proxy SOCKS para controlar as configurações de proxy em seu navegador e acessar sua IU do Apache Airflow.

Seções

- Etapa 5: definir as configurações de proxy
- Etapa 6: abra a IU do Apache Airflow
- Próximas etapas

Rede privada

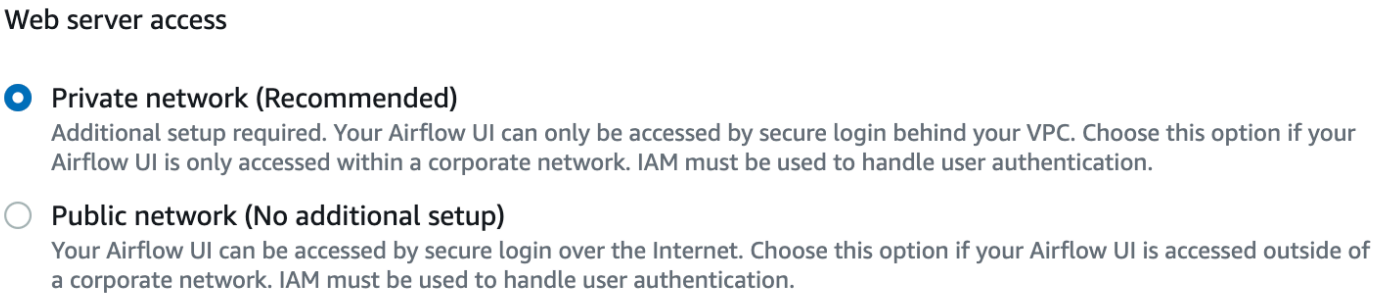
Este tutorial pressupõe que você tenha escolhido o modo de acesso à rede privada para seu servidor Web Apache Airflow.



O modo de acesso à rede privada limita o acesso à interface do usuário do Apache Airflow aos usuários da Amazon VPC que receberam acesso à política do IAM do seu ambiente.

Ao criar um ambiente com acesso privado ao servidor web, você deve empacotar todas as suas dependências em um arquivo wheel do Python (`.whl`) e, em seguida, referenciar `.whl` em seu `requirements.txt`. Para obter instruções sobre como empacotar e instalar suas dependências usando o wheel, consulte Gerenciando dependências usando o Python wheel.

A imagem a seguir mostra onde encontrar a opção Rede privada no console do Amazon MWAA.



Casos de uso

Você pode usar este tutorial depois de criar um ambiente Amazon MWAA. Você deve usar o mesmo Amazon VPC, grupos de segurança VPC e sub-redes públicas do seu ambiente.

Antes de começar

1. Verifique as permissões do usuário. Certifique-se de que sua conta no AWS Identity and Access Management (IAM) tenha permissões suficientes para criar e gerenciar recursos de VPC.
2. Use sua VPC do Amazon MWAA. Este tutorial pressupõe que você esteja associando o bastion host a uma VPC existente. O Amazon VPC deve estar na mesma região de seu ambiente do Amazon MWAA e ter duas sub-redes privadas, como definido em Criar a rede VPC.
3. Crie uma chave SSH. Você precisa criar uma chave SSH do Amazon EC2 (.pem) na mesma região do seu ambiente Amazon MWAA para se conectar aos servidores virtuais. Se você não tiver uma chave SSH, consulte Criar ou importar um par de chaves no Guia do usuário do Amazon EC2.

Objetivos

Neste tutorial, você irá:

1. Crie uma instância do Linux Bastion Host usando um Modelo AWS CloudFormation para uma VPC existente.
2. Autorize o tráfego de entrada para o grupo de segurança da instância bastion usando uma regra de entrada na porta 22.
3. Autorize o tráfego de entrada do grupo de segurança de um ambiente do Amazon MWAA para o grupo de segurança da instância do Bastion.

4. Crie um túnel SSH para a instância do bastion.
5. Instale e configure o FoxyProxy complemento para o navegador Firefox para visualizar a interface do usuário do Apache Airflow.

Etapa 1: criar a instância do bastion

A seção a seguir descreve as etapas para criar a instância Linux Bastion usando um AWS CloudFormation modelo para uma VPC existente no AWS CloudFormation console.

Para criar o Linux Bastion Host

1. Abra a página Deploy Quick Start no AWS CloudFormation console.
2. Use o seletor de região na barra de navegação para escolher a mesma AWS região do seu ambiente Amazon MWAA.
3. Escolha Próximo.
4. Digite um nome no campo de texto Nome da pilha, como.`mwaa-linux-bastion`
5. No painel Parâmetros, Configuração de rede, escolha as seguintes opções:
   a. Escolha o seu ID da VPC do ambiente do Amazon VPC.
   b. Escolha o seu ID da sub-rede pública 1 do ambiente do Amazon VPC.
   c. Escolha o seu ID da sub-rede pública 2 do ambiente do Amazon VPC.
   d. Insira o intervalo de endereços mais estreito possível (por exemplo, um intervalo CIDR interno) em CIDR de acesso externo permitido ao Bastion.

   > **Note**
   >
   > A maneira mais simples de identificar um intervalo é usar o mesmo intervalo CIDR de suas sub-redes públicas. Por exemplo, as sub-redes públicas no AWS CloudFormation modelo na Criar a rede VPC página são e. `10.192.10.0/24` `10.192.11.0/24`

6. No painel Configuração do Amazon EC2, escolha o seguinte:
   a. Escolha sua chave SSH na lista suspensa em Nome do par de chaves.
   b. Insira um nome em Nome do bastion host.
   c. Escolha verdadeiro para Encaminhamento TCP.

> **Warning**
>
> O encaminhamento TCP deve ser definido como verdadeiro nesta etapa. Caso contrário, não será possível criar um túnel SSH na etapa seguinte.

7. Escolha Avançar, Avançar.
8. Selecione a confirmação e então escolha Criar pilha.

Para saber mais sobre a arquitetura do seu Linux Bastion Host, consulte Linux Bastion Hosts on the AWS Cloud: Architecture.

Etapa 2: criar o túnel ssh

As etapas a seguir descrevem como criar o túnel ssh para o seu bastion do Linux. Um túnel SSH recebe a solicitação do seu endereço IP local para o linux bastion, e é por isso que o encaminhamento de TCP para o linux bastion foi configurado para `true` nas etapas anteriores.

macOS/Linux

Para criar um túnel via linha de comando

1. Abra a página Instâncias no console do Amazon EC2.
2. Escolha uma instância.
3. Copie o endereço em DNS IPv4 pública. Por exemplo, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. No prompt de comando, navegue até o diretório em que sua chave SSH está armazenada.
5. Execute o comando a seguir para se conectar à instância do bastion usando ssh. Substitua o valor da amostra pelo nome da sua chave SSH em `mykeypair.pem`.

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

Para criar um túnel usando PuTTY

1. Abra a página Instâncias no console do Amazon EC2.

2. Escolha uma instância.
3. Copie o endereço em DNS IPv4 pública. Por exemplo, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Abra o PuTTY, e selecione Sessão.
5. Insira o nome do host em Nome do host como ec2-user@*YOUR_PUBLIC_IPV4_DNS* e a porta como 22.
6. Expanda a guia SSH e selecione Auth. Em Arquivo de chave privada para autenticação, escolha seu arquivo “ppk” local.
7. Em SSH, escolha a guia Túneis e selecione as opções Dinâmico e Automático.
8. Em Porta de origem adicione a porta 8157 (ou qualquer outra porta não usada) e deixe a porta Destino em branco. Escolha Adicionar.
9. Escolha a guia Sessão e insira o nome da sessão. Por exemplo, `SSH Tunnel`.
10. Escolha Salvar, Abrir.

> **Note**
>
> Talvez seja necessário inserir uma frase secreta para sua chave pública.

> **Note**
>
> Se você receber um `Permission denied (publickey)` erro, recomendamos usar a ferramenta AWSSupport-TroubleshootSSH e escolher Executar esta automação (console) para solucionar problemas na configuração do SSH.

Etapa 3: configurar o grupo de segurança bastion como uma regra de entrada

O acesso aos servidores e o acesso regular à Internet a partir dos servidores são permitidos com um grupo de segurança especial de manutenção conectado a esses servidores. As etapas a seguir descrevem como configurar o grupo de segurança bastion como uma fonte de tráfego de entrada para o grupo de segurança VPC de um ambiente.

1. Abra a página Ambientes no console do Amazon MWAA.

2. Escolha um ambiente.
3. No painel Rede, escolha Grupo de segurança VPC.
4. Escolha Editar regras de entrada.
5. Escolha Adicionar regra.
6. Escolha sua ID do grupo de segurança da VPC na lista suspensa Fonte.
7. Deixe as opções restantes em branco ou defina-as com seus valores padrão.
8. Escolha Salvar regras.

Etapa 4: copiar o URL do Apache Airflow

As etapas a seguir descrevem como abrir o console do Amazon MWAA e copiar o URL para a IU do Apache Airflow.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Copie o URL na IU do Airflow para as etapas subsequentes.

Etapa 5: definir as configurações de proxy

Se você usar um túnel SSH com o encaminhamento de portas dinâmicas, deverá usar um complemento de gerenciamento de proxy SOCKS para controlar as configurações de proxy no seu navegador. Por exemplo, você pode usar o `--proxy-server` recurso do Chromium para iniciar uma sessão do navegador ou usar a FoxyProxy extensão no navegador Mozilla FireFox .

Opção um: configure um túnel SSH usando o encaminhamento de portas locais

Se não quiser usar um proxy SOCKS, é possível configurar um túnel SSH usando o encaminhamento de portas locais. O comando de exemplo a seguir acessa a interface web do Amazon ResourceManagerEC2 encaminhando o tráfego na porta local 8157.

1. Abra uma nova janela do prompt de comando.
2. Digite o seguinte comando para abrir um túnel SSH.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-
vpce.YOUR_REGION.airflow.amazonaws.com:443
 ubuntu@YOUR_PUBLIC_IPV4_DNS.YOUR_REGION.compute.amazonaws.com
```

`-L` significa o uso do encaminhamento de portas locais, que permite especificar uma porta local usada para encaminhar dados à porta remota identificada no servidor Web local do nó principal.

3. Digite `http://localhost:8157/` em seu navegador.

> **Note**
>
> Talvez seja necessário usar `https://localhost:8157/`.

Opção dois: proxies via linha de comando

A maioria dos navegadores da Web permite que você configure proxies por meio de uma linha de comando ou parâmetro de configuração. Por exemplo, com o Chromium, é possível iniciar o navegador com o seguinte comando:

```
chromium --proxy-server="socks5://localhost:8157"
```

Ele inicia uma sessão do navegador que usa o túnel ssh que você criou nas etapas anteriores para proxy de suas solicitações. É possível abrir sua URL privada do ambiente Amazon MWAA (com https://) da seguinte forma:

```
https://YOUR_VPC_ENDPOINT_ID-vpce.YOUR_REGION.airflow.amazonaws.com/home.
```

Opção três: uso de proxies FoxyProxy para o Mozilla Firefox

O exemplo a seguir demonstra uma configuração FoxyProxy padrão (versão 7.5.1) para o Mozilla Firefox. FoxyProxy fornece um conjunto de ferramentas de gerenciamento de proxy. Permite que seja usado um servidor proxy para URLs que corresponda aos padrões correspondentes aos domínios usados pela IU do Apache Airflow.

1. No Firefox, abra a página de extensão FoxyProxy padrão.
2. Escolha Adicionar ao Firefox.
3. Escolha Adicionar.
4. Escolha o FoxyProxy ícone na barra de ferramentas do seu navegador e escolha Opções.
5. Copie o código a seguir e salve localmente como `mwaa-proxy.json`. Substitua o exemplo do valor em *YOUR_HOST_NAME* pelo seu URL do Apache Airflow.

```
{
  "e0b7kh1606694837384": {
    "type": 3,
    "color": "#66cc66",
    "title": "airflow",
    "active": true,
    "address": "localhost",
    "port": 8157,
    "proxyDNS": false,
    "username": "",
    "password": "",
    "whitePatterns": [
      {
        "title": "airflow-ui",
        "pattern": "YOUR_HOST_NAME",
        "type": 1,
        "protocols": 1,
        "active": true
      }
    ],
    "blackPatterns": [],
    "pacURL": "",
    "index": -1
  },
  "k20d21508277536715": {
    "active": true,
    "title": "Default",
    "notes": "These are the settings that are used when no patterns match a URL.",
    "color": "#0055E5",
    "type": 5,
    "whitePatterns": [
      {
        "title": "all URLs",
        "active": true,
        "pattern": "*",
        "type": 1,
        "protocols": 1
      }
    ],
    "blackPatterns": [],
    "index": 9007199254740991
  },
  "logging": {
```

```
    "active": true,
    "maxSize": 500
  },
  "mode": "patterns",
  "browserVersion": "82.0.3",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

6. No painel Importar configurações do painel FoxyProxy 6.0+, escolha Importar configurações e selecione o arquivo. `mwaa-proxy.json`
7. Escolha OK.

Etapa 6: abra a IU do Apache Airflow

As etapas a seguir descrevem como abrir sua IU do Apache Airflow.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha Abrir a IU do Airflow.

Próximas etapas

- Saiba como executar comandos CLI do Airflow em um túnel SSH para um bastion host em Referência de comandos da CLI do Apache Airflow.
- Aprenda a fazer upload de um código DAG para seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Tutorial: Restringir o acesso de um usuário do Amazon MWAA a um subconjunto de DAGs

O Amazon MWAA gerencia o acesso ao seu ambiente mapeando suas entidades principais do IAM para um ou mais perfis padrão do Apache Airflow. O tutorial a seguir mostra como é possível restringir usuários individuais do Amazon MWAA a visualizar e interagir somente com um DAG específico ou com um conjunto de DAGs.

> **Note**
>
> As etapas deste tutorial podem ser concluídas usando o acesso federado, desde que as perfis do IAM possam ser assumidas.

Tópicos

Pré-requisitos

Para concluir as etapas deste tutorial, você precisará do seguinte:

- Um ambiente Amazon MWAA com vários DAGs
- Um diretor do IAM, `Admin` com AdministratorAccesspermissões, e um usuário do IAM`MWAAUser`, como principal para o qual você pode limitar o acesso ao DAG. Para obter mais informações sobre funções de administrador, consulte Função de trabalho de administrador no Guia do usuário do IAM

  > **Note**
  >
  > Não anexe políticas de permissão diretamente aos seus usuários do IAM. Recomendamos configurar perfis do IAM que os usuários possam assumir para obter acesso temporário aos seus recursos do Amazon MWAA.

- AWS Command Line Interface versão 2 instalada.

Etapa 1: forneça acesso ao servidor web Amazon MWAA a sua entidade principal do IAM com o perfil padrão **`Public`** do Apache Airflow.

Para conceder permissão usando o AWS Management Console

1. Faça login na sua AWS conta com uma `Admin` função e abra o console do IAM.
2. No painel de navegação à esquerda, selecione Usuários e escolha o usuário do Amazon MWAA IAM.
3. Na página de detalhes do usuário, em Resumo, escolha a guia Permissões e, em seguida, escolha Políticas de permissões para expandir o cartão e escolha Adicionar permissões.
4. Na seção Definir permissões, escolha Anexar políticas existentes diretamente e escolha Criar política.
5. Na página Criar política, escolha JSON e, em seguida, copie e cole a seguinte política de permissões JSON no editor de políticas. Essa política concede acesso do servidor web ao usuário com o perfil padrão `Public` do Apache Airflow.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": "airflow:CreateWebLoginToken",
        "Resource": [

 "arn:aws:airflow:YOUR_REGION:YOUR_ACCOUNT_ID:role/YOUR_ENVIRONMENT_NAME/Public"
            ]
        }
    ]
 }
```

Etapa dois: criar um novo perfil personalizado do Apache Airflow

Para criar um novo perfil usando a IU do Apache Airflow

1. Usando sua função de administrador do IAM, abra o console do Amazon MWAA e inicie a interface de usuário do Apache Airflow do seu ambiente.

2. No painel de navegação na parte superior, passe o mouse sobre Segurança para abrir a lista suspensa e escolha Listar funções para visualizar as funções padrão do Apache Airflow.
3. Na lista de funções, selecione Usuário e, na parte superior da página, escolha Ações para abrir o menu suspenso. Escolha Copiar função e confirme Ok

> **Note**
>
> Copie os perfis Ops ou Viewer para conceder mais ou menos acesso, respectivamente.

4. Localize a nova função que você criou na tabela e escolha Editar registro.
5. Na página Switch Role, faça o seguinte:

   - Em Nome, digite um novo nome para a função no campo de texto. Por exemplo, **`Restricted`**.
   - Para obter a lista de permissões, remova `can read on DAGs` e `can edit on DAGs`, em seguida, adicione permissões de leitura e gravação para o conjunto de DAGs ao qual você deseja fornecer acesso. Por exemplo, para um DAG, `example_dag.py`, adicione **`can read on DAG:`*`example_dag`*** e **`can edit on DAG:`*`example_dag`***.

   Escolha Save (Salvar). Agora você deve ter um novo perfil que limita o acesso a um subconjunto de DAGs disponíveis em seu ambiente Amazon MWAA. Agora é possível atribuir esse perfil a qualquer usuário existente do Apache Airflow.

Etapa três: atribuir o perfil que você criou ao seu usuário do Amazon MWAA

Para atribuir o novo perfil

1. Usando as credenciais de acesso para `MWAAUser`, execute o comando da CLI a seguir para recuperar o URL do servidor da web do ambiente.

```
$ aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq
 '.Environment.WebserverUrl'
```

   Se o teste for bem-sucedido, você verá o seguinte resultado:

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. Com `MWAAUser` login no AWS Management Console, abra uma nova janela do navegador e acesse o seguinte URL. Substitua `Webserver-URL` por suas informações.

   ```
   https://<Webserver-URL>/home
   ```

   Se for bem-sucedido, você verá uma página de erro `Forbidden` porque o `MWAAUser` ainda não recebeu permissão para acessar a IU do Apache Airflow.

3. Depois de fazer `Admin` login no AWS Management Console, abra o console do Amazon MWAA novamente e inicie a interface de usuário do Apache Airflow do seu ambiente.

4. No painel da interface do usuário, expanda a lista suspensa Segurança e, desta vez, escolha Listar usuários.

5. Na tabela de usuários, encontre o novo usuário do Apache Airflow e escolha Editar registro. O primeiro nome do usuário corresponderá ao seu nome de usuário do IAM no seguinte padrão: `user/mwaa-user`.

6. Na página Editar usuário, na seção Função, adicione a nova função personalizada que você criou e escolha Salvar.

   > **Note**
   >
   > O campo Sobrenome é obrigatório, mas um espaço satisfaz o requisito.

   A entidade principal `Public` do IAM concede a permissão `MWAAUser` para acessar a IU do Apache Airflow, enquanto o novo perfil fornece as permissões adicionais necessárias para ver seus DAGs.

> **Important**
>
> Qualquer um dos 5 perfis padrão (como `Admin`) não autorizadas pelo IAM que são adicionados usando a IU do Apache Airflow será removido no próximo login do usuário.

Próximas etapas

- Para saber mais sobre como gerenciar o acesso ao seu ambiente Amazon MWAA e ver exemplos de política do IAM JSON que é possível usar para os usuários do seu ambiente, consulte the section called “Como acessar um ambiente do Amazon MWAA”

Recursos relacionados

- Controle de acesso (documentação do Apache Airflow): saiba mais sobre os perfis padrão do Apache Airflow no site de documentação do Apache Airflow.

Tutorial: Automatize o gerenciamento de seus próprios endpoints de ambiente no Amazon MWAA

Se você usa AWS Organizationspara gerenciar várias AWS contas que compartilham recursos, o Amazon MWAA permite que você crie e gerencie seus próprios endpoints Amazon VPC. Isso significa que você pode usar políticas de segurança mais rígidas que permitam acessar somente os recursos exigidos pelo seu ambiente.

Quando você cria um ambiente em uma Amazon VPC compartilhada, a conta que possui a Amazon VPC principal (proprietário) compartilha as duas sub-redes privadas exigidas pela Amazon MWAA com outras contas (participantes) que pertencem à mesma organização. As contas de participantes que compartilham essas sub-redes podem então visualizar, criar, modificar e excluir ambientes na VPC compartilhada.

Quando você cria um ambiente em uma Amazon VPC compartilhada ou de outra forma restrita por políticas, a Amazon MWAA primeiro cria os recursos de VPC do serviço e, em seguida, `PENDING`insere um estado por até 72 horas.

Quando o status do ambiente muda de `CREATING` para`PENDING`, o Amazon MWAA envia uma EventBridge notificação à Amazon sobre a mudança de estado. Isso permite que a conta do proprietário crie os endpoints necessários em nome dos participantes com base nas informações do serviço de endpoint do console ou da API do Amazon MWAA, ou programaticamente. A seguir, criamos novos endpoints do Amazon VPC usando uma função Lambda e uma EventBridge regra que escuta as notificações de mudança de estado do Amazon MWAA.

Aqui, criamos os novos endpoints na mesma Amazon VPC do ambiente. Para configurar uma Amazon VPC compartilhada, crie a EventBridge regra e a função Lambda na conta do proprietário e no ambiente Amazon MWAA na conta do participante.

Tópicos

Pré-requisitos

Para concluir as etapas deste tutorial, você precisará do seguinte:

- ...

Crie a Amazon VPC

Use o AWS CloudFormation modelo e o AWS CLI comando a seguir para criar uma nova Amazon VPC. O modelo configura os recursos da Amazon VPC e modifica a política do endpoint para restringir o acesso a uma fila específica.

1. Faça o download do AWS CloudFormation modelo e, em seguida, descompacte o `.yml` arquivo.
2. Em uma nova janela do prompt de comando, navegue até a pasta em que você salvou o modelo e use `create-stack`para criar a pilha. O `--template-body` sinalizador especifica o caminho para o modelo.

```
$ aws cloudformation create-stack --stack-name stack-name --template-body file://
cfn-vpc-private-network.yml
```

Na próxima seção, você criará a função Lambda.

Criar a função do Lambda

Use o código Python a seguir e a política JSON do IAM para criar uma nova função e função de execução do Lambda. Essa função cria endpoints Amazon VPC para um servidor web Apache Airflow privado e uma fila do Amazon SQS. O Amazon MWAA usa o Amazon SQS para enfileirar tarefas com o Celery entre vários trabalhadores ao escalar seu ambiente.

1. Baixe o código da função Python.
2. Faça o download da política de permissão do IAM e, em seguida, descompacte o arquivo.
3. Abra um prompt de comando e navegue até a pasta em que você salvou a política de permissão JSON. Use o `create-role`comando IAM para criar a nova função.

   ```
   $ aws iam create-role --role-name function-role \
   --assume-role-policy-document file://lambda-mwaa-vpce-policy.json
   ```

   Observe o ARN da função na AWS CLI resposta. Na próxima etapa, especificaremos essa nova função como a função de execução da função usando seu ARN.
4. Navegue até a pasta em que você salvou o código da função e use o `create-function`comando para criar uma nova função.

   ```
   $ aws lambda create-function --function-name mwaa-vpce-lambda \
   --zip-file file://mwaa-lambda-shared-vpc.zip --runtime python3.8 --role
    arn:aws:iam::123456789012:role/function-role --handler lambda_handler
   ```

   Observe a função ARN da AWS CLI resposta. Na próxima etapa, especificamos o ARN para configurar a função como destino para uma nova EventBridge regra.

Na próxima seção, você criará a EventBridge regra que invoca essa função quando o ambiente entra em um `PENDING` estado.

Crie a EventBridge regra

Faça o seguinte para criar uma nova regra que escute as notificações do Amazon MWAA e tenha como alvo sua nova função Lambda.

1. Use o EventBridge `put-rule` comando para criar uma nova EventBridge regra.

   ```
   $ aws events put-rule --name "mwaa-lambda-rule" \
   ```

```
--event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment
 Status Change\"]}"
```

O padrão de eventos escuta as notificações que o Amazon MWAA envia sempre que o status de um ambiente muda.

```
{
  "source": ["aws.airflow"],
  "detail-type": ["MWAA Environment Status Change"]
}
```

2. Use o `put-targets` comando para adicionar a função Lambda como destino para a nova regra.

```
$ aws events put-targets --rule "mwaa-lambda-rule" \
--targets "Id"="1","Arn"="arn:aws::lambda:region:123456789012:function:mwaa-vpce-
lambda"
```

Você está pronto para criar um novo ambiente Amazon MWAA com endpoints Amazon VPC gerenciados pelo cliente.

Crie o ambiente Amazon MWAA

Use o console do Amazon MWAA para criar um novo ambiente com endpoints Amazon VPC gerenciados pelo cliente.

1. Abra o console do Amazon MWAA e escolha Criar um ambiente.
2. Em Nome, insira um nome exclusivo.
3. Para a versão Airflow, escolha a versão mais recente.
4. Escolha um bucket do Amazon S3 e uma pasta DAGs, como **`dags/`** para usar com o ambiente, e escolha Avançar.
5. Na página Definir configurações avançadas, faça o seguinte:
   a. Para Virtual Private Cloud, escolha a Amazon VPC que você criou na etapa anterior.
   b. Para acesso ao servidor Web, escolha Rede pública (acessível pela Internet).
   c. Em Grupos de segurança, escolha o grupo de segurança com o qual você criouAWS CloudFormation. Como os grupos de segurança dos AWS PrivateLink endpoints da etapa

anterior são autorreferenciados, você deve escolher o mesmo grupo de segurança para seu ambiente.

d. Para gerenciamento de endpoints, escolha endpoints gerenciados pelo cliente.

6. Mantenha as configurações padrão restantes e escolha Avançar.
7. Revise suas seleções e escolha Criar ambiente.

> **Tip**
>
> Para obter mais informações sobre a configuração de um novo ambiente, consulte Introdução ao Amazon MWAA.

Quando o ambiente está`PENDING`, o Amazon MWAA envia uma notificação que corresponde ao padrão de evento que você definiu para sua regra. A regra invoca sua função Lambda. A função analisa o evento de notificação e obtém as informações de endpoint necessárias para o servidor web e a fila do Amazon SQS. Em seguida, ele cria os endpoints em sua Amazon VPC.

Quando os endpoints estão disponíveis, o Amazon MWAA retoma a criação do seu ambiente. Quando estiver pronto, o status do ambiente muda para `AVAILABLE` e você pode acessar o servidor web Apache Airflow usando o console Amazon MWAA.

Exemplos de código para o Amazon Managed Workflows for Apache Airflow

Este guia contém exemplos de código, incluindo DAGs e plug-ins personalizados, que você pode usar em um ambiente do Amazon Managed Workflows for Apache Airflow. Para obter mais exemplos de uso do Apache Airflow com AWS serviços, consulte o `dags`diretório no repositório do Apache Airflow. GitHub

Amostras

- Como usar um DAG para importar variáveis na CLI
- Como criar uma conexão SSH usando o SSHOperator
- Como usar uma chave secreta em AWS Secrets Manager para uma conexão Snowflake do Apache Airflow
- Como usar um DAG para gravar métricas personalizadas no CloudWatch
- Limpeza do banco de dados do Aurora PostgreSQL em um ambiente do Amazon MWAA
- Como exportar metadados do ambiente para arquivos CSV no Amazon S3
- Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow
- Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow
- Como criar um plugin personalizado com a Oracle
- Criar um plug-in personalizado que gera variáveis de ambiente de runtime
- Como alterar o fuso horário de um DAG no Amazon MWAA
- Como atualizar um token CodeArtifact
- Criação de um plug-in personalizado com o Apache Hive e o Hadoop
- Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow
- Como invocar DAGs com uma função do Lambda
- Como invocar DAGs em diferentes ambientes do Amazon MWAA
- Como usar o Amazon MWAA com Amazon RDS para Microsoft SQL Server
- Como usar o Amazon MWAA com o Amazon EMR
- Como usar o Amazon MWAA com o Amazon EKS
- Como conectar-se ao Amazon ECS usando o ECSOperator

- Como usar DBT com o Amazon MWAA
- AWS blogs e tutoriais

Como usar um DAG para importar variáveis na CLI

O código de exemplo a seguir importa variáveis usando a CLI no Amazon Managed Workflows for Apache Airflow.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Permissões

Sua conta da AWS precisa ter acesso à política `AmazonMWAAAirflowCliAccess`. Para saber mais, consulte Política de CLI do Apache Airflow: Acesso ao AmazonMWAA AirflowCli.

Dependências

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

O código de exemplo a seguir usa três entradas: o nome do seu ambiente Amazon MWAA (em `mwaa_env`), a Região AWS do seu ambiente (em `aws_region`) e o arquivo local que contém as variáveis que você deseja importar (em `var_file`).

```
import boto3
import json
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwaa_env=''
aws_region=''
var_file=''

try:
    opts, args = getopt.getopt(argv, 'e:v:r:', ['environment', 'variable-
file','region'])
    #if len(opts) == 0 and len(opts) > 3:
    if len(opts) != 3:
        print ('Usage: -e MWAA environment -v variable file location and filename -r
 aws region')
    else:
        for opt, arg in opts:
            if opt in ("-e"):
                mwaa_env=arg
            elif opt in ("-r"):
                aws_region=arg
            elif opt in ("-v"):
                var_file=arg

        boto3.setup_default_session(region_name="{}".format(aws_region))
        mwaa_env_name = "{}".format(mwaa_env)

        client = boto3.client('mwaa')
        mwaa_cli_token = client.create_cli_token(
            Name=mwaa_env_name
        )

        with open ("{}".format(var_file), "r") as myfile:
```

```
            fileconf = myfile.read().replace('\n', '')

        json_dictionary = json.loads(fileconf)
        for key in json_dictionary:
            print(key, " ", json_dictionary[key])
            val = (key + " " + json_dictionary[key])
            mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
            mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
            raw_data = "variables set {0}".format(val)
            mwaa_response = requests.post(
                mwaa_webserver_hostname,
                headers={
                    'Authorization': mwaa_auth_token,
                    'Content-Type': 'text/plain'
                    },
                data=raw_data
                )
            mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
            mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')
            print(mwaa_response.status_code)
            print(mwaa_std_err_message)
            print(mwaa_std_out_message)

except:
    print('Use this script with the following options: -e MWAA environment -v variable
 file location and filename -r aws region')
    print("Unexpected error:", sys.exc_info()[0])
    sys.exit(2)
```

Próximas etapas

- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Como criar uma conexão SSH usando o **`SSHOperator`**

O exemplo a seguir descreve como é possível usar `SSHOperator` em um gráfico acíclico direcionado (DAG) para se conectar a uma instância remota do Amazon EC2 a partir do seu

ambiente do Amazon Managed Workflows for Apache Airflow. É possível usar uma abordagem semelhante para se conectar a qualquer instância remota com acesso SSH.

No exemplo a seguir, você faz upload de uma chave secreta SSH (`.pem`) para o diretório `dags` do seu ambiente no Amazon S3. Em seguida, você instala as dependências necessárias usando `requirements.txt` e cria uma nova conexão do Apache Airflow na interface do usuário. Por fim, você grava um DAG que cria uma conexão SSH com a instância remota.

Tópicos

Version (Versão)

- É possível usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.
- Uma chave secreta SSH. A amostra de código pressupõe que você tenha uma instância do Amazon EC2 e uma `.pem` na mesma região do seu ambiente Amazon MWAA. Se você não tiver uma chave, consulte Criar ou importar um par de chaves no Guia do usuário do Amazon EC2.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Requisitos

Adicione o parâmetro a seguir para `requirements.txt` para instalar o pacote `apache-airflow-providers-ssh` no servidor web. Depois que o ambiente for atualizado e o Amazon MWAA instalar com sucesso a dependência, você verá um novo tipo de conexão SSH na interface do usuário.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
apache-airflow-providers-ssh
```

> **Note**
>
> `-c` define o URL das restrições em `requirements.txt`. Isso garante que o Amazon MAA instale a versão correta do pacote para seu ambiente.

Copie sua chave secreta para o Amazon S3

Use o AWS Command Line Interface comando a seguir para copiar sua `.pem` chave para o `dags` diretório do seu ambiente no Amazon S3.

```
$ aws s3 cp your-secret-key.pem s3://your-bucket/dags/
```

O Amazon MWAA copia o conteúdo em `dags`, incluindo a chave `.pem`, para o diretório local `/usr/local/airflow/dags/`. Ao fazer isso, o Apache Airflow pode acessar a chave.

Crie uma nova conexão com o Apache Airflow

Para criar uma nova conexão SSH usando a IU do Apache Airflow

1. Abra a página Ambientes no console do Amazon MWAA.
2. Na lista de ambientes, escolha Abrir a IU do Airflow em seu ambiente.
3. Na página IU do Apache Airflow, selecione Admin na barra de navegação superior para expandir a lista suspensa e selecione Conexões.
4. Na página Listar conexões, escolha o botão + ou Adicionar um novo registro para adicionar uma nova conexão.
5. Na página Adicionar conexão, forneça as seguintes informações:

a. Em Id da conexão, insira **`ssh_new`**.

b. Em Tipo de conexão, selecione SSH na lista suspensa.

> **Note**
>
> Se o tipo de conexão SSH não estiver disponível na lista, o Amazon MWAA não instalou o pacote de `apache-airflow-providers-ssh` necessário. Atualize seu arquivo `requirements.txt` para incluir esse pacote e tente novamente.

c. Para Host, insira o endereço IP para a instância do Amazon EC2 à qual você deseja se conectar. Por exemplo, **`12.345.67.89`**.

d. Em Nome de usuário, insira **`ec2-user`** se você está se conectando a uma instância do Amazon EC2. Seu nome de usuário pode ser diferente, dependendo do tipo de instância remota à qual você deseja que o Apache Airflow se conecte.

e. Para Extra, insira o seguinte par de chave-valor no formato JSON:

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

Esse par de chave-valor instrui o Apache Airflow a procurar a chave secreta no diretório local `/dags`.

Exemplo de código

O DAG a seguir usa `SSHOperator` para se conectar à sua instância de destino do Amazon EC2 e, em seguida, `hostname` executa o comando Linux para imprimir o nome da instância. É possível modificar o DAG para executar qualquer comando ou script na instância remota.

1. Abra um terminal e navegue até o diretório em que seu código DAG está armazenado. Por exemplo: .

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `ssh.py`.

```
from airflow.decorators import dag
from datetime import datetime
from airflow.providers.ssh.operators.ssh import SSHOperator
```

```
@dag(
    dag_id="ssh_operator_example",
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    catchup=False,
    )
def ssh_dag():
    task_1=SSHOperator(
        task_id="ssh_task",
        ssh_conn_id='ssh_new',
        command='hostname',
    )

my_ssh_dag = ssh_dag()
```

3. Execute o AWS CLI comando a seguir para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a interface do usuário do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se tiver êxito, você verá uma saída semelhante à seguinte nos logs de tarefas para `ssh_task` no DAG `ssh_operator_example`:

```
[2022-01-01, 12:00:00 UTC] {{base.py:79}} INFO - Using connection to: id: ssh_new.
 Host: 12.345.67.89, Port: None,
Schema: , Login: ec2-user, Password: None, extra: {'key_file': '/usr/local/airflow/
dags/your-secret-key.pem'}
[2022-01-01, 12:00:00 UTC] {{ssh.py:264}} WARNING - Remote Identification Change is
 not verified. This won't protect against Man-In-The-Middle attacks
[2022-01-01, 12:00:00 UTC] {{ssh.py:270}} WARNING - No Host Key Verification. This
 won't protect against Man-In-The-Middle attacks
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Connected (version 2.0,
 client OpenSSH_7.4)
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Authentication (publickey)
 successful!
[2022-01-01, 12:00:00 UTC] {{ssh.py:139}} INFO - Running command: hostname
[2022-01-01, 12:00:00 UTC]{{ssh.py:171}} INFO - ip-123-45-67-89.us-
west-2.compute.internal
[2022-01-01, 12:00:00 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=ssh_operator_example, task_id=ssh_task, execution_date=20220712T200914,
 start_date=20220712T200915, end_date=20220712T200916
```

Como usar uma chave secreta em AWS Secrets Manager para uma conexão Snowflake do Apache Airflow

O exemplo a seguir chama AWS Secrets Manager para a obtenção de uma chave secreta para uma conexão Snowflake do Apache Airflow no Amazon Managed Workflows for Apache Airflow. Pressupõe-se que você tenha concluído as etapas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- O back-end do Secrets Manager como uma opção de configuração do Apache Airflow, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.
- Uma string de conexão do Apache Airflow no Secrets Manager, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Permissões

- As permissões do Secrets Manager são mostradas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Requisitos

Para usar o código de amostra nesta página, adicione as seguintes dependências ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python.

```
apache-airflow-providers-snowflake==1.3.0
```

Exemplo de código

As etapas a seguir descrevem como criar o código DAG que chama o Secrets Manager para obter o segredo.

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

   ```
   cd dags
   ```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `snowflake_connection.py`.

   ```
   """
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

   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 airflow import DAG
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
from airflow.utils.dates import days_ago

snowflake_query = [
    """use warehouse "MY_WAREHOUSE";""",
    """select * from "SNOWFLAKE_SAMPLE_DATA"."WEATHER"."WEATHER_14_TOTAL" limit
 100;""",
]

with DAG(dag_id='snowflake_test', schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    snowflake_select = SnowflakeOperator(
        task_id="snowflake_select",
        sql=snowflake_query,
        snowflake_conn_id="snowflake_conn",
    )
```

Próximas etapas

- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Como usar um DAG para gravar métricas personalizadas no CloudWatch

Você pode usar o exemplo de código a seguir para gravar um gráfico acíclico direcionado (DAG) que executa `PythonOperator` para recuperar métricas em nível de SO para um ambiente Amazon MWAA. O DAG, então, publica os dados como métricas personalizadas para o Amazon CloudWatch.

As métricas personalizadas de nível de SO fornecem visibilidade adicional sobre como os operadores do ambiente estão utilizando recursos como memória virtual e CPU. Você pode usar essas informações para selecionar a classe de ambiente mais adequada à sua workload.

Tópicos

- Versão
- Pré-requisitos

- Permissões
- Dependências
- Exemplo de código

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de exemplo nesta página, você precisará de:

- Um ambiente Amazon MWAA.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Dependências

- Nenhuma dependência adicional é necessária para usar o exemplo de código desta página.

Exemplo de código

1. No prompt de comando, navegue até a pasta em que seu código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo do exemplo de código a seguir e salve-o localmente como `dag-custom-metrics.py`. Substitua `MWAA-ENV-NAME` pelo nome do seu ambiente.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
```

```
from datetime import datetime
import os,json,boto3,psutil,socket

def publish_metric(client,name,value,cat,unit='None'):
    environment_name = os.getenv("MWAA_ENV_NAME")
    value_number=float(value)
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    print('writing value',value_number,'to metric',name)
    response = client.put_metric_data(
        Namespace='MWAA-Custom',
        MetricData=[
            {
                'MetricName': name,
                'Dimensions': [
                    {
                        'Name': 'Environment',
                        'Value': environment_name
                    },
                    {
                        'Name': 'Category',
                        'Value': cat
                    },
                    {
                        'Name': 'Host',
                        'Value': ip_address
                    },
                ],
                'Timestamp': datetime.now(),
                'Value': value_number,
                'Unit': unit
            },
        ]
    )
    print(response)
    return response

def python_fn(**kwargs):
    client = boto3.client('cloudwatch')

    cpu_stats = psutil.cpu_stats()
    print('cpu_stats', cpu_stats)

    virtual = psutil.virtual_memory()
```

```
    cpu_times_percent = psutil.cpu_times_percent(interval=0)

    publish_metric(client=client, name='virtual_memory_total',
 cat='virtual_memory', value=virtual.total, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_available',
 cat='virtual_memory', value=virtual.available, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_used', cat='virtual_memory',
 value=virtual.used, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_free', cat='virtual_memory',
 value=virtual.free, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_active',
 cat='virtual_memory', value=virtual.active, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_inactive',
 cat='virtual_memory', value=virtual.inactive, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_percent',
 cat='virtual_memory', value=virtual.percent, unit='Percent')

    publish_metric(client=client, name='cpu_times_percent_user',
 cat='cpu_times_percent', value=cpu_times_percent.user, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_system',
 cat='cpu_times_percent', value=cpu_times_percent.system, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_idle',
 cat='cpu_times_percent', value=cpu_times_percent.idle, unit='Percent')

    return "OK"


with DAG(dag_id=os.path.basename(__file__).replace(".py", ""),
 schedule_interval='*/5 * * * *', catchup=False, start_date=days_ago(1)) as dag:
    t = PythonOperator(task_id="memory_test", python_callable=python_fn,
 provide_context=True)
```

3. Execute o seguinte comando AWS CLI para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a IU do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se o DAG for executado com sucesso, você deverá ver algo semelhante em seus logs do Apache Airflow:

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO -
 cpu_stats scpustats(ctx_switches=3253992384, interrupts=1964237163,
 soft_interrupts=492328209, syscalls=0)
```

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 16024199168.0 to metric virtual_memory_total
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 14356287488.0 to metric virtual_memory_available
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': '6ef60085-07ab-4865-8abf-dc94f90cab46', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': '6ef60085-07ab-4865-8abf-dc94f90cab46',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 1342296064.0 to metric virtual_memory_used
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
...
[2022-08-16, 10:54:46 UTC] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-08-16, 10:54:46 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=dag-custom-metrics, task_id=memory_test, execution_date=20220816T175444,
 start_date=20220816T175445, end_date=20220816T175446
[2022-08-16, 10:54:46 UTC] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
```

Limpeza do banco de dados do Aurora PostgreSQL em um ambiente do Amazon MWAA

O Amazon Managed Workflows para Apache Airflow usa um banco de dados Aurora PostgreSQL como banco de dados de metadados do Apache Airflow, onde o DAG é executado e as instâncias das tarefas são armazenadas. O código de exemplo a seguir limpa periodicamente as entradas do banco de dados Aurora PostgreSQL dedicado para seu ambiente Amazon MWAA.

Tópicos

Version (Versão)

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Dependências

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

O DAG a seguir limpa o banco de dados de metadados das tabelas especificadas em. `TABLES_TO_CLEAN` O exemplo exclui dados das tabelas especificadas nos últimos sete dias. Para ajustar até que ponto as entradas são excluídas, `MAX_AGE_IN_DAYS` defina um valor diferente.

Apache Airflow v2

```
from airflow import settings
from airflow.utils.dates import days_ago
from airflow.models import DagTag, DagModel, DagRun, ImportError, Log, SlaMiss,
 RenderedTaskInstanceFields, TaskInstance, TaskReschedule, XCom
from airflow.decorators import dag, task
from airflow.utils.dates import days_ago
from time import sleep

from airflow.version import version
major_version, minor_version = int(version.split('.')[0]), int(version.split('.')
[1])
```

```
if major_version >= 2 and minor_version >= 6:
    from airflow.jobs.job import Job
else:
    # The BaseJob class was renamed as of Apache Airflow v2.6
    from airflow.jobs.base_job import BaseJob as Job

# Delete entries for the past seven days. Adjust MAX_AGE_IN_DAYS to set how far back
 this DAG cleans the database.
MAX_AGE_IN_DAYS = 7
MIN_AGE_IN_DAYS = 0
DECREMENT = -7

# This is a list of (table, time) tuples.
# table = the table to clean in the metadata database
# time  = the column in the table associated to the timestamp of an entry
#         or None if not applicable.
TABLES_TO_CLEAN = [[Job, Job.latest_heartbeat],
    [TaskInstance, TaskInstance.execution_date],
    [TaskReschedule, TaskReschedule.execution_date],
    [DagTag, None],
    [DagModel, DagModel.last_parsed_time],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [XCom, XCom.execution_date],
]

@task()
def cleanup_db_fn(x):
    session = settings.Session()

    if x[1]:
        for oldest_days_ago in range(MAX_AGE_IN_DAYS, MIN_AGE_IN_DAYS, DECREMENT):
            earliest_days_ago = max(oldest_days_ago + DECREMENT, MIN_AGE_IN_DAYS)
            print(f"deleting {str(x[0])} entries between {earliest_days_ago} and
 {oldest_days_ago} days old...")
            earliest_date = days_ago(earliest_days_ago)
            oldest_date = days_ago(oldest_days_ago)
            query = session.query(x[0]).filter(x[1] >= oldest_date).filter(x[1] <=
 earliest_date)
            query.delete(synchronize_session= False)
            session.commit()
```

```
            sleep(5)
    else:
        # No time column specified for the table. Delete all entries
        print("deleting", str(x[0]), "...")
        query = session.query(x[0])
        query.delete(synchronize_session= False)
        session.commit()

    session.close()


@dag(
    dag_id="cleanup_db",
    schedule_interval="@weekly",
    start_date=days_ago(7),
    catchup=False,
    is_paused_upon_creation=False
)

def clean_db_dag_fn():
    t_last=None
    for x in TABLES_TO_CLEAN:
        t=cleanup_db_fn(x)
        if t_last:
            t_last >> t
        t_last = t

clean_db_dag = clean_db_dag_fn()
```

Como exportar metadados do ambiente para arquivos CSV no Amazon S3

O exemplo de código a seguir mostra como você pode criar um gráfico acíclico direcionado (DAG) que consulta o banco de dados em busca de uma variedade de informações de execução do DAG e grava os dados para arquivos `.csv` armazenados no Amazon S3.

Talvez você queira exportar informações do banco de dados Aurora PostgreSQL do seu ambiente para inspecionar os dados localmente, arquivá-los no armazenamento de objetos ou combiná-los com ferramentas como o operador Amazon S3 para Amazon Redshift e a limpeza do banco de

dados, a fim de mover os metadados do Amazon MWAA do ambiente, mas preservá-los para análise futura.

É possível consultar o banco de dados para qualquer um dos objetos listados nos Modelos do Apache Airflow. Esse exemplo de código usa três modelos, `DagRun`, `TaskFail` e `TaskInstance`, que fornecem informações relevantes para executar o DAG.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.
- Um novo bucket do Amazon S3 no qual você deseja exportar suas informações de metadados.

Permissões

O Amazon MWAA precisa de permissão para que a ação `s3:PutObject` do Amazon S3 grave as informações de metadados consultadas no Amazon S3. Adicione a seguinte declaração de política ao perfil de execução do seu ambiente.

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::your-new-export-bucket"
```

```
}
```

Essa política limita o acesso de gravação somente ao *your-new-export-bucket*.

Requisitos

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

As etapas a seguir descrevem como você pode criar um DAG que consulta o Aurora PostgreSQL e grava o resultado em seu novo bucket do Amazon S3.

1. Em seu terminal, navegue até o diretório em que seu código DAG está armazenado. Por exemplo:

   ```
   cd dags
   ```

2. Copie o conteúdo do exemplo de código a seguir e salve-o localmente como `metadata_to_csv.py`. Você pode alterar o valor atribuído para `MAX_AGE_IN_DAYS` para controlar a idade dos registros mais antigos que seu DAG consulta no banco de dados de metadados.

   ```
   from airflow.decorators import dag, task
   from airflow import settings
   import os
   import boto3
   from airflow.utils.dates import days_ago
   from airflow.models import DagRun, TaskFail, TaskInstance
   import csv, re
   from io import StringIO

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   MAX_AGE_IN_DAYS = 30
   S3_BUCKET = '<your-export-bucket>'
   S3_KEY = 'files/export/{0}.csv'

   # You can add other objects to export from the metadatabase,
   OBJECTS_TO_EXPORT = [
   ```

```
    [DagRun,DagRun.execution_date],
    [TaskFail,TaskFail.execution_date],
    [TaskInstance, TaskInstance.execution_date],
]

@task()
def export_db_task(**kwargs):
    session = settings.Session()
    print("session: ",str(session))

    oldest_date = days_ago(MAX_AGE_IN_DAYS)
    print("oldest_date: ",oldest_date)

    s3 = boto3.client('s3')

    for x in OBJECTS_TO_EXPORT:
        query = session.query(x[0]).filter(x[1] >= days_ago(MAX_AGE_IN_DAYS))
        print("type",type(query))
        allrows=query.all()
        name=re.sub("[<>']", "", str(x[0]))
        print(name,": ",str(allrows))

        if len(allrows) > 0:
            outfileStr=""
            f = StringIO(outfileStr)
            w = csv.DictWriter(f, vars(allrows[0]).keys())
            w.writeheader()
            for y in allrows:
                w.writerow(vars(y))
            outkey = S3_KEY.format(name[6:])
            s3.put_object(Bucket=S3_BUCKET, Key=outkey, Body=f.getvalue())

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=days_ago(1),
    )
def export_db():
    t = export_db_task()

metadb_to_s3_test = export_db()
```

3. Execute o seguinte comando AWS CLI para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a IU do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se for bem-sucedido, você exibirá uma saída semelhante à seguinte nos logs de tarefas da tarefa `export_db`:

```
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.dagrun.DagRun : [your-tasks]
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskfail.TaskFail :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskinstance.TaskInstance :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=metadb_to_s3, task_id=export_db, execution_date=20220101T000000,
 start_date=20220101T000000, end_date=20220101T000000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

Agora você pode acessar e baixar os arquivos `.csv` exportados em seu novo bucket do Amazon S3 em `/files/export/`.

Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow

O exemplo a seguir chama AWS Secrets Manager para obter uma chave secreta para uma variável Apache Airflow no Amazon Managed Workflows for Apache Airflow. Pressupõe-se que você tenha concluído as etapas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.
- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- O back-end do Secrets Manager como uma opção de configuração do Apache Airflow, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.
- Uma string variável do Apache Airflow no Secrets Manager, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Permissões

- As permissões do Secrets Manager são mostradas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Requisitos

- Para usar esse exemplo de código com o Apache Airflow v1, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v1 em seu ambiente.

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

As etapas a seguir descrevem como criar o código DAG que chama o Secrets Manager para obter o segredo.

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `secrets-manager-var.py`.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.models import Variable
from airflow.utils.dates import days_ago
from datetime import timedelta
import os
DAG_ID = os.path.basename(__file__).replace(".py", "")
DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}
def get_variable_fn(**kwargs):
    my_variable_name = Variable.get("test-variable", default_var="undefined")
    print("my_variable_name: ", my_variable_name)
    return my_variable_name
with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['variable']
```

```
) as dag:
    get_variable = PythonOperator(
        task_id="get_variable",
        python_callable=get_variable_fn,
        provide_context=True
    )
```

Próximas etapas

- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow

O exemplo a seguir chama AWS Secrets Manager para obter uma chave secreta para uma conexão Apache Airflow no Amazon Managed Workflows for Apache Airflow. Pressupõe-se que você tenha concluído as etapas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- O back-end do Secrets Manager como uma opção de configuração do Apache Airflow, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.
- Uma string de conexão do Apache Airflow no Secrets Manager, conforme mostrado em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Permissões

- As permissões do Secrets Manager são mostradas em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager.

Requisitos

- Para usar esse exemplo de código com o Apache Airflow v1, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v1 em seu ambiente.

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

As etapas a seguir descrevem como criar o código DAG que chama o Secrets Manager para obter o segredo.

Apache Airflow v2

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

   ```
   cd dags
   ```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `secrets-manager.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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 airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]
```

```
    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Apache Airflow v1

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `secrets-manager.py`.

```
from airflow import DAG, settings, secrets
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from airflow.contrib.hooks.aws_hook import AwsHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
```

```
    'depends_on_past': False
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsHook()
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Próximas etapas

- Saiba como fazer o upload do código DAG neste exemplo para a `dags` pasta em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.

Como criar um plugin personalizado com a Oracle

O exemplo a seguir mostra as etapas para criar um plug-in personalizado usando a Oracle para Amazon MWAA e pode ser combinado com outros plug-ins e binários personalizados em seu arquivo plugins.zip.

Sumário

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.
- O registro de logs operador é habilitado em qualquer nível de log, `CRITICAL` ou superior, para seu ambiente. Para obter mais informações sobre os tipos de log do Amazon MWAA e como gerenciar seus grupos de logs, consulte the section called “Como visualizar logs do Airflow”

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Requisitos

Para usar o código de amostra nesta página, adicione as seguintes dependências ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python.

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

Apache Airflow v1

```
cx_Oracle==8.1.0
apache-airflow[oracle]==1.10.12
```

Exemplo de código

As etapas a seguir descrevem como criar o código do DAG que testará o plugin personalizado.

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

   ```
   cd dags
   ```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `oracle.py`.

   ```
   from airflow import DAG
   from airflow.operators.python_operator import PythonOperator
   from airflow.utils.dates import days_ago
   import os
   import cx_Oracle

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   def testHook(**kwargs):
       cx_Oracle.init_oracle_client()
       version = cx_Oracle.clientversion()
       print("cx_Oracle.clientversion",version)
   ```

```
    return version

with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    hook_test = PythonOperator(
        task_id="hook_test",
        python_callable=testHook,
        provide_context=True
    )
```

Criar o plugin personalizado

Esta seção descreve como baixar as dependências, criar o plug-in personalizado e o plugins.zip.

Download de dependências

O Amazon MWAA extrairá o conteúdo do plugins.zip em `/usr/local/airflow/plugins` sobre cada contêiner de agendador e trabalho do Amazon MWAA. Isso é usado para adicionar binários ao seu ambiente. As etapas a seguir descrevem como montar os arquivos necessários para o plugin personalizado.

Extraia a imagem de contêiner do Amazon Linux

1. Em seu prompt de comando, extraia a imagem do contêiner Amazon Linux e execute o contêiner localmente. Por exemplo:

   ```
   docker pull amazonlinux
   docker run -it amazonlinux:latest /bin/bash
   ```

   Seu prompt de comando deve invocar uma linha de comando bash. Por exemplo:

   ```
   bash-4.2#
   ```

2. Instale o recurso de I/O assíncrono nativo do Linux (libaio).

   ```
   yum -y install libaio
   ```

3. Mantenha essa janela aberta para as etapas subsequentes. Copiaremos os seguintes arquivos localmente: `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1`.

Baixe a pasta do cliente

1. Instale o pacote de descompactação localmente. Por exemplo:

```
sudo yum install unzip
```

2. Crie um diretório `oracle_plugin`. Por exemplo:

```
mkdir oracle_plugin
cd oracle_plugin
```

3. Use o comando cURL a seguir para baixar o instantclient-basic-linux.x64-18.5.0.0.0dbru.zip do Oracle Instant Client Downloads for Linux x86-64 (64-bit).

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. Descompacte o arquivo `client.zip`. Por exemplo:

```
unzip *.zip
```

Extrair arquivos do Docker

1. Em um novo prompt de comando, exiba e grave o ID do contêiner do Docker. Por exemplo:

```
docker container ls
```

   Seu prompt de comando deve retornar todos os contêineres e seus IDs. Por exemplo:

```
debc16fd6970
```

2. Em seu diretório `oracle_plugin`, extraia os arquivos `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1` para a pasta local `instantclient_18_5`. Por exemplo:

```
docker cp debc16fd6970:/lib64/libaio.so.1 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.0 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.1 instantclient_18_5/
```

Plug-in personalizado

O Apache Airflow executará o conteúdo dos arquivos Python na pasta de plugins na inicialização. Isto é usado para definir e modificar variáveis de ambiente. As seguintes etapas descrevem o código de exemplo para o plugin personalizado.

- Copie o conteúdo da amostra de código a seguir e salve localmente como `env_var_plugin_oracle.py`.

```
from airflow.plugins_manager import AirflowPlugin
import os

os.environ["LD_LIBRARY_PATH"]='/usr/local/airflow/plugins/instantclient_18_5'
os.environ["DPI_DEBUG_LEVEL"]="64"

class EnvVarPlugin(AirflowPlugin):
    name = 'env_var_plugin'
```

Plugins.zip

As etapas a seguir mostram como criar `plugins.zip`. O conteúdo deste exemplo pode ser combinado com seus outros plug-ins e binários em um único arquivo `plugins.zip`.

Compacte o conteúdo do diretório do plug-in

1. Em um prompt de comando, navegue até o diretório `oracle_plugin`. Por exemplo:

```
cd oracle_plugin
```

2. Compacte o diretório `instantclient_18_5` em plugins.zip. Por exemplo:

```
zip -r ../plugins.zip ./
```

3. Você deve ver o seguinte em seu prompt de comando:

```
oracle_plugin$ ls
client.zip  instantclient_18_5
```

4. Remova o arquivo `client.zip`. Por exemplo:

```
rm client.zip
```

Compacte o arquivo env_var_plugin_oracle.py

1. Adicione o arquivo `env_var_plugin_oracle.py` à raiz do plugins.zip. Por exemplo:

```
zip plugins.zip env_var_plugin_oracle.py
```

2. Agora seu arquivo plugins.zip deve incluir o seguinte:

```
env_var_plugin_oracle.py
instantclient_18_5/
```

Opções de configuração Airflow

Se você estiver usando o Apache Airflow v2, adicione `core.lazy_load_plugins : False` como uma opção de configuração do Apache Airflow. Para saber mais, consulte Usando opções de configuração para carregar plug-ins em 2.

Próximas etapas

- Saiba mais sobre como fazer o upload do `requirements.txt` arquivo neste exemplo para seu bucket do Amazon S3 em Como instalar dependências do Python.
- Saiba como fazer o upload do código DAG neste exemplo para a `dags` pasta em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Criar um plug-in personalizado que gera variáveis de ambiente de runtime

O exemplo a seguir mostra as etapas para criar um plug-in personalizado que gera ambientes variáveis no runtime em um ambiente Amazon Managed Workflows for Apache Airflow.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Requisitos

- Para usar esse exemplo de código com o Apache Airflow v1, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v1 em seu ambiente.

Plug-in personalizado

O Apache Airflow executará o conteúdo dos arquivos Python na pasta de plugins na inicialização. Isto é usado para definir e modificar variáveis de ambiente. As seguintes etapas descrevem o código de exemplo para o plugin personalizado.

1. No prompt de comando, navegue até o diretório em que seus plugins estão armazenados. Por exemplo:

```
cd plugins
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `env_var_plugin.py` no arquivo acima..

```
from airflow.plugins_manager import AirflowPlugin
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/lib/python3.7/
site-packages"
os.environ["JAVA_HOME"]="/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"

class EnvVarPlugin(AirflowPlugin):
     name = 'env_var_plugin'
```

Plugins.zip

As etapas a seguir mostram como criar `plugins.zip`. O conteúdo deste exemplo pode ser combinado com outros plug-ins e binários em um único arquivo `plugins.zip`.

1. Em sua prompt linha de comando, navegue até o diretório `hive_plugin` da etapa anterior. Por exemplo:

```
cd plugins
```

2. Compacte o conteúdo em sua pasta `plugins`.

```
zip -r ../plugins.zip ./
```

Opções de configuração do JEG

Se você estiver usando o Apache Airflow v2, adicione `core.lazy_load_plugins : False` como uma opção de configuração do Apache Airflow. Para saber mais, consulte Usando opções de configuração para carregar plug-ins em 2.

Próximas etapas

- Saiba mais sobre como fazer o upload do `requirements.txt` arquivo neste exemplo para seu bucket do Amazon S3 em Como instalar dependências do Python.
- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Como alterar o fuso horário de um DAG no Amazon MWAA

Por padrão, o Apache Airflow programa seu gráfico acíclico direcionado (DAG) em UTC+0. As etapas a seguir mostram como você pode alterar o fuso horário no qual o Amazon MWAA executa seus DAGs com o Pendulum. Como opção, este tópico demonstra como você pode criar um plug-in personalizado para alterar o fuso horário dos logs do Apache Airflow do seu ambiente.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Crie um plug-in para alterar o fuso horário nos logs do Airflow

O Apache Airflow executará os arquivos Python no diretório `plugins` na inicialização. Com o plug-in a seguir, você pode substituir o fuso horário do executor, o que modifica o fuso horário no qual o Apache Airflow grava os logs.

1. Crie um diretório com o nome `plugins` para seu plug-in personalizado e navegue até o diretório. Por exemplo:

```
$ mkdir plugins
$ cd plugins
```

2. Copie o conteúdo do exemplo do código a seguir e salve localmente como `dag-timezone-plugin.py` no arquivo `plugins`.

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. No diretório `plugins`, crie um arquivo Python vazio chamado `__init__.py`. Seu diretório `plugins` deve ser semelhante ao seguinte:

```
plugins/
|-- __init__.py
|-- dag-timezone-plugin.py
```

Criar uma **`plugins.zip`**

As etapas a seguir mostram como criar `plugins.zip`. O conteúdo deste exemplo pode ser combinado com outros plug-ins e binários em um único arquivo `plugins.zip`.

1. No prompt de comando, navegue até o diretório `plugins` da etapa anterior. Por exemplo:

```
cd plugins
```

2. Compacte o conteúdo em seu diretório `plugins`.

```
zip -r ../plugins.zip ./
```

3. Faça upload de `plugins.zip` para o seu bucket S3.

```
$ aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

Exemplo de código

Para alterar o fuso horário padrão (UTC+0) no qual o DAG é executado, usaremos uma biblioteca chamada Pendulum, uma biblioteca Python para trabalhar com data e hora com reconhecimento de fuso horário.

1. No prompt de comando, navegue até o diretório em que seus DAGs estão armazenados. Por exemplo:

```
$ cd dags
```

2. Copie o conteúdo do exemplo a seguir e salve como `tz-aware-dag.py`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
# Import the Pendulum library.
import pendulum

# Instantiate Pendulum and set your timezone.
local_tz = pendulum.timezone("America/Los_Angeles")

with DAG(
    dag_id = "tz_test",
    schedule_interval="0 12 * * *",
    catchup=False,
    start_date=datetime(2022, 1, 1, tzinfo=local_tz)
) as dag:
```

```
    bash_operator_task = BashOperator(
        task_id="tz_aware_task",
        dag=dag,
        bash_command="date"
    )
```

3. Execute o seguinte comando AWS CLI para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a IU do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se tiver êxito, você exibirá uma saída semelhante à seguinte nos logs de tarefas para `tz_aware_task` no DAG `tz_test`:

```
[2022-08-01, 12:00:00 PDT] {{subprocess.py:74}} INFO - Running command: ['bash', '-
c', 'date']
[2022-08-01, 12:00:00 PDT] {{subprocess.py:85}} INFO - Output:
[2022-08-01, 12:00:00 PDT] {{subprocess.py:89}} INFO - Mon Aug  1 12:00:00 PDT 2022
[2022-08-01, 12:00:00 PDT] {{subprocess.py:93}} INFO - Command exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=tz_test, task_id=tz_aware_task, execution_date=20220801T190033,
 start_date=20220801T190035, end_date=20220801T190035
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

Próximas etapas

- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Como atualizar um token CodeArtifact

Se você estiver usando o CodeArtifact para instalar dependências do Python, o Amazon MWAA exigirá um token ativo. Para permitir que o Amazon MWAA acesse um repositório CodeArtifact em runtime, você pode usar um script de startup e configurar `PIP_EXTRA_INDEX_URL` com o token.

O tópico a seguir descreve como você pode criar um script de startup que usa a operação da API CodeArtifact `get_authorization_token` para recuperar um novo token sempre que seu ambiente for inicializado ou atualizado.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.
- Um repositório CodeArtifact onde você armazena dependências para seu ambiente.

Permissões

Para atualizar o token CodeArtifact e gravar o resultado no Amazon S3, o Amazon MWAA deve ter as seguintes permissões no perfil de execução.

- A ação `codeartifact:GetAuthorizationToken` permite que o Amazon MWAA recupere um novo token do CodeArtifact. A política a seguir concede permissão para cada domínio

do CodeArtifact que você criar. Você pode restringir ainda mais o acesso aos seus domínios modificando o valor do recurso na instrução e especificando somente os domínios que você deseja que seu ambiente acesse.

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- A ação `sts:GetServiceBearerToken` é necessária para chamar a operação da API CodeArtifact `GetAuthorizationToken`. Essa operação retorna um token que deve ser usado ao usar um gerenciador de pacotes, como `pip` com CodeArtifact. Para usar um gerenciador de pacotes com um repositório CodeArtifact, um perfil de execução do seu ambiente deve permitir `sts:GetServiceBearerToken`, conforme mostrado na declaração da política a seguir.

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

Exemplo de código

As etapas a seguir descrevem como você pode criar um script de inicialização que atualize o token CodeArtifact.

1. Copie o conteúdo da amostra de código a seguir e salve localmente como `code_artifact_startup_script.sh`.

```
#!/bin/sh

# Startup script for MWAA, see https://docs.aws.amazon.com/mwaa/latest/userguide/
using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
```

```
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"
echo "CodeArtifact startup setup complete"
```

2. Navegue até a pasta em que você salvou o script. Use `cp` em uma nova janela de prompt para fazer o upload do script em seu bucket. Substitua *`your-s3-bucket`* por suas informações.

```
$ aws s3 cp code_artifact_startup_script.sh s3://your-s3-bucket/
code_artifact_startup_script.sh
```

Se tiver êxito, o Amazon S3 envia o caminho da URL para o objeto:

```
upload: ./code_artifact_startup_script.sh to s3://your-s3-bucket/
code_artifact_startup_script.sh
```

Depois de fazer o upload do script, seu ambiente atualiza e executa o script no startup.

Próximas etapas

- Saiba como usar scripts de startup para personalizar seu ambiente em the section called “Como usar um script de startup”.
- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Criação de um plug-in personalizado com o Apache Hive e o Hadoop

O Amazon MWAA extrai o conteúdo de um `plugins.zip` para `/usr/local/airflow/plugins`. Isso pode ser usado para adicionar binários aos seus contêineres. Além disso, o Apache Airflow executa o conteúdo dos arquivos Python na pasta `plugins` em inicialização, permitindo que você defina e modifique variáveis de ambiente. O exemplo a seguir mostra as etapas para criar um plug-in personalizado usando o Apache Hive e o Hadoop em um ambiente Amazon Managed Workflows for Apache Airflow e pode ser combinado com outros plug-ins e binários personalizados.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Requisitos

Para usar o código de exemplo nesta seção, adicione as seguintes dependências ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python.

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Apache Airflow v1

```
apache-airflow[hive]==1.10.12
```

Download de dependências

O Amazon MWAA extrairá o conteúdo do plugins.zip em `/usr/local/airflow/plugins` sobre cada contêiner de agendador e trabalho do Amazon MWAA. Isso é usado para adicionar binários ao seu ambiente. As etapas a seguir descrevem como montar os arquivos necessários para o plugin personalizado.

1. No prompt de comando, navegue até o diretório em que você gostaria de criar seu plug-in. Por exemplo:

   ```
   cd plugins
   ```

2. Baixe o Hadoop de um espelho, por exemplo:

   ```
   wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
   ```

3. Baixe o Hive de um espelho, por exemplo:

   ```
   wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
   ```

4. Crie um diretório. Por exemplo:

```
mkdir hive_plugin
```

5. Extraia o Hadoop.

```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. Extraia o Hive.

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

Plug-in personalizado

O Apache Airflow executará o conteúdo dos arquivos Python na pasta de plugins na inicialização. Isto é usado para definir e modificar variáveis de ambiente. As seguintes etapas descrevem o código de exemplo para o plugin personalizado.

1. Em um prompt de comando, navegue até o diretório `hive_plugin`. Por exemplo:

```
cd hive_plugin
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `hive_plugin.py` no diretório `hive_plugin`.

```
from airflow.plugins_manager import AirflowPlugin
import os
os.environ["JAVA_HOME"]="/usr/lib/jvm/jre"
os.environ["HADOOP_HOME"]='/usr/local/airflow/plugins/hadoop-3.3.0'
os.environ["HADOOP_CONF_DIR"]='/usr/local/airflow/plugins/hadoop-3.3.0/etc/hadoop'
os.environ["HIVE_HOME"]='/usr/local/airflow/plugins/apache-hive-3.1.2-bin'
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/plugins/
hadoop-3.3.0:/usr/local/airflow/plugins/apache-hive-3.1.2-bin/bin:/usr/local/
airflow/plugins/apache-hive-3.1.2-bin/lib"
os.environ["CLASSPATH"] = os.getenv("CLASSPATH") + ":/usr/local/airflow/plugins/
apache-hive-3.1.2-bin/lib"
class EnvVarPlugin(AirflowPlugin):
    name = 'hive_plugin'
```

3. Copie o conteúdo do texto a seguir e salve localmente como `.airflowignore` no diretório `hive_plugin`.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

As etapas a seguir mostram como criar `plugins.zip`. O conteúdo deste exemplo pode ser combinado com outros plug-ins e binários em um único `plugins.zip` arquivo.

1. Em sua prompt linha de comando, navegue até o diretório `hive_plugin` da etapa anterior. Por exemplo:

```
cd hive_plugin
```

2. Compacte o conteúdo em sua pasta `plugins`.

```
zip -r ../hive_plugin.zip ./
```

Exemplo de código

As etapas a seguir descrevem como criar o código do DAG que testará o plugin personalizado.

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `hive.py`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="hive_test_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    hive_test = BashOperator(
        task_id="hive_test",
```

```
        bash_command='hive --help'
    )
```

Opções de configuração do JEG

Se você estiver usando o Apache Airflow v2, adicione `core.lazy_load_plugins : False` como uma opção de configuração do Apache Airflow. Para saber mais, consulte Usando opções de configuração para carregar plug-ins em 2.

Próximas etapas

- Saiba mais sobre como fazer o upload do `requirements.txt` arquivo neste exemplo para seu bucket do Amazon S3 em Como instalar dependências do Python.
- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow

O exemplo a seguir mostra como corrigir o PythonVirtualEnvOperator do Apache Airflow com um plug-in personalizado no Amazon Managed Workflows para o Apache Airflow.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.
- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Permissões

- Nenhuma permissão adicional é necessária para usar o exemplo de código nesta página.

Requisitos

Para usar o código de exemplo nesta seção, adicione as seguintes dependências ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python.

```
virtualenv
```

Código de exemplo de plugin personalizado

O Apache Airflow executará o conteúdo dos arquivos Python na pasta de plugins na inicialização. Esse plug-in corrigirá o `PythonVirtualenvOperator` integrado durante o processo de inicialização para torná-lo compatível com o Amazon MWAA. As seguintes etapas mostram o código de exemplo para o plugin personalizado.

Apache Airflow v2

1. Em seu prompt de comando, navegue até o diretório `plugins` acima. Por exemplo:

```
cd plugins
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `virtual_python_plugin.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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 airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str,
 system_site_packages: bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v1

1. Em seu prompt de comando, navegue até o diretório `plugins` acima. Por exemplo:

```
cd plugins
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `virtual_python_plugin.py`.

```
from airflow.plugins_manager import AirflowPlugin
from airflow.operators.python_operator import PythonVirtualenvOperator

def _generate_virtualenv_cmd(self, tmp_dir):
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if self.system_site_packages:
        cmd.append('--system-site-packages')
    if self.python_version is not None:
        cmd.append('--python=python{}'.format(self.python_version))
    return cmd
PythonVirtualenvOperator._generate_virtualenv_cmd=_generate_virtualenv_cmd

class EnvVarPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Plugins.zip

As etapas a seguir mostram como criar `plugins.zip`.

1. Em seu prompt de comando, navegue até o diretório acima que contém `virtual_python_plugin.py`. Por exemplo:

```
cd plugins
```

2. Compacte o conteúdo em sua pasta `plugins`.

```
zip plugins.zip virtual_python_plugin.py
```

Exemplo de código

As etapas a seguir descrevem como criar o código do DAG para o plugin personalizado.

Apache Airflow v2

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `virtualenv_test.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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 airflow import DAG
from airflow.operators.python import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
```

```
        system_site_packages=False,
        dag=dag,
    )
```

Apache Airflow v1

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `virtualenv_test.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

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 airflow import DAG
from airflow.operators.python_operator import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)
```

```
with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Opções de configuração do Airflow

Se você estiver usando o Apache Airflow v2, adicione `core.lazy_load_plugins : False` como uma opção de configuração do Apache Airflow. Para saber mais, consulte Usando opções de configuração para carregar plug-ins em 2.

Próximas etapas

- Saiba mais sobre como fazer o upload do `requirements.txt` arquivo neste exemplo para seu bucket do Amazon S3 em Como instalar dependências do Python.
- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Saiba mais sobre como fazer o upload do `plugins.zip` arquivo neste exemplo para seu bucket do Amazon S3 em Instalando plug-ins personalizados.

Como invocar DAGs com uma função do Lambda

O exemplo de código a seguir usa uma função AWS Lambda para obter um token CLI do Apache Airflow e invocar um gráfico acíclico direcionado (DAG) em um ambiente Amazon MWAA.

Tópicos

Version (Versão)

- É possível usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar esse exemplo de código, você deve:

- Usar o modo de acesso à rede pública para seu ambiente Amazon MWAA.
- Ter uma função do Lambda usando o runtime mais recente do Python.

> **Note**
>
> Se a função do Lambda e seu ambiente Amazon MWAA estiverem na mesma VPC, você poderá usar esse código em uma rede privada. Para essa configuração, o perfil de execução da função do Lambda precisa de permissão para chamar a operação da API CreateNetworkInterface do Amazon Elastic Compute Cloud (Amazon EC2). Você pode fornecer essa permissão usando a política `AWSLambdaVPCAccessExecutionRole` AWS gerenciada.

Permissões

Para usar o exemplo de código nesta página, o perfil de execução do seu ambiente Amazon MWAA precisa de acesso para realizar a ação `airflow:CreateCliToken`. Você pode fornecer essa permissão usando a política `AmazonMWAAAirflowCliAccess` AWS gerenciada:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Para ter mais informações, consulte Política de CLI do Apache Airflow: Acesso ao AmazonMWAA AirflowCli.

Dependências

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

1. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha sua função do Lambda na lista Funções.
3. Na página da função, copie o código a seguir e substitua-o pelos nomes dos seus recursos:
   - `YOUR_ENVIRONMENT_NAME`: o nome do seu ambiente do Amazon MWAA.
   - `YOUR_DAG_NAME`: o nome do DAG que você deseja invocar.

```
import boto3
import http.client
import base64
import ast
mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

def lambda_handler(event, context):
    # get web token
    mwaa_cli_token = client.create_cli_token(
        Name=mwaa_env_name
    )

    conn = http.client.HTTPSConnection(mwaa_cli_token['WebServerHostname'])
    payload = mwaa_cli_command + " " + dag_name
    headers = {
```

```
      'Authorization': 'Bearer ' + mwaa_cli_token['CliToken'],
      'Content-Type': 'text/plain'
    }
    conn.request("POST", "/aws_mwaa/cli", payload, headers)
    res = conn.getresponse()
    data = res.read()
    dict_str = data.decode("UTF-8")
    mydata = ast.literal_eval(dict_str)
    return base64.b64decode(mydata['stdout'])
```

4. Escolha Implantar.
5. Escolha Testar para invocar sua função usando o console Lambda.
6. Para verificar se seu Lambda invocou seu DAG com sucesso, use o console Amazon MWAA para navegar até a IU do Apache Airflow do seu ambiente e faça o seguinte:
   a. Na página DAGs, localize seu novo DAG de destino na lista de DAGs.
   b. Em Última execução, verifique a data e hora da última execução do DAG. Esse carimbo de data/hora deve ser semelhante ao carimbo de data/hora mais recente para `invoke_dag` em seu outro ambiente.
   c. Em Tarefas recentes, verifique se a última execução foi bem-sucedida.

Como invocar DAGs em diferentes ambientes do Amazon MWAA

O exemplo de código a seguir cria um token da CLI do Apache Airflow. O código, então, usa um gráfico acíclico direcionado (directed acyclic graph, DAG) em um ambiente do Amazon MWAA para invocar um DAG em um ambiente diferente do Amazon MWAA.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de exemplo nesta página, você precisará de:

- Dois ambientes Amazon MWAA com acesso ao servidor web de rede pública, incluindo seu ambiente atual.
- Um exemplo de DAG que foi carregado para o bucket do Amazon Simple Storage Service (Amazon S3) do seu ambiente de destino.

Permissões

Para usar o exemplo de código nesta página, o perfil de execução do seu ambiente deve ter permissão para criar um token da CLI do Apache Airflow. Você pode anexar a política `AmazonMWAAAirflowCliAccess` gerenciada pela AWS para conceder essa permissão.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
```

Para obter mais informações, consulte Política de CLI do Apache Airflow: Acesso ao AmazonMWAA AirflowCli.

Dependências

- Para usar esse exemplo de código com o Apache Airflow v2, nenhuma dependência adicional é necessária. O código usa a instalação básica do Apache Airflow v2 em seu ambiente.

Exemplo de código

O exemplo de código a seguir pressupõe que você esteja usando um DAG em seu ambiente atual para invocar um DAG em outro ambiente.

1. Em seu terminal, navegue até o diretório em que seu código DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo do exemplo de código a seguir e salve-o localmente como `invoke_dag.py`. Substitua os seguintes valores por suas informações.

   - `your-new-environment-name`: o nome do outro ambiente onde deseja invocar o DAG.
   - `your-target-dag-id`: o ID do DAG no outro ambiente que você deseja invocar.

```
from airflow.decorators import dag, task
import boto3
from datetime import datetime, timedelta
import os, requests

DAG_ID = os.path.basename(__file__).replace(".py", "")

@task()
def invoke_dag_task(**kwargs):
    client = boto3.client('mwaa')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwaa/cli"
    body = 'dags trigger your-target-dag-id'
    headers = {
        'Authorization' : 'Bearer ' + token['CliToken'],
        'Content-Type': 'text/plain'
        }
    requests.post(url, data=body, headers=headers)

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    dagrun_timeout=timedelta(minutes=60),
    catchup=False
```

```
    )
def invoke_dag():
    t = invoke_dag_task()

invoke_dag_test = invoke_dag()
```

3. Execute o seguinte comando AWS CLI para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a IU do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se o DAG for executado com êxito, você verá uma saída semelhante à seguinte nos logs de tarefas de `invoke_dag_task`.

```
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: None
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=invoke_dag, task_id=invoke_dag_task, execution_date=20220101T120000,
 start_date=20220101T120000, end_date=20220101T120000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

Para verificar se seu DAG foi invocado com sucesso, navegue até a IU do Apache Airflow para seu novo ambiente e faça o seguinte:

a. Na página DAGs, localize seu novo DAG de destino na lista de DAGs.

b. Em Última execução, verifique a data e hora da última execução do DAG. Esse carimbo de data/hora deve ser semelhante ao carimbo de data/hora mais recente para `invoke_dag` em seu outro ambiente.

c. Em Tarefas recentes, verifique se a última execução foi bem-sucedida.

Como usar o Amazon MWAA com Amazon RDS para Microsoft SQL Server

É possível usar o Amazon MWAA para se conectar a um https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.htmlRDS para SQL Server. O exemplo do código a seguir usa

DAGs em um ambiente Amazon Managed Workflows for Apache Airflow para se conectar e executar consultas em um Amazon RDS para Microsoft SQL Server.

Tópicos

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.
- O Amazon MWAA e o RDS para SQL Server estão sendo executados na mesma Amazon VPC/
- Os grupos de segurança de VPC do Amazon MWAA e do servidor estão configurados com as seguintes conexões:
  - Uma regra de entrada para a porta `1433` aberta para o Amazon RDS no grupo de segurança do Amazon MWAA
  - Ou uma regra de saída para a porta `1433` aberta do Amazon MWAA para o RDS
- A conexão do Apache Airflow de RDS para SQL Server reflete o nome do host, a porta, o nome de usuário e a senha do banco de dados do servidor SQL do Amazon RDS criado no processo anterior.

Dependências

Para usar o código de amostra nesta seção, adicione a seguinte dependência ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python

Apache Airflow v2

```
apache-airflow-providers-microsoft-mssql==1.0.1
apache-airflow-providers-odbc==1.0.1
pymssql==2.2.1
```

Apache Airflow v1

```
apache-airflow[mssql]==1.10.12
```

Conexão Apache Airflow v2

Caso esteja usando uma conexão no Apache Airflow v2, certifique-se de que o objeto de conexão Airflow inclua os seguintes pares de chave-valor:

1. ID de conexão: mssql_default
2. Tipo de conexão: Amazon Web Services
3. Host: `YOUR_DB_HOST`
4. Esquema:
5. Login: admin
6. Senha:
7. Porta: 1433
8. Extra:

Exemplo de código

1. No prompt de comando, navegue até o diretório em que o código do DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `sql-server.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
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.
"""
import pymssql
import logging
import sys
from airflow import DAG
from datetime import datetime
from airflow.operators.mssql_operator import MsSqlOperator
from airflow.operators.python_operator import PythonOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'mssql_conn_example', default_args=default_args, schedule_interval=None)

drop_db = MsSqlOperator(
   task_id="drop_db",
   sql="DROP DATABASE IF EXISTS testdb;",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_db = MsSqlOperator(
```

```
   task_id="create_db",
   sql="create database testdb;",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_table = MsSqlOperator(
   task_id="create_table",
   sql="CREATE TABLE testdb.dbo.pet (name VARCHAR(20), owner VARCHAR(20));",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

insert_into_table = MsSqlOperator(
   task_id="insert_into_table",
   sql="INSERT INTO testdb.dbo.pet VALUES ('Olaf', 'Disney');",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

def select_pet(**kwargs):
   try:
        conn = pymssql.connect(
            server='sampledb.<xxxxxx>.<region>.rds.amazonaws.com',
            user='admin',
            password='<yoursupersecretpassword>',
            database='testdb'
        )

        # Create a cursor from the connection
        cursor = conn.cursor()
        cursor.execute("SELECT * from testdb.dbo.pet")
        row = cursor.fetchone()

        if row:
            print(row)
   except:
      logging.error("Error when creating pymssql database connection: %s",
 sys.exc_info()[0])

select_query = PythonOperator(
```

```
    task_id='select_query',
    python_callable=select_pet,
    dag=dag,
)

drop_db >> create_db >> create_table >> insert_into_table >> select_query
```

Próximas etapas

- Saiba mais sobre como fazer o upload do `requirements.txt` arquivo neste exemplo para seu bucket do Amazon S3 em Como instalar dependências do Python.
- Saiba como fazer o upload do código DAG neste exemplo para a pasta `dags` em seu bucket do Amazon S3 em Como adicionar ou atualizar DAGs.
- Explore exemplos de scripts e outros exemplos de módulos pymssql.
- Saiba mais sobre a execução do código SQL em um banco de dados Microsoft SQL específico usando mssql_operator no Guia de referência do Apache Airflow.

Como usar o Amazon MWAA com o Amazon EMR

O exemplo a seguir demonstra como usar o Amazon Managed Workflows para Apache Airflow com o Amazon EMR.

Tópicos

- Versão
- Exemplo de código

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

Exemplo de código

```
    """
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
    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 airflow import DAG

    from airflow.contrib.operators.emr_add_steps_operator import EmrAddStepsOperator
    from airflow.contrib.operators.emr_create_job_flow_operator import
EmrCreateJobFlowOperator
    from airflow.contrib.sensors.emr_step_sensor import EmrStepSensor

    from airflow.utils.dates import days_ago
    from datetime import timedelta
    import os

    DAG_ID = os.path.basename(__file__).replace(".py", "")

    DEFAULT_ARGS = {
        'owner': 'airflow',
        'depends_on_past': False,
        'email': ['airflow@example.com'],
        'email_on_failure': False,
        'email_on_retry': False,
    }

    SPARK_STEPS = [
        {
            'Name': 'calculate_pi',
            'ActionOnFailure': 'CONTINUE',
            'HadoopJarStep': {
                'Jar': 'command-runner.jar',
                'Args': ['/usr/lib/spark/bin/run-example', 'SparkPi', '10'],
            },
        }
    ]
```

```
JOB_FLOW_OVERRIDES = {
    'Name': 'my-demo-cluster',
    'ReleaseLabel': 'emr-5.30.1',
    'Applications': [
        {
            'Name': 'Spark'
        },
    ],
    'Instances': {
        'InstanceGroups': [
            {
                'Name': "Master nodes",
                'Market': 'ON_DEMAND',
                'InstanceRole': 'MASTER',
                'InstanceType': 'm5.xlarge',
                'InstanceCount': 1,
            },
            {
                'Name': "Slave nodes",
                'Market': 'ON_DEMAND',
                'InstanceRole': 'CORE',
                'InstanceType': 'm5.xlarge',
                'InstanceCount': 2,
            }
        ],
        'KeepJobFlowAliveWhenNoSteps': False,
        'TerminationProtected': False,
        'Ec2KeyName': 'mykeypair',
    },
    'VisibleToAllUsers': True,
    'JobFlowRole': 'EMR_EC2_DefaultRole',
    'ServiceRole': 'EMR_DefaultRole'
}

with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['emr'],
) as dag:
```

```
        cluster_creator = EmrCreateJobFlowOperator(
            task_id='create_job_flow',
            job_flow_overrides=JOB_FLOW_OVERRIDES
        )

        step_adder = EmrAddStepsOperator(
            task_id='add_steps',
            job_flow_id="{{ task_instance.xcom_pull(task_ids='create_job_flow',
key='return_value') }}",
            aws_conn_id='aws_default',
            steps=SPARK_STEPS,
        )

        step_checker = EmrStepSensor(
            task_id='watch_step',
            job_flow_id="{{ task_instance.xcom_pull('create_job_flow',
key='return_value') }}",
            step_id="{{ task_instance.xcom_pull(task_ids='add_steps',
key='return_value')[0] }}",
            aws_conn_id='aws_default',
        )

        cluster_creator >> step_adder >> step_checker
```

Como usar o Amazon MWAA com o Amazon EKS

O exemplo a seguir demonstra como usar o Amazon Managed Workflows for Apache Airflow com o Amazon EKS.

Tópicos

- Versão
- Pré-requisitos
- Criar uma chave pública para o Amazon EC2
- Criar um cluster
- Crie um namespace mwaa
- Criar um perfil para o namespace mwaa
- Crie e anexe um perfil do IAM para o cluster do Amazon EKS
- Crie o arquivo requirements.txt
- Crie um mapeamento de identidade para o Amazon EKS

- Criar a kubeconfig
- Criar um DAG
- Adicione o DAG e kube_config.yaml ao bucket do Amazon S3
- Habilite e acione o exemplo

Versão

- O código de amostra nesta página pode ser usado com o Apache Airflow v1 em Python 3.7.

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o exemplo deste tópico, você precisará de:

- Um ambiente Amazon MWAA.
- eksctl. Para saber mais, consulte Instalar eksctl.
- kubectl. Para saber mais, consulte Instalar e configurar o kubectl. Em alguns casos, ele é instalado com o eksctl.
- Um par de chaves do EC2 na região em que criar seu ambiente Amazon MWAA. Para saber mais, consulte Criação ou importação de um key pair.

> **Note**
>
> Ao usar um comando `eksctl`, você pode incluir um `--profile` para especificar um perfil diferente do padrão.

Criar uma chave pública para o Amazon EC2

Use o comando a seguir para criar uma chave pública de seu par de chaves privadas.

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

Para saber mais, consulte Recuperar a chave pública para seu par de chaves.

Criar um cluster

Use o seguinte comando para criar o cluster. Se quiser um nome personalizado para o cluster ou criá-lo em uma região diferente, substitua o nome e os valores da região. Você deve criar o cluster na mesma região em que criou o ambiente do Amazon MWAA. Substitua os valores das sub-redes para que correspondam às sub-redes na sua rede Amazon VPC que você usa para o Amazon MWAA. Substitua o valor de `ssh-public-key` para corresponder à chave que você usa. Você pode usar uma chave existente do Amazon EC2 que esteja na mesma região ou criar uma nova chave na mesma região em que você criar seu ambiente Amazon MWAA.

```
eksctl create cluster \
--name mwaa-eks \
--region us-west-2 \
--version 1.18 \
--nodegroup-name linux-nodes \
--nodes 3 \
--nodes-min 1 \
--nodes-max 4 \
--with-oidc \
--ssh-access \
--ssh-public-key MyPublicKey \
--managed \
--vpc-public-subnets "subnet-11111111111111111, subnet-2222222222222222" \
--vpc-private-subnets "subnet-33333333333333333, subnet-44444444444444444"
```

Leva algum tempo para concluir a criação do cluster. Depois de concluído, você pode verificar se o cluster foi criado com sucesso e se o provedor IAM OIDC está configurado usando o seguinte comando:

```
eksctl utils associate-iam-oidc-provider \
--region us-west-2 \
--cluster mwaa-eks \
--approve
```

Crie um namespace **`mwaa`**

Depois de confirmar que o cluster foi criado com sucesso, use o comando a seguir para criar um namespace para os pods.

```
kubectl create namespace mwaa
```

Criar um perfil para o namespace **mwaa**

Depois de criar o namespace, crie um perfil e uma associação de perfil para um usuário do Amazon MWAA no EKS que possa executar pods em um namespace do MWAA. Se você usou um nome diferente para o namespace, substitua mwaa em `-n` *mwaa* pelo nome usado.

```
cat << EOF | kubectl apply -f - -n mwaa
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: mwaa-role
rules:
  - apiGroups:
      - ""
      - "apps"
      - "batch"
      - "extensions"
    resources:
      - "jobs"
      - "pods"
      - "pods/attach"
      - "pods/exec"
      - "pods/log"
      - "pods/portforward"
      - "secrets"
      - "services"
    verbs:
      - "create"
      - "delete"
      - "describe"
      - "get"
      - "list"
      - "patch"
      - "update"
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: mwaa-role-binding
subjects:
```

```
- kind: User
  name: mwaa-service
roleRef:
  kind: Role
  name: mwaa-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

Confirme se o novo perfil pode acessar o cluster do Amazon EKS ao executar o comando a seguir. Certifique-se de usar o nome correto caso não tenha usado *mwaa*:

```
kubectl get pods -n mwaa --as mwaa-service
```

Você verá uma mensagem de retorno dizendo:

```
No resources found in mwaa namespace.
```

Crie e anexe um perfil do IAM para o cluster do Amazon EKS

Você deve criar um perfil do IAM e depois vinculá-lo ao cluster Amazon EKS (k8s) para que ele possa ser usado para autenticação por meio do IAM. O perfil é usado somente para fazer login no cluster e não tem nenhuma permissão para o console ou as chamadas de API.

Crie um novo perfil para o ambiente do Amazon MWAA usando as etapas em Perfil de execução do Amazon MWAA. No entanto, em vez de criar e anexar as políticas descritas neste tópico, anexe a seguinte política:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:${MWAA_REGION}:${ACCOUNT_NUMBER}:environment/
${MWAA_ENV_NAME}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
```

```
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:${MWAA_REGION}:${ACCOUNT_NUMBER}:log-group:airflow-
${MWAA_ENV_NAME}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
```

```
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:${MWAA_REGION}:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:${ACCOUNT_NUMBER}:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "sqs.${MWAA_REGION}.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:${MWAA_REGION}:${ACCOUNT_NUMBER}:cluster/
${EKS_CLUSTER_NAME}"
        }
    ]
}
```

Depois de criar o perfil, edite seu ambiente do Amazon MWAA para usar o perfil que você criou como o perfil de execução para o ambiente. Para alterar o perfil, edite o ambiente a ser usado. Você seleciona a função de execução em Permissões.

Problemas conhecidos:

- Há um problema conhecido com ARNs de perfil com subcaminhos que não conseguem se autenticar com o Amazon EKS. A solução alternativa para isso é criar o perfil de serviço manualmente, em vez de usar o criado pelo próprio Amazon MWAA. Para saber mais, consulte

Funções com caminhos não funcionam quando o caminho é incluído em seu ARN no aws-auth configmap

- Se a lista de serviços do Amazon MWAA não estiver disponível no IAM, você precisará escolher uma política de serviço alternativa, como o Amazon EC2, e depois atualizar a política de confiança do perfil de acordo com o seguinte:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "airflow-env.amazonaws.com",
          "airflow.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para saber mais, consulte Como usar políticas de confiança com funções do IAM.

Crie o arquivo requirements.txt

Para usar o código de amostra nesta seção, certifique-se de ter adicionado uma das seguintes opções de banco de dados ao seu `requirements.txt`. Para saber mais, consulte Como instalar dependências do Python.

Apache Airflow v2

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Apache Airflow v1

```
awscli
kubernetes==12.0.1
```

Crie um mapeamento de identidade para o Amazon EKS

Use o ARN para o perfil que você criou no comando a seguir para criar um mapeamento de identidade para o Amazon EKS. Altere a região *your-region* para a região em que o ambiente foi criado. Substitua o ARN do perfil e, por fim, substitua *mwaa-execution-role* pelo perfil de execução do seu ambiente.

```
eksctl create iamidentitymapping \
--region your-region \
--cluster mwaa-eks \
--arn arn:aws:iam::111222333444:role/mwaa-execution-role \
--username mwaa-service
```

Criar a **kubeconfig**

Use o comando a seguir para criar `kubeconfig`:

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

Se você usou um perfil específico ao executar `update-kubeconfig`, precisará remover a seção `env:` adicionada ao arquivo kube_config.yaml para que ela funcione corretamente com o Amazon MWAA. Para fazer isso, exclua do arquivo o seguinte e então salve-o:

```
env:
- name: AWS_PROFILE
  value: profile_name
```

Criar um DAG

Use o exemplo de código a seguir para criar um arquivo Python, como `mwaa_pod_example.py` para o DAG.

Apache Airflow v2

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
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 airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
 KubernetesPodOperator

default_args = {
   'owner': 'aws',
   'depends_on_past': False,
   'start_date': datetime(2019, 2, 20),
   'provide_context': True
}

dag = DAG(
   'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                       namespace="mwaa",
                       image="ubuntu:18.04",
                       cmds=["bash"],
                       arguments=["-c", "ls"],
                       labels={"foo": "bar"},
                       name="mwaa-pod-test",
                       task_id="pod-task",
                       get_logs=True,
                       dag=dag,
                       is_delete_operator_pod=False,
                       config_file=kube_config_path,
                       in_cluster=False,
                       cluster_context='aws'
```

```
                        )
```

Apache Airflow v1

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
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 airflow import DAG
from datetime import datetime
from airflow.contrib.operators.kubernetes_pod_operator import KubernetesPodOperator

default_args = {
   'owner': 'aws',
   'depends_on_past': False,
   'start_date': datetime(2019, 2, 20),
   'provide_context': True
}

dag = DAG(
   'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                        namespace="mwaa",
                        image="ubuntu:18.04",
                        cmds=["bash"],
                        arguments=["-c", "ls"],
                        labels={"foo": "bar"},
                        name="mwaa-pod-test",
                        task_id="pod-task",
```

```
                    get_logs=True,
                    dag=dag,
                    is_delete_operator_pod=False,
                    config_file=kube_config_path,
                    in_cluster=False,
                    cluster_context='aws'
                    )
```

Adicione o DAG e **`kube_config.yaml`** ao bucket do Amazon S3

Coloque o DAG criado e o arquivo `kube_config.yaml` no bucket do Amazon S3 para o ambiente Amazon MWAA. Você pode colocar arquivos em seu bucket usando o console do Amazon S3 ou a AWS Command Line Interface.

Habilite e acione o exemplo

No Apache Airflow, habilite o exemplo e, em seguida, acione-o.

Depois de ser executado e concluído com sucesso, use o comando a seguir para verificar o pod:

```
kubectl get pods -n mwaa
```

Você deve ver saída semelhante a:

```
NAME READY STATUS RESTARTS AGE
mwaa-pod-test-aa11bb22cc334444555566667778888 0/1 Completed 0 2m23s
```

Em seguida, você pode verificar a saída do pod com o comando a seguir. Substitua o valor do nome pelo valor retornado do comando anterior:

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc334444555566667778888
```

Como conectar-se ao Amazon ECS usando o **`ECSOperator`**

O tópico descreve como você pode usar `ECSOperator` para se conectar a um contêiner do Amazon Elastic Container Service (Amazon ECS) do Amazon MWAA. Nas etapas a seguir, você adicionará as permissões necessárias ao perfil de execução do seu ambiente, usará um modelo AWS CloudFormation para criar um cluster Amazon ECS Fargate e, por fim, criará e carregará um DAG que se conecta ao seu novo cluster.

Tópicos

Versão

- Você pode usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Para usar o código de amostra nesta página, você precisará do seguinte:

- Um ambiente Amazon MWAA.

Permissões

- O perfil de execução do seu ambiente precisa de permissão para executar tarefas no Amazon ECS. Você pode anexar a política gerenciada por AWS AmazonECS_FullAccess ao seu perfil de execução ou criar e anexar a seguinte política ao seu perfil de execução.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
```

```
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ecs-tasks.amazonaws.com"
                }
            }
        }
    ]
}
```

- Além de adicionar as permissões necessárias para executar tarefas no Amazon ECS, você também deve modificar a declaração de política do CloudWatch Logs em seu perfil de execução do Amazon MWAA para permitir acesso ao grupo de logs de tarefas do Amazon ECS, conforme mostrado a seguir. O grupo de logs do Amazon ECS é criado pelo modelo AWS CloudFormation em the section called “Crie um cluster do Amazon ECS”.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents",
        "logs:GetLogEvents",
        "logs:GetLogRecord",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults"
    ],
    "Resource": [
        "arn:aws:logs:region:account-id:log-group:airflow-environment-name-*",
        "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
    ]
}
```

Para obter mais informações sobre o perfil de execução do Amazon MWAA e como anexar uma política, consulte Perfil de execução.

Crie um cluster do Amazon ECS

Ao usar o modelo AWS CloudFormation a seguir, você criará um cluster Amazon ECS Fargate para usar com seu fluxo de trabalho do Amazon MWAA. Para obter mais informações, consulte Como criar uma definição de tarefa no Guia do desenvolvedor do Amazon Elastic Container Service.

1. Crie um arquivo JSON com o seguinte conteúdo e salve-o como `ecs-mwaa-cfn.json`.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "This template deploys an ECS Fargate cluster with an Amazon
 Linux image as a test for MWAA.",
    "Parameters": {
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "Select a VPC that allows instances access to ECR, as
 used with MWAA."
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "Select at two private subnets in your selected VPC, as
 used with MWAA."
        },
        "SecurityGroups": {
            "Type": "List<AWS::EC2::SecurityGroup::Id>",
            "Description": "Select at least one security group in your selected
 VPC, as used with MWAA."
        }
    },
    "Resources": {
        "Cluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": {
                    "Fn::Sub": "${AWS::StackName}-cluster"
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "Properties": {
                "LogGroupName": {
                    "Ref": "AWS::StackName"
```

```
                },
                "RetentionInDays": 30
            }
        },
        "ExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "ecs-tasks.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        },
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Fn::Sub": "${AWS::StackName}-task"
                },
                "Cpu": 2048,
                "Memory": 4096,
                "NetworkMode": "awsvpc",
                "ExecutionRoleArn": {
                    "Ref": "ExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Fn::Sub": "${AWS::StackName}-container"
                        },
                        "Image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/
amazonlinux:latest",
                        "PortMappings": [
```

```
                        {
                            "Protocol": "tcp",
                            "ContainerPort": 8080,
                            "HostPort": 8080
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "awslogs-region": {
                                "Ref": "AWS::Region"
                            },
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-stream-prefix": "ecs"
                        }
                    }
                }
            ],
            "RequiresCompatibilities": [
                "FARGATE"
            ]
        }
    },
    "Service": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "ServiceName": {
                "Fn::Sub": "${AWS::StackName}-service"
            },
            "Cluster": {
                "Ref": "Cluster"
            },
            "TaskDefinition": {
                "Ref": "TaskDefinition"
            },
            "DesiredCount": 1,
            "LaunchType": "FARGATE",
            "PlatformVersion": "1.3.0",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "ENABLED",
                    "Subnets": {
```

```
                            "Ref": "SubnetIds"
                        },
                        "SecurityGroups": {
                            "Ref": "SecurityGroups"
                        }
                    }
                }
            }
        }
    }
}
```

2. Em seu prompt de comando, use o seguinte comando AWS CLI para criar uma nova pilha. Você deve substituir os valores `SecurityGroups` e `SubnetIds` por valores para os grupos de segurança e sub-redes do seu ambiente do Amazon MWAA.

```
$ aws cloudformation create-stack \
--stack-name my-ecs-stack --template-body file://ecs-mwaa-cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=your-mwaa-security-group \
ParameterKey=SubnetIds,ParameterValue=your-mwaa-subnet-1\\,your-mwaa-subnet-1 \
--capabilities CAPABILITY_IAM
```

Como alternativa, você pode usar o seguinte script shell. O script recupera os valores necessários para os grupos de segurança e sub-redes do seu ambiente usando o comando `get-environment` AWS CLI e, em seguida, cria a pilha adequadamente. Para executar o script.

a. Copie e salve o script como `ecs-stack-helper.sh` no mesmo diretório do seu modelo AWS CloudFormation.

```
#!/bin/bash

joinByString() {
  local separator="$1"
  shift
  local first="$1"
  shift
  printf "%s" "$first" "${@/#/$separator}"
}

response=$(aws mwaa get-environment --name $1)
```

```
securityGroupId=$(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SecurityGroupIds[]')
subnetIds=$(joinByString '\,' $(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SubnetIds[]'))

aws cloudformation create-stack --stack-name $2 --template-body file://ecs-
cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=$securityGroupId \
ParameterKey=SubnetIds,ParameterValue=$subnetIds \
--capabilities CAPABILITY_IAM
```

b. Execute o script usando os comandos a seguir. Substitua `environment-name` e `stack-name` por suas informações.

```
$ chmod +x ecs-stack-helper.sh
$ ./ecs-stack-helper.bash environment-name stack-name
```

Se tiver êxito, você verá o seguinte resultado exibindo sua nova ID de pilha AWS CloudFormation.

```
{
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

Depois que sua pilha AWS CloudFormation estiver concluída e AWS provisionar seus recursos do Amazon ECS, você estará pronto para criar e carregar seu DAG.

Exemplo de código

1. Abra um prompt de comando e navegue até o diretório em que seu código DAG está armazenado. Por exemplo:

```
cd dags
```

2. Copie o conteúdo da amostra de código a seguir e salve localmente como `mwaa-ecs-operator.py` e, em seguida, faça o upload de seu novo DAG para o Amazon S3.

```
from http import client
```

```
from airflow import DAG
from airflow.providers.amazon.aws.operators.ecs import ECSOperator
from airflow.utils.dates import days_ago
import boto3

CLUSTER_NAME="mwaa-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
 information.
CONTAINER_NAME="mwaa-ecs-test-container" #Replace value for CONTAINER_NAME with
 your information.
LAUNCH_TYPE="FARGATE"

with DAG(
    dag_id = "ecs_fargate_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1)
) as dag:
    client=boto3.client('ecs')
    services=client.list_services(cluster=CLUSTER_NAME,launchType=LAUNCH_TYPE)

 service=client.describe_services(cluster=CLUSTER_NAME,services=services['serviceArns'])

    ecs_operator_task = ECSOperator(
        task_id = "ecs_operator_task",
        dag=dag,
        cluster=CLUSTER_NAME,
        task_definition=service['services'][0]['taskDefinition'],
        launch_type=LAUNCH_TYPE,
        overrides={
            "containerOverrides":[
                {
                    "name":CONTAINER_NAME,
                    "command":["ls", "-l", "/"],
                },
            ],
        },

        network_configuration=service['services'][0]['networkConfiguration'],
        awslogs_group="mwaa-ecs-zero",
        awslogs_stream_prefix=f"ecs/{CONTAINER_NAME}",
    )
```

> **Note**
>
> No exemplo do DAG, para `awslogs_group`, talvez você precise modificar o grupo de logs com o nome do seu grupo de logs de tarefas do Amazon ECS. O exemplo pressupõe um grupo de logs chamado `mwaa-ecs-zero`. Para `awslogs_stream_prefix`, use o prefixo do fluxo de logs de tarefas do Amazon ECS. O exemplo pressupõe um prefixo de fluxo de log, `ecs`.

3. Execute o seguinte comando AWS CLI para copiar o DAG para o bucket do seu ambiente e, em seguida, acionar o DAG usando a IU do Apache Airflow.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Se tiver êxito, você verá uma saída semelhante à seguinte nos logs de tarefas para `ecs_operator_task` no DAG `ecs_fargate_dag`:

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:300}} INFO - Running ECS Task -
Task definition: arn:aws:ecs:us-west-2:123456789012:task-definition/mwaa-ecs-test-
task:1 - on cluster mwaa-ecs-test-cluster
[2022-01-01, 12:00:00 UTC] {{ecs-operator-test.py:302}} INFO - ECSOperator
 overrides:
{'containerOverrides': [{'name': 'mwaa-ecs-test-container', 'command': ['ls', '-l',
 '/']}]}
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:379}} INFO - ECS task ID is:
 e012340b5e1b43c6a757cf012c635935
[2022-01-01, 12:00:00 UTC] {{ecs.py:313}} INFO - Starting ECS Task Log Fetcher
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] total
 52
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 bin -> usr/bin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x   2 root root 4096 Apr  9  2019 boot
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   5 root root  340 Jul 19 17:54 dev
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   1 root root 4096 Jul 19 17:54 etc
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 home
```

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 lib -> usr/lib
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    9 Jun 13 18:51 lib64 -> usr/lib64
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:51 local
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 media
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 mnt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 opt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x 103 root root    0 Jul 19 17:54 proc
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
x-\-\-   2 root root 4096 Apr  9  2019 root
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:52 run
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    8 Jun 13 18:51 sbin -> usr/sbin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 srv
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x  13 root root    0 Jul 19 17:54 sys
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 drwxrwxrwt   2 root root 4096 Jun 13 18:51 tmp
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  13 root root 4096 Jun 13 18:51 usr
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  18 root root 4096 Jun 13 18:52 var
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:328}} INFO - ECS Task has been successfully
 executed
```

Como usar DBT com o Amazon MWAA

Este tópico demonstra como é possível usar o DBT e o Postgres com o Amazon MWAA. Nas etapas a seguir, você adicionará as dependências necessárias ao seu `requirements.txt` e fará o upload de um exemplo de projeto de DBT para o bucket Amazon S3 do seu ambiente. Em seguida, você

usará um exemplo de DAG para verificar se o Amazon MWAA instalou as dependências e, por fim, usará `BashOperator` para executar o projeto de DBT.

Tópicos

Version (Versão)

- É possível usar o exemplo de código nesta página com o Apache Airflow v2 e superior no Python 3.10.

Pré-requisitos

Antes de concluir as etapas a seguir, você precisará do seguinte:

- Um ambiente do Amazon MWAA usando o Apache Airflow v2.2.2. Esse exemplo foi gravado e testado com a versão 2.2.2. Talvez seja necessário modificar o exemplo para usá-lo com outras versões do Apache Airflow.
- Um exemplo de projeto de DBT. Para começar a usar o dbt com o Amazon MWAA, você pode criar uma bifurcação e clonar o projeto dbt starter a partir do repositório dbt-labs. GitHub

Dependências

Para usar o Amazon MWAA com dbt, adicione o seguinte script de inicialização ao seu ambiente. Para saber mais, consulte Uso de um script de inicialização com o Amazon MWAA.

```
#!/bin/bash

if [[ "${MWAA_AIRFLOW_COMPONENT}" != "worker" ]]
then
    exit 0
```

```
fi

echo "------------------------------"
echo "Installing virtual Python env"
echo "------------------------------"

pip3 install --upgrade pip

echo "Current Python version:"
python3 --version
echo "..."

sudo pip3 install --user virtualenv
sudo mkdir python3-virtualenv
cd python3-virtualenv
sudo python3 -m venv dbt-env
sudo chmod -R 777 *

echo "------------------------------"
echo "Activating venv in"
$DBT_ENV_PATH
echo "------------------------------"

source dbt-env/bin/activate
pip3 list

echo "------------------------------"
echo "Installing libraries..."
echo "------------------------------"

# do not use sudo, as it will install outside the venv
pip3 install dbt-redshift==1.6.1 dbt-postgres==1.6.1

echo "------------------------------"
echo "Venv libraries..."
echo "------------------------------"

pip3 list
dbt --version

echo "------------------------------"
echo "Deactivating venv..."
echo "------------------------------"
```

```
deactivate
```

Nas seções a seguir, você fará o upload do diretório do seu projeto de DBT para o Amazon S3 e executará um DAG que valida se o Amazon MWAA instalou com sucesso as dependências DBT necessárias.

Faça o upload de um projeto de DBT para o Amazon S3

Para poder usar um projeto de DBT com seu ambiente Amazon MWAA, é possível fazer o upload todo o diretório do projeto na pasta `dags` do seu ambiente. Quando o ambiente é atualizado, o Amazon MWAA baixa o diretório DBT para a pasta local `usr/local/airflow/dags/`.

Para fazer upload de um projeto de DBT no Amazon S3

1. Navegue até o diretório em que você clonou o projeto inicial de DBT.
2. Execute o seguinte AWS CLI comando do Amazon S3 para copiar recursivamente o conteúdo do projeto para a `dags` pasta do seu ambiente usando o parâmetro. `--recursive` O comando cria um subdiretório chamado `dbt` que é possível usar para todos os seus projetos de DBT. Se o subdiretório já existir, os arquivos do projeto serão copiados para o diretório existente e um novo diretório não será criado. O comando também cria um subdiretório dentro do diretório `dbt` para esse projeto inicial específico.

   ```
   $ aws s3 cp dbt-starter-project s3://mwaa-bucket/dags/dbt/dbt-starter-project --
   recursive
   ```

   É possível usar nomes diferentes para os subdiretórios do projeto para organizar vários projetos de DBT dentro do diretório principal `dbt`.

Use um DAG para verificar a instalação da dependência de DBT

O DAG a seguir usa um código `BashOperator` e um comando bash para verificar se o Amazon MWAA instalou com sucesso as dependências DBT especificadas em `requirements.txt`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
```

```
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="/usr/local/airflow/.local/bin/dbt --version"
    )
```

Faça o seguinte para visualizar os logs de tarefas e verificar se o DBT e as dependências dele foram instalados.

1. Navegue até o console do Amazon MWAA e escolha Abrir IU do Airflow na lista de ambientes disponíveis.
2. Na IU do Apache Airflow, encontre o DAG `dbt-installation-test` na lista e escolha a data na coluna `Last Run` para abrir a última tarefa bem-sucedida.
3. Usando a Exibição de gráfico, escolha a `bash_command` tarefa para abrir os detalhes da instância da tarefa.
4. Escolha Log para abrir os logs de tarefas e, em seguida, verifique se os logs listam com êxito a versão do DBT que especificamos em `requirements.txt`.

Use um DAG para executar um projeto de DBT

O DAG a seguir usa um código `BashOperator` para copiar os projetos de DBT que você fez o upload para o Amazon S3 do diretório local `usr/local/airflow/dags/` para o diretório acessível para gravação `/tmp` e, em seguida, executa o projeto de DBT. Os comandos bash pressupõem um projeto inicial de DBT intitulado `dbt-starter-project`. Modifique o nome do diretório de acordo com o nome do diretório do seu projeto.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

# assumes all files are in a subfolder of DAGs called dbt

with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False, start_date=days_ago(1))
 as dag:
    cli_command = BashOperator(
        task_id="bash_command",
```

```
        bash_command="source /usr/local/airflow/python3-virtualenv/dbt-env/bin/
activate;\
cp -R /usr/local/airflow/dags/dbt /tmp;\
echo 'listing project files:';\
ls -R /tmp;\
cd /tmp/dbt/mwaa_dbt_test_project;\
/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt run --project-dir /tmp/dbt/
mwaa_dbt_test_project --profiles-dir ..;\
cat /tmp/dbt_logs/dbt.log;\
rm -rf /tmp/dbt/mwaa_dbt_test_project"
    )
```

AWS blogs e tutoriais

- Como trabalhar com o Amazon EKS e o Amazon MWAA for Apache Airflow v2.x

Práticas recomendadas para o Amazon Managed Workflows for Apache Airflow

Este guia descreve as práticas recomendadas ao usar o Amazon Managed Workflows for Apache Airflow.

Tópicos

Ajuste de desempenho para o Apache Airflow no Amazon MWAA

Esta página descreve as práticas recomendadas para ajustar o desempenho de um ambiente Amazon Managed Workflows for Apache Airflow usando Como usar opções de configuração do Apache Airflow no Amazon MWAA.

Sumário

Como adicionar uma opção de configuração do Apache Airflow

O procedimento a seguir descreve as etapas para adicionar uma opção de configuração do Airflow ao seu ambiente.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. Escolha Adicionar configuração personalizada no painel Opções de configuração do Airflow.
6. Escolha uma configuração na lista suspensa e insira um valor, ou digite uma configuração personalizada e insira um valor.
7. Escolha Adicionar configuração personalizada para cada configuração que você deseja adicionar.
8. Escolha Salvar.

Para saber mais, consulte Como usar opções de configuração do Apache Airflow no Amazon MWAA.

Agendador do Apache Airflow

O agendador do Apache Airflow é um componente central do Apache Airflow. Um problema com o agendador pode impedir que os DAGs sejam analisados e as tarefas sejam agendadas. Para obter mais informações sobre o ajuste do agendador do Apache Airflow, consulte Ajustar o desempenho do agendador no site de documentação do Apache Airflow.

Parâmetros

Esta seção descreve as opções de configuração disponíveis para o agendador do Apache Airflow e casos de uso do agendador.

Apache Airflow v2

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	celery.sync_parallelism	1	O número de processos que o Celery Executor usa para sincroniz	Você pode usar essa opção para evitar conflitos de fila limitando os processos que o Celery

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
			ar o estado da tarefa.	Executor usa. Por padrão, um valor é definido como 1 para evitar erros na entrega de registros de tarefas ao CloudWatch Logs. Definir o valor como `0` significa usar o número máximo de processos , mas pode causar erros ao entregar logs de tarefas.

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	scheduler.processor_poll_interval	1	O número de segundos de espera entre o processamento consecutivo do arquivo DAG no “loop” do agendador.	Você pode usar essa opção para liberar o uso da CPU no Agendador ao aumentar o tempo de inatividade do Agendador após terminar de recuperar os resultados da análise do DAG, localizar e enfileirar tarefas e executar tarefas em fila no Executor. Aumentar esse valor consome o número de threads do Agendador executados em um ambiente em `scheduler.parsing_processes` para o Apache Airflow v2 e em `scheduler.max_threads` para

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				o Apache Airflow v1. Isso pode reduzir a capacidade dos Agendadores de analisar DAGs e aumentar o tempo necessário para que os DAGs apareçam no servidor Web.
v2	scheduler.max_dagruns_to_create_per_loop	10	O número máximo de DAGs a serem criados DagRunspor “loop” do Scheduler.	Você pode usar essa opção para liberar recursos para agendar tarefas diminuindo o número máximo de DagRunspara o “loop” do Agendador.

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	scheduler.parsing_processes	2	O número de threads que o Agendador pode executar paralelamente para agendar DAGs.	Você pode usar essa opção para liberar recursos ao diminuir o número de processos que o Agendador executa paralelamente para analisar DAGs. Recomendamos manter esse número baixo se a análise do DAG estiver afetando o agendamento de tarefas. Você deve especificar um valor menor que a contagem de vCPUs em seu ambiente. Para saber mais, consulte Limites.

Limites

Esta seção descreve os limites que você deve considerar ao ajustar os parâmetros padrão do agendador.

scheduler.parsing_processes, scheduler.max_threads

Dois threads são permitidos por vCPU para uma classe de ambiente. Pelo menos um thread deve ser reservado para o agendador de uma classe de ambiente. Se você notar um atraso no agendamento de tarefas, talvez seja necessário aumentar sua classe de ambiente. Por exemplo, um ambiente grande tem uma instância de contêiner Fargate de 4 vCPUs para seu agendador. Isso significa que um máximo total de 7 threads está disponível para uso em outros processos. Ou seja, dois threads multiplicaram quatro vCPUs, menos um para o próprio agendador. O valor que você especifica em `scheduler.max_threads` e não `scheduler.parsing_processes` deve exceder o número de threads disponíveis para uma classe de ambiente (conforme mostrado a seguir):

- mw1.small: não deve exceder 1 thread para outros processos. O thread restante é reservado para o Agendador.
- mw1.medium: não deve exceder 3 threads de outros processos. O thread restante é reservado para o Agendador.
- mw1.large: não deve exceder 7 threads de outros processos. O thread restante é reservado para o Agendador.

Pastas do DAG

O Agendador do Apache Airflow verifica continuamente a pasta DAGs em seu ambiente. Quaisquer arquivos `plugins.zip` contidos ou arquivo Python (`.py`) contendo instruções de importação “airflow”. Todos os objetos Python DAG resultantes são então colocados em um arquivo DagBagpara que esse arquivo seja processado pelo Agendador para determinar quais tarefas, se houver, precisam ser agendadas. A análise do arquivo Dag ocorre independentemente de os arquivos conterem objetos DAG viáveis.

Parâmetros

Esta seção descreve as opções de configuração disponíveis para a pasta DAGs e os casos de uso dela.

Apache Airflow v2

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	scheduler.dag_dir_list_interval	300 segundos	O número de segundos em que a pasta DAGs deve ser examinada em busca de novos arquivos.	Você pode usar essa opção para liberar recursos ao aumentar o número de segundos para analisar a pasta DAGs. Recomendamos aumentar esse valor se você estiver vendo longos tempos de análise em `total_parse_time` `metrics`, o qual pode ser em virtude de um grande número de arquivos na sua pasta DAGs.
v2	scheduler.min_file_process_interval	30 segundos	O número de segundos após os quais o agendador analisa um DAG e as atualizações do DAG são refletidas.	Você pode usar essa opção para liberar recursos ao aumentar o número de segundos que o agendador espera antes

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				de analisar um DAG. Por exemplo, se você especificar um valor de `30`, o arquivo DAG será analisado a cada 30 segundos. Recomendamos manter esse número alto para diminuir o uso da CPU em seu ambiente.

Arquivos DAG

Como parte do loop do agendador do Apache Airflow, arquivos DAG individuais são analisados para extrair objetos do DAG Python. No Apache Airflow v2 e superior, o agendador analisa um número máximo de processos de análise ao mesmo tempo. O número de segundos especificado em `scheduler.min_file_process_interval` deve passar antes que o mesmo arquivo seja analisado novamente.

Parâmetros

Esta seção descreve as opções de configuração disponíveis para arquivos DAG do Apache Airflow e os casos de uso deles.

Apache Airflow v2

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	core.dag_file_processor_timeout	50 segundos	O número de segundos antes que o DagFileprocessador expire o processamento de um arquivo DAG.	Você pode usar essa opção para liberar recursos aumentando o tempo necessário até que o DagFileprocessador atinja o tempo limite. Recomendamos aumentar esse valor se você estiver vendo tempos limite nos logs de processamento do DAG que resultam na falta de carregamento de DAGs viáveis.
v2	core.dagbag_import_timeout	30 segundos	O tempo limite do número de segundos antes de importar um arquivo do Python.	Você pode usar essa opção para liberar recursos ao aumentar o tempo necessário até que o Agendador atinja o tempo limite ao importar um

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				arquivo Python para extrair os objetos DAG. Essa opção é processada como parte do “loop” do Agendador e deve conter um valor menor que o valor especificado em `core.dag_file_processor_timeout` .

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	core.min_serialized_dag_update_interval	30	O número mínimo de segundos após os quais os DAGs serializados no banco de dados são atualizados.	Você pode usar essa opção para liberar recursos ao aumentar o número de segundos após os quais os DAGs serializados no banco de dados são atualizados. Recomendamos aumentar esse valor se você tiver um número grande de DAGs, ou DAGs complexos. Aumentar esse valor reduz a carga no Agendador e no banco de dados à medida que os DAGs são serializados.

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	core.min_serialized_dag_fetch_interval	10	O número de segundos em que um DAG serializado é recuperado do banco de dados quando já está carregado no. DagBag	Você pode usar essa opção para liberar recursos ao aumentar o número de segundos em que um DAG serializado é recuperado. O valor deve ser maior que o valor especificado em `core.min_serialized_dag_update_interval` para reduzir as taxas de “gravação” do banco de dados. Aumentar esse valor reduz a carga no servidor Web e no banco de dados à medida que os DAGs são serializados.

Tarefas

Tanto o agendador quanto os operadores do Apache Airflow estão envolvidos em tarefas de enfileiramento e desenfileiramento. O agendador leva as tarefas analisadas prontas para serem agendadas de um status vazio para um status agendado. O executor, também em execução no contêiner do agendador no Fargate, coloca essas tarefas em fila e define o status como Em fila. Quando os operadores têm capacidade, retiram a tarefa da fila e definem o status como executando, que posteriormente muda o status para Êxito ou Falha com base no sucesso ou falha da tarefa.

Parâmetros

Esta seção descreve as opções de configuração disponíveis para tarefas do Apache Airflow e os casos de uso delas.

As opções de configuração padrão que o Amazon MWAA substitui estão marcadas *em vermelho.*

Apache Airflow v2

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	core.parallelism	10000	O número máximo de instâncias de tarefas que podem ter o status “Em execução”.	Você pode usar essa opção para liberar recursos ao aumentar o número de instâncias de tarefas que podem ser executada s simultane amente. O valor especific ado deve ser o número de Operadore s disponíve is “vezes” a

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				densidade de tarefas dos Operadores. Recomendamos alterar esse valor somente quando você estiver vendo um grande número de tarefas bloqueadas no estado “Em execução” ou “Em fila”.

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	core.dag_concurrency	10000	O número de instâncias de tarefas que podem ser executadas simultaneamente para cada DAG.	Você pode usar essa opção para liberar recursos ao aumentar o número de instâncias de tarefas autorizadas a serem executadas simultaneamente. Por exemplo, se você tiver cem DAGs com dez tarefas paralelas e quiser que todos os DAGs sejam executados simultaneamente, é possível calcular o paralelismo máximo como o número de Operadores disponíveis “vezes” a densidade de tarefas em `celery.worker_conc`

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				`urrency` dos Operadores, dividido pelo número de DAGs (por exemplo, 100).
v2	core.execute_tasks_new_python_interpreter	`True`	Determina se o Apache Airflow executa tarefas bifurcando o processo pai ou criando um novo processo em Python.	Quando definido como `True`, o Apache Airflow reconhece as alterações feitas nos seus plug-ins como um novo processo Python criado para executar tarefas.
v2	celery.worker_concurrency	N/D	O Amazon MWAA substitui a instalação básica do Airflow por essa opção para escalar Operadores como parte de seu componente de escalonamento automático.	*`Qualquer valor especificado para essa opção é ignorado.`*

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
v2	celery.worker_autoscale	mw1.small: 5,0 mw1.medium: 10,0 mw1.large: 20,0 mw1.xlarge - 40,0 mw1.2xlarge - 80,0	A simultaneidade de tarefas para Operadores.	Você pode usar essa opção para liberar recursos ao reduzir a simultaneidade `minimum`, `maximum` de tarefas dos Operadores. Os Operadores aceitam até `maximum` tarefas simultâneas configuradas, independentemente de haver recursos suficientes para fazer isso. Se as tarefas forem agendadas sem recursos suficientes, elas falharão imediatamente. Recomendamos alterar esse valor em tarefas que consomem muitos recursos ao reduzir os valores para

Version (Versão)	Opção de configuração	Padrão	Descrição	Caso de uso
				serem menores que os padrões a fim de permitir mais capacidad e por tarefa.

Como gerenciar dependências do Python em requirements.txt

Esta página descreve as práticas recomendadas para instalar e gerenciar dependências do Python em um arquivo `requirements.txt` para um ambiente Amazon Managed Workflows for Apache Airflow.

Sumário

Como testar DAGs usando o utilitário Amazon MWAA CLI

- O utilitário da interface de linha de comandos (CLI) replica localmente um ambiente do Amazon Managed Workflows for Apache Airflow.
- A CLI cria localmente uma imagem de contêiner Docker semelhante a uma imagem de produção do Amazon MWAA. Isso permite que você execute um ambiente Apache Airflow local para desenvolver e testar DAGs, plug-ins personalizados e dependências antes da implantação no Amazon MWAA.
- Para executar a CLI, consulte o aws-mwaa-local-runner em. GitHub

Instalando dependências do Python usando o formato de arquivo de requisitos PyPi .org

A seção a seguir descreve as diferentes maneiras de instalar dependências do Python de acordo com o Formato de Arquivo de Requisitos PyPi .org.

Opção 1: dependências do Python do Python Package Index

A seção a seguir descreve como especificar dependências do Python do Python Package Index em um arquivo `requirements.txt`.

Apache Airflow v2

1. Testar localmente. Adicione bibliotecas adicionais de forma iterativa para encontrar a combinação certa de pacotes e suas versões, antes de criar um arquivo `requirements.txt`. Para executar o utilitário Amazon MWAA CLI, consulte o aws-mwaa-local-runner em. GitHub
2. Revise os extras do pacote Apache Airflow. Para ver uma lista dos pacotes instalados para o Apache Airflow v2 no Amazon MWAA, consulte Amazon MWAA local runner no site. `requirements.txt` GitHub
3. Adicione uma declaração de restrições. Adicione o arquivo de restrições do seu ambiente Apache Airflow v2 na parte superior do seu arquivo `requirements.txt`. Os arquivos de restrições do Apache Airflow especificam as versões do provedor disponíveis no momento de um lançamento do Apache Airflow.

   A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para

garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.

No exemplo a seguir, substitua *{environment-version}* pelo número da versão do seu ambiente e *{Python-version} pela versão do Python compatível com* seu ambiente.

Para obter informações sobre a versão do Python compatível com seu ambiente Apache Airflow, consulte Versões do Apache Airflow.

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

Se o arquivo de restrições determinar que o pacote `xyz==1.0` não é compatível com outros pacotes em seu ambiente, `pip3 install` não conseguirá impedir que bibliotecas incompatíveis sejam instaladas em seu ambiente. Se a instalação falhar em algum pacote, você poderá visualizar os registros de erros de cada componente do Apache Airflow (o agendador, o trabalhador e o servidor web) no fluxo de registros correspondente em Logs. CloudWatch Para obter mais informações sobre os tipos de log, consulte the section called "Como visualizar logs do Airflow".

4. Pacotes do Apache Airflow. Adicione os extras do pacote e a versão (==). Isso ajuda a evitar que pacotes com o mesmo nome, mas com versões diferentes, sejam instalados em seu ambiente.

```
apache-airflow[package-extra]==2.5.1
```

5. Bibliotecas Python. Adicione o nome do pacote e a versão (==) em seu arquivo `requirements.txt`. Isso ajuda a evitar que uma atualização futura de última hora do PyPidomínio.org seja aplicada automaticamente.

```
library == version
```

Example Boto3 e psycopg2-binary

Esse exemplo de código é fornecido para fins de demonstração. As bibliotecas boto e psycopg2-binary estão incluídas na instalação básica do Apache Airflow v2 e não precisam ser especificadas em um arquivo `requirements.txt`.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Se um pacote for especificado sem uma versão, o Amazon MWAA instalará a versão mais recente do pacote em .org. PyPi Esta versão pode entrar em conflito com outros pacotes em seu `requirements.txt`.

Apache Airflow v1

1. Testar localmente. Adicione bibliotecas adicionais de forma iterativa para encontrar a combinação certa de pacotes e suas versões, antes de criar um arquivo `requirements.txt`. Para executar o utilitário Amazon MWAA CLI, consulte o aws-mwaa-local-runner em. GitHub
2. Revise os extras do pacote Airflow. Revise a lista de pacotes disponíveis para o Apache Airflow v1.10.12 em https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt.
3. Adicione o arquivo de restrições. Adicione o arquivo de restrições do Apache Airflow v1.10.12 na parte superior do seu arquivo `requirements.txt`. Se o arquivo de restrições determinar que o pacote `xyz==1.0` não é compatível com outros pacotes em seu ambiente, `pip3 install` não conseguirá impedir que bibliotecas incompatíveis sejam instaladas em seu ambiente.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-1.10.12/constraints-3.7.txt"
   ```
4. Pacotes do Apache Airflow v1.10.12. Adicione os extras do pacote Airflow e a versão v1.10.12 do Apache Airflow (==). Isso ajuda a evitar que pacotes com o mesmo nome, mas com versões diferentes, sejam instalados em seu ambiente.

   ```
   apache-airflow[package]==1.10.12
   ```

   Example Secure Shell (SSH)

   O arquivo `requirements.txt` de exemplo a seguir, instala o SSH para o Apache Airflow v1.10.12.

```
apache-airflow[ssh]==1.10.12
```

5. Bibliotecas Python. Adicione o nome do pacote e a versão (==) em seu arquivo `requirements.txt`. Isso ajuda a evitar que uma atualização futura de última hora do PyPidomínio.org seja aplicada automaticamente.

   ```
   library == version
   ```

   Example Boto3

   O arquivo `requirements.txt` de exemplo a seguir, instala a biblioteca Boto3 para o Apache Airflow v1.10.12.

   ```
   boto3 == 1.17.4
   ```

   Se um pacote for especificado sem uma versão, o Amazon MWAA instalará a versão mais recente do pacote em .org. PyPi Esta versão pode entrar em conflito com outros pacotes em seu `requirements.txt`.

Opção dois: wheel do Python (.whl)

O wheel do Python é um formato de pacote projetado para enviar bibliotecas com artefatos compilados. Há vários benefícios em usar pacotes wheel como um método para instalar dependências no Amazon MWAA:

- Instalação mais rápida: os arquivos WHL são copiados para o contêiner como um único ZIP e depois instalados localmente, sem precisar baixar um por um.
- Menos conflitos: você pode determinar a compatibilidade de versões de seus pacotes com antecedência. Como resultado, não há necessidade de `pip` para elaborar recursivamente versões compatíveis.
- Mais resiliência: com bibliotecas hospedadas externamente, os requisitos posteriores podem mudar, resultando em incompatibilidade de versões entre contêineres em um ambiente do Amazon MWAA. Por não depender de uma fonte externa para dependências, cada contêiner tem as mesmas bibliotecas, independentemente de quando cada contêiner é instanciado.

Recomendamos os seguintes métodos para instalar dependências do Python a partir de um arquivo wheel do Python (`.whl`) no seu arquivo `requirements.txt`.

Métodos

- Como usar o arquivo plugins.zip em um bucket do Amazon S3
- Como usar um arquivo WHL hospedado em uma URL
- Como criar arquivos WHL a partir de um DAG

Como usar o arquivo **`plugins.zip`** em um bucket do Amazon S3

O programador do Apache Airflow, os trabalhadores e o servidor web (para o Apache Airflow v2.2.2 e versões posteriores) procuram plug-ins personalizados durante a inicialização no contêiner Fargate gerenciado AWS para seu ambiente em. `/usr/local/airflow/plugins/*` Esse processo começa antes de `pip3 install -r requirements.txt` do Amazon MWAA para as dependências do Python e da inicialização do serviço no Apache Airflow. Um arquivo `plugins.zip` pode ser usado para qualquer arquivo que você não queira que seja alterado continuamente durante a execução do ambiente ou que talvez não queira conceder acesso aos usuários que gravem DAGs. Por exemplo, arquivos de wheel de biblioteca Python, arquivos PEM de certificado e arquivos YAML de configuração.

A seção a seguir descreve como instalar um wheel que está no arquivo `plugins.zip` em seu bucket do Amazon S3.

1. Baixe os arquivos WHL necessários Você pode usar `pip download` com seu arquivo `requirements.txt` existente no executor local do Amazon MWAA ou em outro contêiner do Amazon Linux 2 para solucionar e baixar os arquivos necessários do wheel do Python.

   ```
   $ pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
   $ cd "$AIRFLOW_HOME/plugins"
   $ zip "$AIRFLOW_HOME/plugins.zip" *
   ```

2. Especifique o caminho em seu **`requirements.txt`**. Especifique o diretório de plug-ins na parte superior do seu requirements.txt usando `--find-links`e instrua `pip` a não instalar de outras fontes usando `--no-index`, conforme mostrado no seguinte

   ```
   --find-links /usr/local/airflow/plugins
   --no-index
   ```

Example wheel em requirements.txt

O exemplo a seguir pressupõe que você tenha carregado o wheel em um arquivo `plugins.zip` na raiz do seu bucket do Amazon S3. Por exemplo: .

```
--find-links /usr/local/airflow/plugins
--no-index

numpy
```

O Amazon MWAA busca o wheel `numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl` da pasta `plugins` e o instala em seu ambiente.

Como usar um arquivo WHL hospedado em uma URL

A seção a seguir descreve como instalar um wheel hospedado em uma URL. A URL deve ser acessível publicamente ou de dentro do Amazon VPC personalizado que você especificou para seu ambiente do Amazon MWAA.

- Forneça um URL. Forneça o URL de um wheel em seu arquivo `requirements.txt`.

  Example arquivo wheel em um URL público

  O exemplo a seguir baixa um wheel de um site público.

  ```
  --find-links https://files.pythonhosted.org/packages/
  --no-index
  ```

  O Amazon MWAA busca o wheel a partir da URL que você especificou e a instala em seu ambiente.

  > **Note**
  >
  > Os URLs não são acessíveis a partir de servidores Web privados que instalam requisitos no Amazon MWAA v2.2.2 e posterior.

Como criar arquivos WHL a partir de um DAG

Se você tem um servidor web privado usando o Apache Airflow v2.2.2 ou posterior e não consegue instalar os requisitos porque seu ambiente não tem acesso a repositórios externos, você pode usar o seguinte DAG para pegar seus requisitos existentes do Amazon MWAA e empacotá-los no Amazon S3:

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

S3_BUCKET = 'my-s3-bucket'
S3_KEY = 'backup/plugins_whl.zip'

with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/
requirements/requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3
 cp /tmp/plugins.zip s3://{S3_BUCKET}/{S3_KEY}"
    )
```

Depois de executar o DAG, use esse novo arquivo como seu `plugins.zip` do Amazon MWAA, opcionalmente, empacotado com outros plug-ins. Em seguida, atualize seu `requirements.txt` precedido por `--find-links /usr/local/airflow/plugins` e `--no-index` sem adicionar`--constraint`.

Esse método permite que você use as mesmas bibliotecas off-line.

Opção três: dependências do Python hospedadas em um repositório privado PyPi compatível com /PEP-503

A seção a seguir descreve como instalar um extra do Apache Airflow hospedado em uma URL privada com autenticação.

1. Adicione seu nome de usuário e senha como opções de configuração do Apache Airflow. Por exemplo: .

   - `foo.user` : *YOUR_USER_NAME*
   - `foo.pass` : *YOUR_PASSWORD*

2. crie seu arquivo `requirements.txt`. Substitua os espaços reservados no exemplo a seguir pelo seu URL privado e pelo nome de usuário e senha que você adicionou como opções de configuração do Apache Airflow. Por exemplo: .

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
```

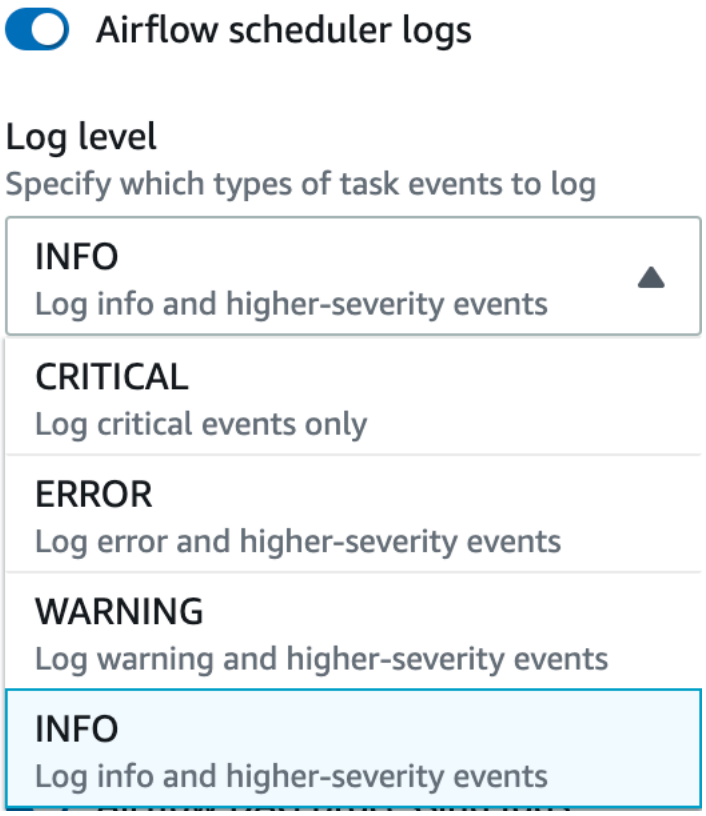
3. adicione outras bibliotecas ao seu arquivo `requirements.txt`. Por exemplo: .

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
my-private-package==1.2.3
```

Como habilitar logs no console do Amazon MWAA

A função de execução do seu ambiente Amazon MWAA precisa de permissão para enviar registros para CloudWatch Logs. Para atualizar as permissões de um perfil de execução, consulte Perfil de execução do Amazon MWAA.

Você pode ativar os logs do Apache Airflow no nível `INFO`, `WARNING`, `ERROR` e `CRITICAL`. Quando você escolhe um nível de log, o Amazon MWAA envia logs desse nível e de todos os níveis mais altos de severidade. Por exemplo, se você habilitar registros no `INFO` nível, o Amazon MWAA enviará `INFO` registros e `WARNINGERROR`, e níveis de `CRITICAL` log para CloudWatch Logs. Recomendamos habilitar os logs do Apache Airflow no nível `INFO` do Agendador para visualizar os logs recebidos para `requirements.txt`.



Visualização de registros no console CloudWatch de registros

Você pode visualizar os registros do Apache Airflow para o Agendador ao agendar seus fluxos de trabalho e analisar sua pasta `dags`. As etapas a seguir descrevem como abrir o grupo de logs para o Scheduler no console Amazon MWAA e visualizar os registros do Apache Airflow no console Logs. CloudWatch

Para visualizar os logs de um **`requirements.txt`**

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha grupo de logs de agendador do Airflow no painel Monitoramento.
4. Escolha o log `requirements_install_ip` em Fluxos de logs.
5. Você deve ver a lista de pacotes que foram instalados no ambiente em `/usr/local/airflow/.local/bin`. Por exemplo: .

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
```

```
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Analise a lista de pacotes e verifique se algum deles encontrou algum erro durante a instalação. Se algo der errado, é possível ver um erro semelhante ao seguinte:

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

Como visualizar erros na IU do Apache Airflow

Você também pode verificar sua IU do Apache Airflow para identificar se um erro pode estar relacionado a outro problema. O erro mais comum que você pode encontrar com o Apache Airflow no Amazon MWAA é:

```
Broken DAG: No module named x
```

se você ver esse erro na IU do Apache Airflow, provavelmente está faltando uma dependência necessária em seu arquivo `requirements.txt`.

Fazendo login no Apache Airflow

Você precisa de Política de acesso ao Apache Airflow UI: AmazonMWA Access WebServer permissões para sua AWS conta no AWS Identity and Access Management (IAM) para visualizar sua interface de usuário do Apache Airflow.

Para acessar sua IU do Apache Airflow

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha Abrir a IU do Airflow.

Cenários de exemplo de **requirements.txt**

Você pode misturar e combinar diferentes formatos no seu `requirements.txt`. O exemplo a seguir usa uma combinação das diferentes formas de instalar extras.

Example Extras em PyPi .org e um URL público

Você precisa usar a `--index-url` opção ao especificar pacotes de PyPi .org, além de pacotes em uma URL pública, como URLs de repositório personalizados compatíveis com PEP 503.

```
aws-batch == 0.6
phoenix-letter >= 0.3

--index-url http://dist.repoze.org/zope2/2.10/simple
    zopelib
```

Monitoramento e métricas para o Amazon Managed Workflows for Apache Airflow

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho dos fluxos de trabalho gerenciados da Amazon para Apache Airflow e sua solução. AWS Recomendamos coletar dados de monitoramento de todas as partes da sua AWS solução para que você possa depurar com mais facilidade uma falha multiponto, caso ocorra. Este tópico descreve quais recursos AWS fornecem para monitorar seu ambiente Amazon MWAA e responder a possíveis eventos.

> **Note**
>
> As métricas e os registros do Apache Airflow estão sujeitos aos preços padrão da Amazon CloudWatch .

Para obter mais informações sobre o monitoramento do Apache Airflow, consulte Registro em log e monitoramento no site de documentação do Apache Airflow.

Seções

Visão geral do monitoramento no Amazon MWAA

Esta página descreve os AWS serviços usados para monitorar um ambiente Amazon Managed Workflows for Apache Airflow.

Sumário

- AWS CloudTrail visão geral

CloudWatch Visão geral da Amazon

CloudWatch é um repositório de métricas para AWS serviços que permite recuperar estatísticas com base nas métricas e dimensões publicadas por um serviço. Você pode usar essas métricas para configurar alarmes, calcular estatísticas e, em seguida, apresentar os dados em um painel que ajuda a avaliar a integridade do seu ambiente no CloudWatch console da Amazon.

O Apache Airflow já está configurado para enviar métricas StatsD para um ambiente Amazon Managed Workflows for Apache Airflow para a Amazon. CloudWatch

Para saber mais, consulte O que é a Amazon CloudWatch? .

AWS CloudTrail visão geral

CloudTrail é um serviço de auditoria que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Amazon MWAA. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita à Amazon MWAA, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais disponíveis nos registros de auditoria.

Para saber mais, consulte O que é AWS CloudTrail? .

Visualizando registros de auditoria em AWS CloudTrail

AWS CloudTrail é ativado em sua AWS conta quando você a cria. CloudTrail registra a atividade realizada por uma entidade ou AWS serviço do IAM, como Amazon Managed Workflows for Apache Airflow, que é registrada como um evento. CloudTrail Você pode visualizar, pesquisar e baixar os últimos 90 dias do histórico de eventos no CloudTrail console. CloudTrail captura todos os eventos no console do Amazon MWAA e todas as chamadas para as APIs do Amazon MWAA. Ele não captura ações somente para leitura, como `GetEnvironment`, ou ação `PublishMetrics`. Esta página descreve como usar para monitorar eventos CloudTrail para o Amazon MWAA.

Sumário

Criando uma trilha em CloudTrail

Você precisa criar uma trilha para visualizar um registro contínuo de eventos em sua AWS conta, incluindo eventos para o Amazon MWAA. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Se você não criar uma trilha, ainda poderá ver o histórico de eventos disponível no CloudTrail console. Por exemplo, usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita à Amazon MWAA, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais. Para saber mais, consulte Criação de uma trilha para sua AWS conta.

Visualizando eventos com o histórico de CloudTrail eventos

Você pode solucionar incidentes operacionais e de segurança nos últimos 90 dias no CloudTrail console visualizando o histórico de eventos. Por exemplo, você pode visualizar eventos relacionados à criação, modificação ou exclusão de recursos (como usuários do IAM ou outros AWS recursos) em sua AWS conta por região. Para saber mais, consulte Visualização de eventos com histórico de CloudTrail eventos.

1. Abra o console do CloudTrail.
2. Escolha Histórico de eventos.
3. Selecione os eventos que quer visualizar e, em seguida, escolha Comparar detalhes do evento.

Exemplo de trilha para **`CreateEnvironment`**

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket Amazon S3 especificado.

CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, como a data e a hora da ação ou os parâmetros da solicitação. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API e não aparecem em nenhuma ordem específica. O exemplo a seguir é uma entrada de log para a ação `CreateEnvironment`, que foi negada devido à falta de permissões. Os valores em `AirflowConfigurationOptions` foram alterados para fins de privacidade.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "00123456ABC7DEF8HIJK",
        "arn": "arn:aws:sts::012345678901:assumed-role/root/myuser",
        "accountId": "012345678901",
        "accessKeyId": "",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "00123456ABC7DEF8HIJK",
                "arn": "arn:aws:iam::012345678901:role/user",
                "accountId": "012345678901",
                "userName": "user"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-10-07T15:51:52Z"
            }
        }
    },
    "eventTime": "2020-10-07T15:52:58Z",
    "eventSource": "airflow.amazonaws.com",
    "eventName": "CreateEnvironment",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.178",
    "userAgent": "PostmanRuntime/7.26.5",
    "errorCode": "AccessDenied",
    "requestParameters": {
        "SourceBucketArn": "arn:aws:s3:::my-bucket",
        "ExecutionRoleArn": "arn:aws:iam::012345678901:role/AirflowTaskRole",
        "AirflowConfigurationOptions": "***",
        "DagS3Path": "sample_dag.py",
        "NetworkConfiguration": {
            "SecurityGroupIds": [
                "sg-01234567890123456"
            ],
            "SubnetIds": [
                "subnet-01234567890123456",
                "subnet-65432112345665431"
            ]
```

```
        },
        "Name": "test-cloudtrail"
    },
    "responseElements": {
        "message": "Access denied."
    },
    "requestID": "RequestID",
    "eventID": "EventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "012345678901"
}
```

Próximas etapas

- Saiba como configurar outros AWS serviços para os dados de eventos coletados em CloudTrail registros em Serviços e integrações CloudTrail compatíveis.
- Saiba como ser notificado ao CloudTrail publicar novos arquivos de log em um bucket do Amazon S3 em Configurar notificações do Amazon SNS para. CloudTrail

Visualizando registros de fluxo de ar na Amazon CloudWatch

O Amazon MWAA pode enviar registros do Apache Airflow para a Amazon. CloudWatch É possível visualizar logs de vários ambientes em um único local para facilmente identificar atrasos nas tarefas do Apache Airflow ou erros no fluxo de trabalho sem a necessidade de outras ferramentas de terceiros. Os registros do Apache Airflow precisam estar habilitados no console Amazon Managed Workflows for Apache Airflow para visualizar o processamento, as tarefas, o servidor Web e os logins do Worker do Apache Airflow DAG. CloudWatch

Sumário

Definição de preço

- Aplicam-se taxas de CloudWatch registros padrão. Para obter mais informações, consulte CloudWatch os preços.

Antes de começar

- Você deve ter uma função que possa visualizar os logins CloudWatch. Para ter mais informações, consulte Como acessar um ambiente do Amazon MWAA.

Tipos de log

O Amazon MWAA cria um grupo de registros para cada opção de registro do Airflow que você habilita e envia os registros para os grupos de registros associados a CloudWatch um ambiente. O grupo de logs é nomeado no seguinte formato: `YourEnvironmentName-`**`LogType`**. Por exemplo, se seu ambiente foi nomeado como `Airflow-v202-Public`, os logs de tarefas do Apache Airflow serão enviados para `Airflow-v202-Public-`**`Task`**.

Tipo de log	Descrição
`YourEnvironmentName-` **`DAGProcesing`**	Os logs do gerenciador do processador do DAG (a parte do agendador que processa os arquivos do DAG).
`YourEnvironmentName-` **`Scheduler`**	Os logs gerados pelo agendador do Airflow.
`YourEnvironmentName-` **`Task`**	Os logs de tarefas gerados por um DAG.
`YourEnvironmentName-` **`WebServer`**	Os logs gerados pela interface web do Airflow.
`YourEnvironmentName-` **`Worker`**	Os logs gerados como parte do fluxo de trabalho e da execução do DAG.

Como habilitar registros do Apache Airflow

Você pode ativar os logs do Apache Airflow no nível `INFO`, `WARNING`, `ERROR` e `CRITICAL`. Quando você escolhe um nível de log, o Amazon MWAA envia logs desse nível e de todos os níveis mais

altos de severidade. Por exemplo, se você habilitar registros no `INFO` nível, o Amazon MWAA enviará `INFO` registros e `WARNINGERROR`, e níveis de `CRITICAL` log para CloudWatch Logs.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Selecione a opção Editar.
4. Escolha Próximo.
5. Escolha uma ou mais das seguintes opções de registro em log:
    a. Escolha grupo de logs de agendador do Airflow no painel Monitoramento.
    b. Escolha Grupo de logs do servidor web no Airflow no painel Monitoramento.
    c. Escolha Grupo de logs do operador no Airflow no painel Monitoramento.
    d. Escolha Grupo de logs de processamento de DAG no Airflow no painel Monitoramento.
    e. Escolha Grupo de logs de tarefa no Airflow no painel Monitoramento.
    f. Escolha o nível de registro em log em Nível de log.
6. Selecione Next (Próximo).
7. Selecione Save (Salvar).

Como visualizar logs do Apache Airflow

A seção a seguir descreve como visualizar os registros do Apache Airflow no CloudWatch console.

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha um grupo de logs no painel Monitoramento.
4. Escolha um log em Fluxo de logs.

Exemplos de logs do agendador

Você pode visualizar os registros do Apache Airflow para o Agendador ao agendar seus fluxos de trabalho e analisar sua pasta `dags`. As etapas a seguir descrevem como abrir o grupo de logs para o Scheduler no console Amazon MWAA e visualizar os registros do Apache Airflow no console Logs. CloudWatch

Para visualizar os logs de um **`requirements.txt`**

1. Abra a página Ambientes no console do Amazon MWAA.
2. Escolha um ambiente.
3. Escolha grupo de logs de agendador do Airflow no painel Monitoramento.
4. Escolha o log `requirements_install_ip` em Fluxos de logs.
5. Você deve ver a lista de pacotes que foram instalados no ambiente em `/usr/local/airflow/.local/bin`. Por exemplo: .

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Analise a lista de pacotes e verifique se algum deles encontrou algum erro durante a instalação. Se algo der errado, é possível ver um erro semelhante ao seguinte:

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

Próximas etapas

- Saiba como configurar um CloudWatch alarme em Usando CloudWatch alarmes da Amazon.
- Saiba como criar um CloudWatch painel em Usando CloudWatch painéis.

Painéis de monitoramento e alarmes no Amazon MWAA

Você pode criar um painel personalizado na Amazon CloudWatch e adicionar alarmes para uma métrica específica para monitorar o status de saúde de um ambiente Amazon Managed Workflows for Apache Airflow. Quando há um alarme em um painel, ele fica vermelho quando está no estado `ALARM`, facilitando o monitoramento proativo da integridade de um ambiente Amazon MWAA.

O Apache Airflow expõe métricas para vários processos, incluindo o número de processos do DAG, o tamanho do pacote do DAG, as tarefas atualmente em execução, as falhas e os sucessos das tarefas. Quando você cria um ambiente, o Airflow é configurado para enviar automaticamente métricas de um ambiente Amazon MWAA para. CloudWatch Esta página descreve como criar um painel de status de saúde para as métricas do Airflow em um CloudWatch ambiente Amazon MWAA.

Sumário

Metrics

Você pode criar um painel e um alarme personalizados para qualquer uma das métricas disponíveis para sua versão do Apache Airflow. Cada métrica corresponde a um indicador-chave de desempenho (KPI) do Apache Airflow. Para visualizar uma lista de métricas, consulte:

- Métricas do ambiente Apache Airflow v2 em CloudWatch

Visão geral dos estados de alarme

Um alarme de métrica tem estes estados possíveis:

- `OK`: a métrica ou a expressão está dentro do limite definido.
- `ALARM`: a métrica ou a expressão está fora do limite definido.
- `INSUFFICIENT_DATA`: o alarme acabou de ser acionado, a métrica não está disponível ou não há dados suficientes para a métrica determinar o estado do alarme.

Exemplos de painéis e alarmes personalizados

Você pode criar um painel de monitoramento personalizado que exibe gráficos de métricas selecionadas para seu ambiente do Amazon MWAA.

Sobre essas métricas

A lista a seguir descreve cada uma das métricas criadas no painel personalizado pelas definições do tutorial e do modelo nesta seção.

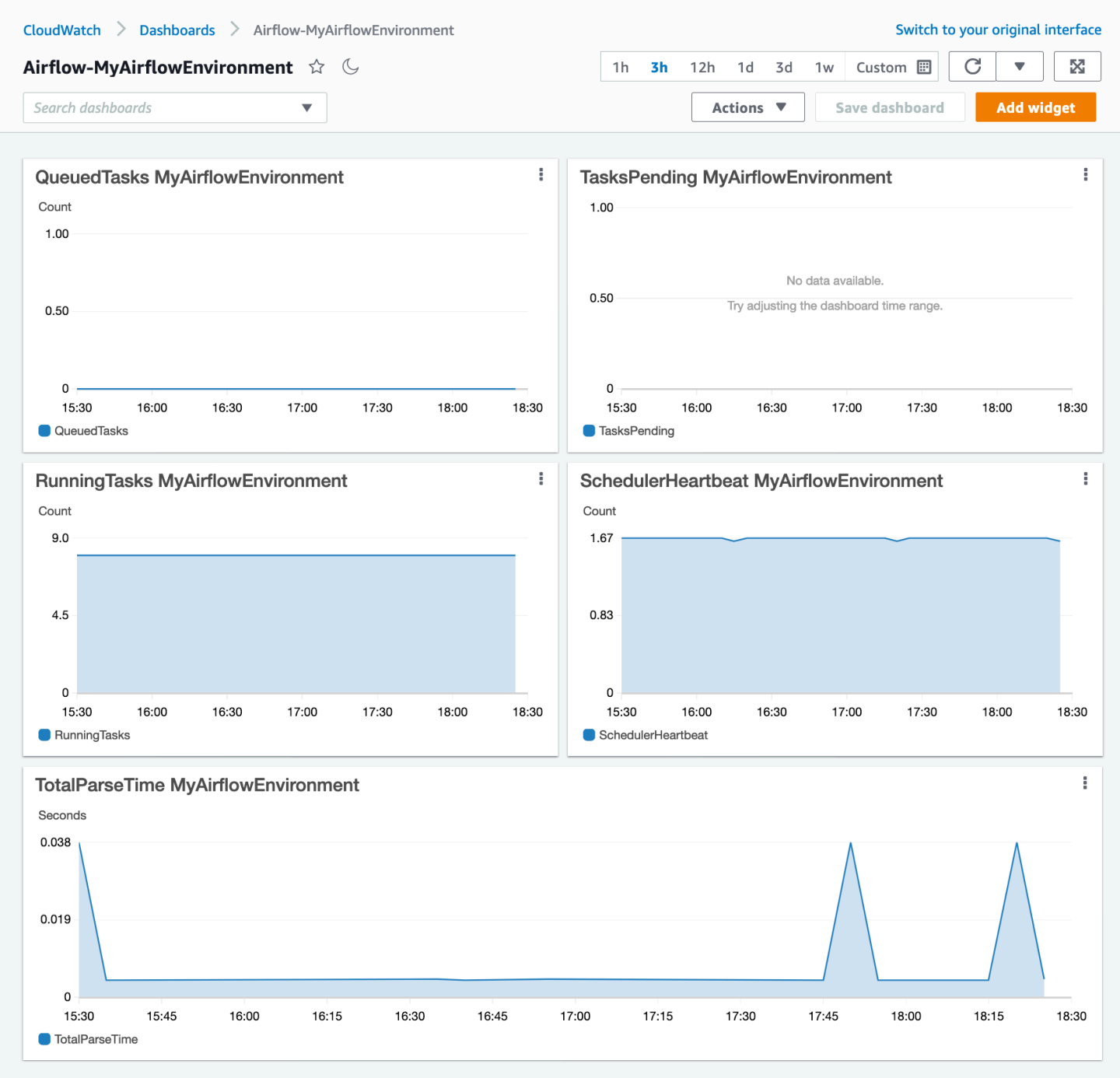
- QueuedTasks- O número de tarefas com estado em fila. Corresponde à métrica `executor.queued_tasks` do Apache Airflow.
- TasksPending- O número de tarefas pendentes no executor. Corresponde à métrica `scheduler.tasks.pending` do Apache Airflow.

> **Note**
>
> Não se aplica ao Apache Airflow v2.2 e superior.

- RunningTasks- O número de tarefas em execução no executor. Corresponde à métrica `executor.running_tasks` do Apache Airflow.
- SchedulerHeartbeat- O número de check-ins que o Apache Airflow realiza no trabalho do agendador. Corresponde às métricas `scheduler_heartbeat` do Apache Airflow.
- TotalParseTempo - O número de segundos necessários para digitalizar e importar todos os arquivos DAG uma vez. Corresponde à métrica `dag_processing.total_parse_time` do Apache Airflow.

Sobre o painel

A imagem a seguir mostra o painel de monitoramento criado pelo tutorial e pela definição do modelo nesta seção.



Usando AWS tutoriais

Você pode usar o AWS tutorial a seguir para criar automaticamente um painel de status de saúde para qualquer ambiente Amazon MWAA que esteja implantado atualmente. Ele também cria CloudWatch alarmes para trabalhadores insalubres e falhas de pulsação do agendador em todos os ambientes do Amazon MWAA.

- CloudWatch Automação de painéis para Amazon MWAA

Usando AWS CloudFormation

Você pode usar a definição do AWS CloudFormation modelo nesta seção para criar um painel de monitoramento e CloudWatch, em seguida, adicionar alarmes no CloudWatch console para receber notificações quando uma métrica ultrapassar um limite específico. Para criar a pilha usando essa definição de modelo, consulte Criação de uma pilha no AWS CloudFormation console. Para adicionar um alarme ao painel, consulte Como usar alarmes.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Creates MWAA Cloudwatch Dashboard
Parameters:
  DashboardName:
    Description: Enter the name of the CloudWatch Dashboard
    Type: String
  EnvironmentName:
    Description: Enter the name of the MWAA Environment
    Type: String
Resources:
  BasicDashboard:
    Type: AWS::CloudWatch::Dashboard
    Properties:
      DashboardName: !Ref DashboardName
      DashboardBody:
        Fn::Sub: '{
              "widgets": [
                  {
                      "type": "metric",
                      "x": 0,
                      "y": 0,
                      "width": 12,
                      "height": 6,
                      "properties": {
                          "view": "timeSeries",
                          "stacked": true,
                          "metrics": [
                              [
                                  "AmazonMWAA",
                                  "QueuedTasks",
                                  "Function",
                                  "Executor",
```

```
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "QueuedTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 0,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "RunningTasks",
                        "Function",
                        "Executor",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "RunningTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 12,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
```

```
                        "AmazonMWAA",
                        "SchedulerHeartbeat",
                        "Function",
                        "Scheduler",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "SchedulerHeartbeat ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 12,
            "y": 0,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "TasksPending",
                        "Function",
                        "Scheduler",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "TasksPending ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 0,
            "y": 12,
            "width": 24,
            "height": 6,
            "properties": {
```

```
                    "view": "timeSeries",
                    "stacked": true,
                    "region": "${AWS::Region}",
                    "metrics": [
                        [
                            "AmazonMWAA",
                            "TotalParseTime",
                            "Function",
                            "DAG Processing",
                            "Environment",
                            "${EnvironmentName}"
                        ]
                    ],
                    "title": "TotalParseTime  ${EnvironmentName}",
                    "period": 300
                }
            }
        ]
    }'
```

Como apagar métricas e painéis

Se você excluir um ambiente Amazon MWAA, o painel correspondente também será excluído. CloudWatch as métricas são armazenadas por quinze (15) meses e não podem ser excluídas. O CloudWatch console limita a pesquisa de métricas a duas (2) semanas após a última ingestão de uma métrica para garantir que as instâncias mais atualizadas sejam mostradas para seu ambiente Amazon MWAA. Para saber mais, consulte as CloudWatch perguntas frequentes da Amazon.

Próximas etapas

- Saiba como criar um DAG que consulta o banco de dados de metadados PostgreSQL do Amazon Aurora para seu ambiente e publica métricas personalizadas no in. CloudWatch Como usar um DAG para gravar métricas personalizadas no CloudWatch

Métricas do ambiente Apache Airflow v2 em CloudWatch

O Apache Airflow v2 já está configurado para coletar e enviar métricas StatsD para um ambiente Amazon Managed Workflows for Apache Airflow para a Amazon. CloudWatch A lista completa de métricas que o Apache Airflow envia está disponível na página Métricas em Guia de referência do

Apache Airflow. Esta página descreve as métricas do Apache Airflow disponíveis no CloudWatch console e como acessá-las. CloudWatch

Sumário

Termos

Namespace

Um namespace é um contêiner para as CloudWatch métricas de um AWS serviço. Para o Amazon MWAA, o namespace é AmazonMWAA.

CloudWatch métricas

Uma CloudWatch métrica representa um conjunto ordenado por tempo de pontos de dados específicos de CloudWatch.

Métricas do Apache Airflow

As métricas específicas do Apache Airflow.

Dimensão

Uma dimensão é um par de nome/valor que faz parte da identidade de uma métrica.

Unidade

Uma estatística tem uma unidade de medida. Para o Amazon MWAA, as unidades incluem Contagem, Segundos e Milissegundos. Para o Amazon MWAA, as unidades são definidas com base nas unidades nas métricas originais do Airflow.

Dimensões

Esta seção descreve o agrupamento de CloudWatch dimensões para métricas do Apache Airflow em. CloudWatch

Dimensão	Descrição
DAG	Indica um nome específico de DAG do Apache Airflow.
Nome do arquivo DAG	Indica um nome de arquivo específico de DAG do Apache Airflow.
Função	Essa dimensão é usada para melhorar o agrupamento de métricas em CloudWatch.
Trabalho	Indica um trabalho do Apache Airflow executado pelo Agendador. Sempre tem o valor de Trabalho.
Operador	Indica um operador específico do Apache Airflow.
Grupo	Indica um grupo de operadores específico do Apache Airflow.
Tarefa	Indica uma tarefa específica do Apache Airflow.
HostName	Indica o nome do host de um processo específic

Dimensão	Descrição			
	o do Apache Airflow em execução.			

Acessando métricas no CloudWatch console

Esta seção descreve como acessar as métricas de desempenho CloudWatch de um DAG específico.

Para visualizar as métricas de desempenho para uma dimensão

1. Abra a página Métricas no CloudWatch console.
2. Use o seletor de AWS região para selecionar sua região.
3. Escolha o namespace AmazonMWAA.
4. Na guia Todas as métricas, selecione uma dimensão. Por exemplo, DAG, Ambiente.
5. Escolha uma CloudWatch métrica para uma dimensão. Por exemplo, TaskInstanceSucessos ou TaskInstanceDuração. Escolha Representar graficamente todos os resultados da pesquisa.
6. Escolha a guia Métricas representadas graficamente para visualizar estatísticas de desempenho das métricas do Apache Airflow, como DAG, Ambiente, Tarefa.

Métricas do Apache Airflow disponíveis em CloudWatch

Esta seção descreve as métricas e dimensões do Apache Airflow enviadas para. CloudWatch

Contadores do Apache Airflow

As métricas do Apache Airflow nesta seção contêm dados sobre os Contadores do Apache Airflow.

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
SLAMissed	sla_missed	Contagem	Função, Agendador	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
Note Disponível para Apache Airflow v2.4.3 e superior.				
FailedSLACallback **Note** Disponível para Apache Airflow v2.4.3 e superior.	sla_callback_notification_failure	Contagem	Função, Agendador	
Atualizações **Note** Disponível para Apache Airflow v2.6.3 e superior.	dataset.updates	Contagem	Função, Agendador	
Orphaned **Note** Disponível para Apache Airflow v2.6.3 e superior.	dataset.orphaned	Contagem	Função, Agendador	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
FailedCeleryTaskExecution **Note** Disponível para Apache Airflow v2.4.3 e superior.	celery.execute_command.failure	Contagem	Função, Celery	
FilePathQueueUpdateContagem **Note** Disponível para Apache Airflow v2.6.3 e superior.	dag_processing.file_path_queue_update_count	Contagem	Função, Agendador	
CriticalSectionOcupado	scheduler.critical_section_busy	Contagem	Função, Agendador	
DagBagTamanho	dagbag_size	Contagem	Função, Processamento de DAG	
DagCallbackExceções	dag.callback_exceptions	Contagem	DAG, Todos	
Falha no SLA EmailAttempts	sla_email_notification_failure	Contagem	Função, Agendador	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
TaskInstanceConcluído	ti.finish.{dag_id}.{task_id}.{state}	Contagem	DAG, {dag_id} Tarefa, {task_id} Estado, {state}	
JobEnd	{job_name}_end	Contagem	Trabalho, {job_name}	
JobHeartbeatFalha	{job_name}_heartbeat_failure	Contagem	Trabalho, {job_name}	
JobStart	{job_name}_start	Contagem	Trabalho, {job_name}	
ManagerStalls	dag_processing.manager_stalls	Contagem	Função, Processamento de DAG	
OperatorFailures	operator_failures_{operator_name}	Contagem	Operador, {operator_name}	
OperatorSsuccesses	operator_successes_{operator_name}	Contagem	Operador, {operator_name}	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
OtherCallbackContagem Note Disponível no Apache Airflow v2.6.3 e superior.	dag_processing.other_callback_count	Contagem	Função, Agendador	
Processos	dag_processing.processes	Contagem	Função, Processamento de DAG	
SchedulerHeartbeat	scheduler_heartbeat	Contagem	Função, Agendador	
StartedTaskInstâncias	ti.start.{dag_id}.{task_id}	Contagem	DAG, Todos Tarefa, Todos	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
SlaCallbackContagem	dag_processing.sla_callback_count **Note** Disponível para Apache Airflow v2.6.3 e superior.	Contagem	Função, Agendador	
TasksKilledExternamente	scheduler.tasks.killed_externally	Contagem	Função, Agendador	
TaskTimeoutErro	celery.task_timeout_error	Contagem	Função, Celery	
TaskInstanceCreatedUsingOperador	task_instance_created-{operator_name}	Contagem	Operador, {operator_name}	
TaskInstancePreviouslySucceeded	previously_succeeded	Contagem	DAG, Todos Tarefa, Todos	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
TaskInstanceFalhas	ti_failures	Contagem	DAG, Todos Tarefa, Todos	
TaskInstanceSucessos	ti_successes	Contagem	DAG, Todos Tarefa, Todos	
TaskRemovedde DAG	task_removed_from_dag.{dag_id}	Contagem	DAG, {dag_id}	
TaskRestoredPara DAG	task_restored_to_dag.{dag_id}	Contagem	DAG, {dag_id}	
TriggersSucceeded **Note** Disponível para Apache Airflow v2.7.2 e superior.	triggers.succeeded	Contagem	Função, Trigger	
TriggersFailed **Note** Disponível para Apache Airflow v2.7.2 e superior.	triggers.failed	Contagem	Função, Trigger	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
TriggersBlockedMainThread **Note** Disponível para Apache Airflow v2.7.2 e superior.	triggers.blocked_main_thread	Contagem	Função, Trigger	
TriggerHeartbeat **Note** Disponível para Apache Airflow v2.8.1 e superior.	triggerer_heartbeat	Contagem	Função, gatilho	
TaskInstanceCreatedUsingOperador	airflow.task_instance_created_`{operator_name}` **Note** Disponível para Apache Airflow v2.7.2 e superior.	Contagem	Operador, `{operator_name}`	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
ZombiesKilled	zombies_killed	Contagem	DAG, Todos Tarefa, Todos	

Medidores do Apache Airflow

As métricas do Apache Airflow nesta seção contêm dados sobre os Medidores do Apache Airflow.

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
Erro do DAG FileRefresh	dag_file_refresh_error	Contagem	Função, Processamento de DAG	
ImportErrors	dag_processing.import_errors	Contagem	Função, Processamento de DAG	
Exception Failures	smart_sensor_operator.exception_failures	Contagem	Função, Operador de sensor inteligente	
ExecutedTasks	smart_sensor_operator.executed_tasks	Contagem	Função, Operador de sensor inteligente	
InfraFailures	smart_sensor_operator.infra_failures	Contagem	Função, Operador de	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
			sensor inteligente	
LoadedTasks	smart_sensor_operator.loaded_tasks	Contagem	Função, Operador de sensor inteligente	
TotalParseHora	dag_processing.total_parse_time	Segundos	Função, Processamento de DAG	
TriggeredDagCorre **Note** Disponível no Apache Airflow v2.6.3 e superior.	dataset.triggered_dagruns	Contagem	Função, Agendador	
TriggersRunning **Note** Disponível no Apache Airflow v2.7.2 e superior.	triggers.running.*{hostname}*	Contagem	Função, Trigger HostName, *{nome do host}*	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
PoolDeferredSlots **Note** Disponível no Apache Airflow v2.7.2 e superior.	pool.deferred_slots.`{pool_name}`	Contagem	Pool, {pool_name}	
DAG FileProcessing LastRunSecondsAgo	dag_processing.last_run.seconds_ago.{dag_filename}	Segundos	Nome do arquivo DAG, {dag_filename}	
OpenSlots	executor.open_slots	Contagem	Função, Executor	
OrphanedTasksAdotado	scheduler.orphaned_tasks.adopted	Contagem	Função, Agendador	
OrphanedTasksApurado	scheduler.orphaned_tasks.cleared	Contagem	Função, Agendador	
PokedExceptions	smart_sensor_operator.poked_exception	Contagem	Função, Operador de sensor inteligente	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
PokedSuccess	smart_sensor_operator.poked_success	Contagem	Função, Operador de sensor inteligente	
PokedTasks	smart_sensor_operator.poked_tasks	Contagem	Função, Operador de sensor inteligente	
PoolFailures	pool.open_slots.{pool_name}	Contagem	Pool, {pool_name}	
PoolStarvingTarefas	pool.starving_tasks.{pool_name}	Contagem	Pool, {pool_name}	
PoolOpenSlots	pool.open_slots.{pool_name}	Contagem	Pool, {pool_name}	
PoolQueuedSlots	pool.queued_slots.{pool_name}	Contagem	Pool, {pool_name}	
PoolRunningSlots	pool.running_slots.{pool_name}	Contagem	Pool, {pool_name}	
Processor Timeouts	dag_processing.processor_timeouts	Contagem	Função, Processamento de DAG	
QueuedTasks	executor.queued_tasks	Contagem	Função, Executor	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
RunningTasks	executor.running_tasks	Contagem	Função, Executor	
TasksExecutable	scheduler.tasks.executable	Contagem	Função, Agendador	
TasksPending **Note** Não se aplica ao Apache Airflow v2.2 e superior.	scheduler.tasks.pending	Contagem	Função, Agendador	
TasksRunning	scheduler.tasks.running	Contagem	Função, Agendador	
TasksStarving	scheduler.tasks.starving	Contagem	Função, Agendador	
TasksWithoutDagRun	scheduler.tasks.without_dagrun	Contagem	Função, Agendador	

Temporizadores do Apache Airflow

As métricas do Apache Airflow nesta seção contêm dados sobre os temporizadores do Apache Airflow.

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
CollectDBDags	collect_db_dags	Milissegundos	Função, Processamento de DAG	
CriticalSectionDuração	scheduler.critical_section_duration	Milissegundos	Função, Agendador	
CriticalSectionQueryDuration **Note** Disponível para Apache Airflow v2.5.1 e superior.	scheduler.critical_section_query_duration	Milissegundos	Função, Agendador	
DAG DependencyCheck	dagrun.dependency-check.{dag_id}	Milissegundos	DAG, {dag_id}	
DAG DurationFailed	dagrun.duration.failed.{dag_id}	Milissegundos	DAG, {dag_id}	
DAG DurationSuccess	dagrun.duration.success.{dag_id}	Milissegundos	DAG, {dag_id}	

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão
DAG FileProcessing LastDuration	dag_processing.last_duration.{dag_filename}	Segundos	Nome do arquivo DAG, {dag_filename}
DAG ScheduleDelay	dagrun.schedule_delay.{dag_id}	Milissegundos	DAG, {dag_id}
FirstTaskSchedulingDelay	dagrun.{dag_id}.first_task_scheduling_delay	Milissegundos	DAG, {dag_id}
SchedulerLoopDuração **Note** Disponível para Apache Airflow v2.5.1 e superior.	scheduler.scheduler_loop_duration	Milissegundos	Função, Agendador
TaskInstanceDuração	dag.{dag_id}.{task_id}.duration	Milissegundos	DAG, {dag_id} Tarefa, {task_id}

CloudWatch métrica	Métrica do Apache Airflow	Unidade	Dimensão	
TaskInstanceQueuedDuration	dag.{dag_id}.{t .queued_duration **Note** Disponível para Apache Airflow v2.7.2 e superior.	Milissegundos	DAG, {dag_id} Tarefa, {task_id}	
TaskInstanceScheduledDuration **Note** Disponível para Apache Airflow v2.7.2 e superior.	dag.{dag_id}.{t .scheduled_duration	Milissegundos	DAG, {dag_id} Tarefa, {task_id}	

Como escolher quais métricas são relatadas

Você pode escolher quais métricas do Apache Airflow são emitidas ou bloqueadas pelo Apache Airflow usando as seguintes opções de configuração do Amazon MWAA: CloudWatch

- **`metrics.metrics_allow_list`**— Uma lista de prefixos separados por vírgula que você pode usar para selecionar quais métricas são emitidas pelo seu CloudWatch ambiente. Use essa opção

se quiser que o Apache Airflow não envie todas as métricas disponíveis e, em vez disso, selecione um subconjunto de elementos. Por exemplo, `scheduler,executor,dagrun`.

- **`metrics.metrics_block_list`**: uma lista de prefixos separados por vírgula para filtrar as métricas que começam com os elementos da lista. Por exemplo, `scheduler,executor,dagrun`.

Se você configurar `metrics.metrics_allow_list` e`metrics.metrics_block_list`, o Apache Airflow ignorará `metrics.metrics_block_list`. Se você configura `metrics.metrics_block_list`, mas não `metrics.metrics_allow_list`, o Apache Airflow filtra os elementos que você especifica em `metrics.metrics_block_list`.

> **Note**
>
> As opções de `metrics.metrics_block_list` configuração `metrics.metrics_allow_list` e se aplicam somente ao Apache Airflow v2.6.3 e superior. Para a versão anterior do Apache Airflow, use `metrics.statsd_allow_list` e `metrics.statsd_block_list` em vez disso.

Próximas etapas

- Explore a operação da API Amazon MWAA usada para publicar métricas de integridade do ambiente em. PublishMetrics

Métricas de contêiner, fila e banco de dados para Amazon MWAA

Além das métricas do Apache Airflow, você pode monitorar os componentes subjacentes dos seus ambientes Amazon Managed Workflows for Apache Airflow usando CloudWatch, que coleta dados brutos e processa dados em métricas legíveis, quase em tempo real. Com essas métricas de ambiente, você terá maior visibilidade dos principais indicadores de desempenho, ajudando a dimensionar adequadamente seus ambientes e depurar problemas com seus fluxos de trabalho. Essas métricas se aplicam a todas as versões compatíveis do Apache Airflow no Amazon MWAA.

O Amazon MWAA fornecerá utilização de CPU e memória para cada contêiner do Amazon Elastic Container Service (Amazon ECS) e instância do Amazon Aurora PostgreSQL, e métricas do Amazon Simple Queue Service (Amazon SQS) para o número de mensagens e a idade da mensagem mais

antiga, métricas do Amazon Relational Database Service (Amazon RDS) para conexões de banco de dados, profundidade da fila de disco, operações de gravação, latência e throughput, além de métricas do Amazon RDS Proxy. Essas métricas também incluem o número de operadores básicos, operadores adicionais, agendadores e servidores web.

Essas estatísticas são mantidas por 15 meses, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor de por que um cronograma está falhando e solucionar problemas subjacentes. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

Tópicos

Termos

Namespace

Um namespace é um contêiner para as CloudWatch métricas de um AWS serviço. Para o Amazon MWAA, o namespace é `AWS/MWAA`.

CloudWatch métricas

Uma CloudWatch métrica representa um conjunto ordenado por tempo de pontos de dados específicos de CloudWatch.

Dimensão

Uma dimensão é um par de nome/valor que faz parte da identidade de uma métrica.

Unidade

Uma estatística tem uma unidade de medida. Para o Amazon MWAA, as unidades incluem Contagem.

Dimensões

Esta seção descreve o agrupamento de CloudWatch dimensões para as métricas do Amazon MWAA em. CloudWatch

Dimensão	Descrição
Cluster	Métricas para o mínimo de três contêineres do Amazon ECS que um ambiente Amazon MAA usa para executar componentes do Apache Airflow: agendador, operador e servidor web.
Fila	Métricas para as filas do Amazon SQS que separam o agendador dos operadores. Quando os operadores leem as mensagens , elas são consideradas em trânsito e não estão disponíveis para outros operadores. As mensagens ficam disponíveis para que outros operadores as leiam se não forem excluídas antes do tempo limite de visibilidade de 12 horas.
Banco de dados	Métricas dos clusters Aurora usados pelo Amazon MWAA. Ele inclui métricas para a instância primária do banco de dados e uma réplica de leitura para dar suporte às operações de leitura. O Amazon MWAA publica métricas de banco de dados para as instâncias READER e WRITER.

Acessando métricas no CloudWatch console

Esta seção descreve como acessar suas métricas do Amazon MWAA em. CloudWatch

Para visualizar as métricas de desempenho para uma dimensão

1. Abra a página Métricas no CloudWatch console.

2. Use o seletor de AWS região para selecionar sua região.
3. Escolha o namespace AWS/MWAA.
4. Na guia Todas as métricas, escolha uma dimensão. Por exemplo, Cluster.
5. Escolha uma CloudWatch métrica para uma dimensão. Por exemplo, NumSchedulersou CPUUtilization. Em seguida, escolha Representar graficamente todos os resultados da pesquisa.
6. Escolha a guia Métricas representadas graficamente para visualizar as métricas de desempenho.

Lista de métricas

As tabelas a seguir listam as métricas de cluster, fila e serviço de banco de dados para o Amazon MWAA. Para visualizar descrições de métricas emitidas diretamente do Amazon ECS, Amazon SQS ou Amazon RDS, escolha o respectivo link da documentação.

Tópicos

Métricas de cluster

As métricas a seguir se aplicam a cada agendador, operador base, operador adicional e servidor web. Para obter mais informações e descrições de cada métrica de cluster, consulte Métricas e dimensões disponíveis no Guia do desenvolvedor do Amazon ECS.

Namespace	Métrica	Unidade
`AWS/MWAA`	`CPUUtilization`	Percentual
`AWS/MWAA`	`MemoryUtilization`	Percentual

Avaliando o número de contêineres adicionais de trabalhadores e servidores web

Você pode usar as métricas de componentes fornecidas na dimensão Cluster, conforme descrito no procedimento a seguir, para avaliar quantos trabalhadores adicionais, ou servidores web, um ambiente está usando em um determinado momento. Você pode fazer isso representando graficamente a utilização da CPU ou a MemoryUtilizationmétrica e definindo o tipo de estatística como Contagem de amostras. O valor resultante é o número total de tarefas `RUNNING` do componente `AdditionalWorker`. Compreender o número de instâncias de trabalho adicionais utilizadas pelo seu ambiente pode ajudá-lo a avaliar como seu ambiente se expande e permitir que você otimize o número de trabalhadores adicionais.

Workers

Para avaliar o número de trabalhadores adicionais usando o AWS Management Console

1. Escolha o namespace AWS/MWAA.
2. Na guia Todas as métricas, escolha a dimensão Cluster.
3. Na dimensão Cluster, para o AdditionalWorker, escolha a utilização da CPU ou a métrica. MemoryUtilization
4. Na guia Métricas em gráfico, configure o Período para 1 minuto e Estatística para Contagem de amostras.

Web servers

Para avaliar o número de servidores web adicionais usando o AWS Management Console

1. Escolha o namespace AWS/MWAA.
2. Na guia Todas as métricas, escolha a dimensão Cluster.
3. Na dimensão Cluster, para o AdditionalWebservers, escolha a utilização da CPU ou a métrica. MemoryUtilization
4. Na guia Métricas em gráfico, configure o Período para 1 minuto e Estatística para Contagem de amostras.

Para obter mais informações, consulte Contagem de tarefas do serviço `RUNNING` no Guia do desenvolvedor do Amazon Elastic Container Service.

Métricas de banco de dados

As métricas a seguir se aplicam a cada instância de banco de dados associada ao ambiente Amazon MWAA.

Namespace	Métrica	Unidade
AWS/MWAA	CPUUtilization	Percentual
AWS/MWAA	DatabaseConnections	Contagem
AWS/MWAA	DiskQueueDepth	Contagem
AWS/MWAA	FreeableMemory	Bytes
AWS/MWAA	VolumeWriteIOPS	Contagem a cada cinco minutos
AWS/MWAA	WriteIOPS	Contagem por segundo
AWS/MWAA	WriteLatency	Segundos
AWS/MWAA	WriteThroughput	Bytes por segundo

Métricas de fila

Para obter mais informações sobre unidades e descrições das seguintes métricas de fila, consulte Métricas disponíveis CloudWatch para o Amazon SQS no Guia do desenvolvedor do Amazon Simple Queue Service.

Namespace	Métrica	Unidade
AWS/MWAA	ApproximateAgeOfOldestTask	Segundos
AWS/MWAA	RunningTasks	Contagem

Namespace	Métrica	Unidade
AWS/MWAA	QueuedTasks	Contagem

Métricas do Application Load Balancer

As métricas do Application Load Balancer se aplicam aos servidores web em execução em seu ambiente. O Amazon MWAA usa essas métricas para escalar seus servidores web com base na quantidade de tráfego. Para obter mais informações sobre unidades e descrições das seguintes métricas do balanceador de carga, consulte CloudWatch as métricas do seu Application Load Balancer no Guia do usuário do Application Load Balancers.

Namespace	Métrica	Unidade
AWS/MWAA	ActiveConnectionCount	Contagem

Segurança no Amazon Managed Workflows for Apache Airflow

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você (o cliente). O modelo de responsabilidade compartilhada descreve isso como a segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam ao Amazon MWAA, consulte AWS Serviços no escopo do programa de conformidade AWS .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon Managed Workflows for Apache Airflow. Ela mostra como configurar o Amazon MWAA para atender aos objetivos de segurança e conformidade. Você também aprende a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do Amazon MWAA.

Nesta seção:

Proteção de dados no Amazon Managed Workflows for Apache Airflow

O modelo de responsabilidade AWS compartilhada O modelo se aplica à proteção de dados no Amazon Managed Workflows for Apache Airflow. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que você usa. Para ter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para ter mais informações sobre a proteção de dados na Europa, consulte a AWS postagem do blog Shared Responsibility Model and GDPR no AWS Blog de segurança da.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas de usuário individuais com AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Recomendamos usar o TLS 1.2 ou posterior.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções AWS de criptografia, juntamente com todos os controles de segurança padrão nos AWS serviços.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o Amazon MWAA ou outros AWS serviços usando o console, a API ou AWS os AWS CLI SDKs. Quaisquer dados inseridos em marcações ou campos de formato livre usados para nomes podem ser usados para logs de cobrança ou diagnóstico. Se fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia no Amazon MWAA

Os tópicos a seguir descrevem como o Amazon MWAA protege seus dados em repouso e em trânsito. Use essas informações para saber como o Amazon MWAA se integra AWS KMS para criptografar dados em repouso e como os dados são criptografados usando o protocolo Transport Layer Security (TLS) em trânsito.

Tópicos

Criptografia inativa

No Amazon MWAA, dados em repouso são dados que o serviço salva em mídia persistente.

Você pode usar uma chave pertencente àAWS para criptografia de dados em repouso ou, opcionalmente, fornecer uma chave gerenciada pelo cliente para criptografia adicional ao criar um ambiente. Se você optar por usar uma chave KMS gerenciada pelo cliente, ela deverá estar na mesma conta dos outros AWS recursos e serviços que você está usando com seu ambiente.

Para usar uma chave KMS gerenciada pelo cliente, você deve anexar a declaração de política necessária para CloudWatch acessar sua política de chaves. Quando você usa uma chave KMS gerenciada pelo cliente para seu ambiente, o Amazon MWAA atribui quatro concessões em seu nome. Para obter mais informações sobre as concessões que o Amazon MWAA atribui a uma chave KMS gerenciada pelo cliente, consulte Chaves gerenciadas pelo cliente para criptografia de dados.

Se você não especificar uma chave KMS gerenciada pelo cliente, por padrão, o Amazon MWAA usa uma chave KMS AWS própria para criptografar e descriptografar seus dados. Recomendamos usar uma chave AWS KMS própria para gerenciar a criptografia de dados no Amazon MWAA.

> **Note**
>
> Você paga pelo armazenamento e uso de chaves KMS AWS próprias ou gerenciadas pelo cliente no Amazon MWAA. Para obter mais informações, consulte Preços do AWS KMS.

Artefatos de criptografia

Você especifica os artefatos de criptografia usados para criptografia em repouso especificando uma chave pertencente àAWS ou uma chave gerenciada pelo cliente ao criar seu ambiente do Amazon MWAA. O Amazon MWAA adiciona as concessões necessárias à sua chave especificada.

Amazon S3: os dados do Amazon S3 são criptografados no nível do objeto usando criptografia do lado do servidor (SSE). A criptografia e a descriptografia do Amazon S3 ocorrem no bucket do Amazon S3 onde seu código DAG e os arquivos de suporte são armazenados. Os objetos são criptografados quando são carregados para o Amazon S3 e descriptografados quando são baixados para seu ambiente do Amazon MWAA. Por padrão, se você estiver usando uma chave KMS gerenciada pelo cliente, o Amazon MWAA a usará para ler e descriptografar os dados no seu bucket do Amazon S3.

CloudWatch Registros — Se você estiver usando uma chave KMS própria, os registros do Apache Airflow enviados para o Logs serão criptografados usando a criptografia do lado do servidor (SSE) com CloudWatch a chave KMS de AWS propriedade da CloudWatch Logs. AWS Se você estiver usando uma chave KMS gerenciada pelo cliente, deverá adicionar uma política de chaves à sua chave KMS para permitir que CloudWatch os Logs usem sua chave.

Amazon SQS: o Amazon MWAA cria uma fila do Amazon SQS para seu ambiente. O Amazon MWAA manipula a criptografia de dados passados de e para a fila usando criptografia do lado do servidor (SSE) com uma chave KMS própria ou uma chave AWS KMS gerenciada pelo cliente que você especificar. Você deve adicionar permissões do Amazon SQS à sua função de execução, independentemente de estar usando uma chave KMS AWS própria ou gerenciada pelo cliente.

Aurora PostgreSQL: o Amazon MWAA cria um cluster PostgreSQL para seu ambiente. O Aurora PostgreSQL criptografa o conteúdo com uma chave KMS AWS própria ou gerenciada pelo cliente usando criptografia do lado do servidor (SSE). Se você estiver usando uma chave KMS gerenciada pelo cliente, o Amazon RDS adiciona pelo menos duas concessões à chave: uma para o cluster e outra para a instância do banco de dados. O Amazon RDS pode criar concessões adicionais se você optar por usar sua chave KMS gerenciada pelo cliente em vários ambientes. Para obter mais informações, consulte Proteção de dados no Amazon RDS.

Criptografia em trânsito

Os dados em trânsito são chamados de dados que podem ser interceptados enquanto viajam pela rede.

O Transport Layer Security (TLS) criptografa os objetos do Amazon MWAA em trânsito entre os componentes do Apache Airflow do seu ambiente e outros serviços AWS que se integram ao Amazon MWAA, como o Amazon S3. Para obter mais informações sobre a criptografia da Amazon S3, consulte Como proteger dados usando criptografia.

Como usar chaves gerenciadas pelo cliente para criptografia

Opcionalmente, você pode fornecer uma chave gerenciada pelo cliente para criptografia de dados em seu ambiente. Você deve criar uma chave KMS gerenciada pelo cliente na mesma região da sua instância do ambiente do Amazon MWAA e do seu bucket do Amazon S3 onde você armazena recursos para seus fluxos de trabalho. Se a chave KMS gerenciada pelo cliente que você especificou estiver em uma conta diferente da que foi usada para configurar um ambiente, será necessário especificar a chave usando o respectivo ARN para acesso entre contas. Para obter mais informações sobre como criar chaves, consulte Como criar chaves no Guia do desenvolvedor do AWS Key Management Service .

O que é compatível

AWS KMS recurso	Compatível
Um ID de chave ou ARN AWS KMS.	Sim
Um alias de chave AWS KMS.	Não
Uma chave de várias regiões AWS KMS.	Não

Como usar concessões para criptografia

Este tópico descreve as concessões que o Amazon MWAA atribui a uma chave do KMS gerenciada pelo cliente em seu nome para criptografar e descriptografar os dados.

Como funciona

Há dois mecanismos de controle de acesso baseados em recursos suportados pela chave KMS gerenciada AWS KMS pelo cliente: uma política de chaves e uma concessão.

A política de chave é usada quando a permissão é principalmente estática e usada no modo de serviço síncrono. A concessão é usada quando são necessárias permissões mais dinâmicas e granulares, como quando um serviço precisa definir permissões de acesso diferentes para si mesmo ou para outras contas.

O Amazon MWAA usa e anexa quatro políticas de concessão à sua chave KMS gerenciada pelo cliente. Isso se deve às permissões granulares necessárias para que um ambiente criptografe dados ociosos de CloudWatch Logs, fila do Amazon SQS, banco de dados de banco de dados Aurora PostgreSQL, segredos do Secrets Manager, bucket do Amazon S3 e tabelas do DynamoDB.

Ao criar um ambiente do Amazon MWAA e especificar uma chave KMS gerenciada pelo cliente, o Amazon MWAA anexa as políticas de concessão à sua chave KMS gerenciada pelo cliente. Essas políticas permitem que o Amazon MWAA em `airflow.region}.amazonaws.com` use sua chave KMS gerenciada pelo cliente para criptografar recursos em seu nome que são de propriedade do Amazon MWAA.

O Amazon MWAA cria e anexa concessões adicionais a uma chave KMS especificada em seu nome. Isso inclui políticas para retirar uma concessão se você excluir seu ambiente, usar sua chave KMS gerenciada pelo cliente para criptografia do lado do cliente (CSE) e para a função de AWS Fargate execução que precisa acessar segredos protegidos por sua chave gerenciada pelo cliente no Secrets Manager.

Políticas de concessões

O Amazon MWAA adiciona as seguintes concessões de políticas baseadas em recursos em seu nome a uma chave KMS gerenciada pelo cliente. Essas políticas permitem que o beneficiário e a entidade principal (Amazon MWAA) realizem ações definidas na política.

Concessão 1: usada para criar recursos do plano de dados

```
{
            "Name": "mwaa-grant-for-env-mgmt-role-environment name",
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:CreateGrant",
              "kms:DescribeKey",
```

```
            "kms:RetireGrant"
          ]
        }
```

Concessão 2: usada para o acesso **`ControllerLambdaExecutionRole`**

```
{
          "Name": "mwaa-grant-for-lambda-exec-environment name",
          "GranteePrincipal": "airflow.region.amazonaws.com",
          "RetiringPrincipal": "airflow.region.amazonaws.com",
          "Operations": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey",
            "kms:RetireGrant"
          ]
        }
```

Concessão 3: usada para o acesso **`CfnManagementLambdaExecutionRole`**

```
{
            "Name": " mwaa-grant-for-cfn-mgmt-environment name",
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:DescribeKey"
            ]
          }
```

Concessão 4: usada para o perfil de execução do Fargate para acessar segredos de back-end

```
{
              "Name": "mwaa-fargate-access-for-environment name",
```

```
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:DescribeKey",
              "kms:RetireGrant"
            ]
          }
```

Como anexar políticas de chave a uma chave gerenciada pelo cliente

Se você optar por usar sua própria chave KMS gerenciada pelo cliente com o Amazon MWAA, deverá anexar a seguinte política à chave para permitir que o Amazon MWAA a use para criptografar seus dados.

Se a chave KMS gerenciada pelo cliente que você usou para seu ambiente Amazon MWAA ainda não estiver configurada para funcionar CloudWatch, você deverá atualizar a política de chaves para permitir registros criptografados. CloudWatch Para obter mais informações, consulte Criptografar dados de log no CloudWatch uso do AWS Key Management Service serviço.

O exemplo a seguir representa uma política fundamental para o CloudWatch Logs. Substituir os valores de amostra fornecidos para a região.

```
{
          "Effect": "Allow",
          "Principal": {
          "Service": "logs.us-west-2.amazonaws.com"
        },
        "Action": [
          "kms:Encrypt*",
          "kms:Decrypt*",
          "kms:ReEncrypt*",
          "kms:GenerateDataKey*",
          "kms:Describe*"
        ],
        "Resource": "*",
        "Condition": {
          "ArnLike": {
```

```
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-west-2:*:*"
            }
        }
    }
```

AWS Identity and Access Management

AWS Identity and Access Management (IAM) é um AWS serviço que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar os recursos do Amazon Managed Workflows for Apache Airflow. O IAM é um AWS serviço que você pode usar sem custo adicional.

Este tópico fornece uma visão geral básica de como o Amazon MWAA usa AWS Identity and Access Management (IAM). Para saber mais sobre como gerenciar o acesso ao Amazon MWAA, consulte Gerenciamento de acesso a um ambiente do Amazon MWAA.

Conteúdo

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon MWAA.

Usuário do serviço: se você usar o serviço do Amazon MWAA para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões necessárias. À medida que mais atributos do Amazon MWAA forem usados para realizar o trabalho, talvez sejam necessárias permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um atributo no Amazon MWAA, consulte Solução de problemas de identidade e acesso do Amazon Managed Workflows for Apache Airflow.

Administrador do serviço: se você for o responsável pelos recursos do Amazon MWAA em sua empresa, provavelmente terá acesso total ao Amazon MWAA. Cabe a você determinar quais funcionalidades e atributos do Amazon MWAA os usuários do serviço deverão acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon MWAA, consulte Como o Amazon MWAA funciona com o IAM.

Administrador do IAM: se você for um administrador do IAM, talvez precise saber detalhes sobre como escrever políticas para gerenciar o acesso ao Amazon MWAA. Para visualizar exemplos de políticas baseadas em identidade do Amazon MWAA que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon MWAA.

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação

multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação Multifator no AWS IAM Identity Center Guia do Usuário. Usar a autenticação multifator (MFA) na AWS no Guia do Usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele pode executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do usuário do IAM.

Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos depender de credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne Chaves de Acesso Regularmente para Casos de Uso que exijam Credenciais de Longo Prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um nome de grupo IAMAdmins e atribuir a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a um aplicativo, mas uma função pode ser assumida por qualquer pessoa que precisar dela. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando Criar um Usuário do IAM (Ao Invés de uma Função) no Guia do Usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você

pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte Usando Funções do IAM no Guia do Usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criando um Perfil para um Provedor de Identidades Terceirizado no Guia do Usuário do IAM. Se você usa o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no AWS IAM Identity Center Manual do Usuário.
- Permissões de usuários temporárias do IAM: um usuário ou perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: você pode usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) acesse recursos na sua conta de uma conta diferente. As funções são a forma primária de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para aprender a diferença entre funções e políticas baseadas em recurso para acesso entre contas, consulte Como as Funções do IAM Diferem das Políticas Baseadas em Recurso no Guia do Usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões de chamada da entidade principal, uma função de serviço ou uma função vinculada ao serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros

Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

- Função de Serviço: uma função de serviço é uma função do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criando um Perfil para Delegar Permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para funções vinculadas a serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar uma função do IAM para conceder permissões a aplicativos em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para aprender se deseja usar perfis do IAM, consulte Quando Criar uma Função do IAM (em Vez de um Usuário) no Guia do Usuário do IAM.

Gerenciamento do acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão Geral das Políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM às funções e os usuários podem assumir as funções.

As políticas do IAM definem permissões para uma ação, independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em quais condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade também podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são incorporadas diretamente a um único usuário, grupo ou função. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como selecionar entre uma política gerenciada ou uma política em linha, consulte Selecionar entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recurso

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de função do IAM e as políticas do bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em atributos são políticas em linha que estão localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou funções da conta) têm permissão para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Saiba mais sobre ACLs em Configurações da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade e dos seus limites de permissões. As políticas baseadas em atributo que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de Permissões para Entidades do IAM no Guia do Usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte Como os SCPs Funcionam no AWS Organizations Manual do Usuário do.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para uma função ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas

substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
```

```
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Solução de problemas de identidade e acesso do Amazon Managed Workflows for Apache Airflow

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você possa encontrar ao trabalhar com a Amazon MWAA e o IAM.

Não tenho autorização para executar uma ação no Amazon MWAA

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

Não estou autorizado a realizar iam: PassRole

Caso receba uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, as políticas deverão ser atualizadas para permitir a transmissão de um perfil ao Amazon MWAA.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando um usuário do IAM chamado `marymajor` tenta usar o console para executar uma ação no Amazon MWAA. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Amazon MWAA

Você pode criar uma função que os usuários de outras contas ou pessoas fora da sua organização possam usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon MWAA é compatível com esses atributos, consulte Como o Amazon MWAA funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Saiba como conceder acesso por meio da federação de identidades consultando Concedendo Acesso a Usuários Autenticados Externamente (Federação de Identidades) no Guia do Usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Como o Amazon MWAA funciona com o IAM

O Amazon MWAA usa políticas baseadas em identidade do IAM para conceder permissões às ações e recursos do Amazon MWAA. Para ver exemplos recomendados de políticas do IAM personalizadas que você pode usar para controlar o acesso aos seus recursos do Amazon MWAA, consulte the section called “Como acessar um ambiente do Amazon MWAA”.

Para obter uma visão de alto nível de como o Amazon MWAA e outros AWS serviços funcionam com o IAM, consulte AWS Serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade do Amazon MWAA

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. O Amazon MWAA é compatível com ações, chaves de condição e recursos específicos.

As etapas a seguir mostram como você pode criar uma nova política JSON usando o console do IAM. Essa política fornece acesso somente leitura aos recursos do seu Amazon MWAA.

Para usar o editor de políticas JSON para criar uma política

1. Faça login AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Policies (Políticas).

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
3. Na parte superior da página, escolha Criar política.
4. Na seção Editor de políticas, escolha a opção JSON.
5. Insira o seguinte documento de política JSON:

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:ListEnvironments",
                "airflow:GetEnvironment",
                "airflow:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

6. Escolha Próximo.

> **Note**
>
> É possível alternar entre as opções de editor Visual e JSON a qualquer momento. Porém, se você fizer alterações ou escolher Próximo no editor Visual, o IAM poderá reestruturar a política a fim de otimizá-la para o editor visual. Para obter mais informações, consulte Restruturação de política no Guia do usuário do IAM.

7. Na página Revisar e criar, insira um Nome de política e uma Descrição (opcional) para a política que você está criando. Revise Permissões definidas nessa política para ver as permissões que são concedidas pela política.
8. Escolha Criar política para salvar sua nova política.

Para conhecer todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Há também algumas operações que exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

As instruções de política devem incluir um elemento `Action` ou um elemento `NotAction`. O elemento `Action` lista as ações permitidas pela política. O elemento `NotAction` lista as ações que não são permitidas.

As ações definidas para o Amazon MWAA refletem tarefas que você pode realizar usando o Amazon MWAA. As ações políticas em Detective têm o seguinte prefixo: `airflow:`.

Você pode usar curingas (*) para especificar várias ações. Em vez de listar essas ações separadamente, você pode conceder acesso a todas as ações que terminam, por exemplo, com a palavra `environment`.

Para ver uma lista das ações do Amazon MWAA, consulte Ações definidas pelo Amazon Managed Workflows for Apache Airflow no Guia do usuário IAM.

Exemplos de políticas baseadas em identidade do Amazon MWAA

Para visualizar as políticas do Amazon MWAA, consulte Gerenciamento de acesso a um ambiente do Amazon MWAA.

Por padrão, os usuários e os perfis do IAM não têm permissão para criar ou modificar recursos do Amazon ECS. Eles também não podem realizar tarefas usando a AWS API AWS Management Console AWS CLI, ou.

Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

> **Important**
>
> Recomendamos usar perfis do IAM e credenciais temporárias para fornecer acesso aos seus recursos do Amazon MWAA. Como evitar anexar políticas de permissão diretamente aos usuários do IAM.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Tópicos

- Melhores práticas de política
- Como usar o console do Amazon SNS
- Permitir que os usuários visualizem suas próprias permissões

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon MWAA em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas Gerenciadas pela AWS ou AWS Políticas Gerenciadas para Funções de Trabalho no Guia do Usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e Permissões no IAM no Guia do Usuário do IAM.
- Utilize condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode gravar uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Condição de Elementos de Política JSON do IAM no Guia do Usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM para garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam o idioma de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e ações recomendadas para ajudar você a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de Política do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configurando Acesso à API Protegido por MFA no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas Recomendadas de Segurança no IAM no Guia do Usuário do IAM.

Como usar o console do Amazon SNS

Para usar o console do Amazon MWAA, o usuário ou o perfil devem ter acesso às ações relevantes que correspondam às ações correspondentes na API.

Para visualizar as políticas do Amazon MWAA, consulte Gerenciamento de acesso a um ambiente do Amazon MWAA.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Validação de conformidade para o Amazon Managed Workflows for Apache Airflow

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

  > **Note**
  >
  > Nem todos Serviços da AWS são elegíveis para a HIPAA. Para obter mais informações, consulte Referência dos Serviços Qualificados pela HIPAA.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o

Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os atributos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços com suporte e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Managed Workflows for Apache Airflow

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, alta throughput e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Segurança da infraestrutura no Amazon MWAA

Como um serviço gerenciado, o Amazon Managed Workflows para Apache Airflow é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como

AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o Amazon MWAA pela rede. Os clientes devem ser compatíveis com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com Perfect Forward Secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, suporta esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Análise de configuração e vulnerabilidade no Amazon MWAA

A configuração e os controles de TI são uma responsabilidade compartilhada entre você AWS e você, nosso cliente.

O Amazon Managed Workflows for Apache Airflow periodicamente corrige e atualiza o Apache Airflow em seus ambientes. Você deve garantir que as políticas de acesso apropriadas sejam usadas para suas VPCs.

Para obter mais detalhes, consulte os seguintes recursos da :

- Validação de conformidade para o Amazon Managed Workflows for Apache Airflow
- Modelo de responsabilidade compartilhada
- Amazon Web Services: visão geral dos processos de segurança
- Segurança da infraestrutura no Amazon MWAA
- Práticas recomendadas de segurança para o Amazon MWAA

Práticas recomendadas de segurança para o Amazon MWAA

O Amazon MWAA fornece uma série de atributos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas melhores práticas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

- Use políticas de permissão com permissão mínima. Conceda permissões somente aos recursos ou ações de que os usuários precisam para realizar tarefas.
- Use AWS CloudTrail para monitorar a atividade do usuário em sua conta.
- Certifique-se de que a política de bucket do Amazon S3 e as ACLs de objetos concedam permissões aos usuários do ambiente associado do Amazon MWAA para colocar objetos no bucket. Isso garante que os usuários com permissões para adicionar fluxos de trabalho ao bucket também tenham permissões para executar os fluxos de trabalho no Airflow.
- Use os buckets do Amazon S3 associados aos ambientes do Amazon MWAA. Seu bucket do Amazon S3 pode ter qualquer nome. Não armazene outros objetos no bucket nem use o bucket com outro serviço.

Práticas recomendadas de segurança no Apache Airflow

O Apache Airflow não é multilocatário. Embora existam medidas de controle de acesso para limitar alguns atributos a usuários específicos, implementadas pelo Amazon MWAA, os criadores do DAG têm a capacidade de gravar DAGs que podem alterar os privilégios de usuário do Apache Airflow e interagir com o banco de dados subjacente.

Recomendamos as seguintes etapas ao trabalhar com o Apache Airflow no Amazon MWAA para garantir que o banco de metadados e os DAGs do seu ambiente estejam seguros.

- Use ambientes separados para equipes separadas com acesso de gravação do DAG ou a capacidade de adicionar arquivos à sua pasta `/dags` do Amazon S3, supondo que qualquer coisa acessível pelo Perfil de execução do Amazon MWAA ou pelas conexões do Apache Airflow também esteja acessível aos usuários que possam gravar no ambiente.
- Não forneça acesso direto à pasta DAGs do Amazon S3. Em vez disso, use ferramentas de CI/CD para gravar DAGs no Amazon S3, com uma etapa de validação garantindo que o código do DAG atenda às diretrizes de segurança da sua equipe.

- Impeça o acesso do usuário ao bucket do Amazon S3 do seu ambiente. Em vez disso, use uma fábrica de DAG que gere DAGs com base em um arquivo YAML, JSON ou outro arquivo de definição armazenado em um local separado do bucket Amazon S3 do Amazon MWAA, onde você armazena DAGs.
- Segredos armazenados no Secrets Manager. Embora isso não impeça que os usuários que podem gravar DAGs leiam segredos, isso impedirá que eles modifiquem os segredos que seu ambiente usa.

Como detectar alterações nos privilégios de usuário do Apache Airflow

Você pode usar o CloudWatch Logs Insights para detectar ocorrências de DAGs alterando os privilégios de usuário do Apache Airflow. Para fazer isso, você pode usar uma regra EventBridge programada, uma função Lambda e o CloudWatch Logs Insights para enviar notificações às CloudWatch métricas sempre que um de seus DAGs alterar os privilégios de usuário do Apache Airflow.

Pré-requisitos

Para concluir as etapas a seguir, é necessário:

- Um ambiente do Amazon MWAA com todos os tipos de log do Apache Airflow habilitados no nível do log `INFO`. Para ter mais informações, consulte the section called “Como visualizar logs do Airflow”.

Para configurar notificações para alterações nos privilégios de usuário do Apache Airflow

1. Crie uma função Lambda que execute a seguinte string de consulta do CloudWatch Logs Insights nos cinco grupos de log do ambiente Amazon MWAA (`DAGProcessing`,, `SchedulerTask`, `WebServer` e). `Worker`

   ```
   fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
    by @log
   ```

2. Crie uma EventBridge regra que seja executada em um cronograma, com a função Lambda que você criou na etapa anterior como destino da regra. Configure sua programação usando uma expressão cron ou rate para executar a intervalos regulares.

Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow.

Esta página descreve as versões do Apache Airflow compatíveis com o Amazon Managed Workflows for Apache Airflow e as estratégias que recomendamos para atualizar para a versão mais recente.

Tópicos

Sobre as versões do Amazon MWAA

O Amazon MWAA cria imagens de contêiner que empacotam as versões do Apache Airflow com outros binários e bibliotecas Python comuns. A imagem usa a instalação básica do Apache Airflow para a versão especificada. Ao criar um ambiente, você especifica uma versão de imagem a ser usada. Depois que um ambiente é criado, ele continua usando a versão de imagem especificada até que você o atualize para uma versão posterior.

Versão mais recente

O Amazon MWAA oferece suporte a mais de uma versão do Apache Airflow. Caso não seja especificada uma versão de imagem ao criar um ambiente, o Amazon MWAA cria um ambiente usando a versão mais recente compatível do Apache Airflow.

Versões do Apache Airflow

As seguintes versões do Apache Airflow são compatíveis no Amazon Managed Workflows for Apache Airflow.

Note

- A partir do Apache Airflow v2.2.2, o Amazon MWAA oferece suporte à instalação de requisitos de Python, pacotes de provedores e plug-ins personalizados diretamente no servidor web Apache Airflow.
- A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.

  Para obter mais informações sobre como configurar restrições em seu arquivo de requisitos, consulte Instalando dependências do Python.

Versão do Apache Airflow	Guia do Apache Airflow	Restrições do Apache Airflow	Versão do Python
v2.8.1	Guia de referência do Apache Airflow v2.8.1	Arquivo de restrições do Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guia de referência do Apache Airflow v2.7.2	Arquivo de restrições do Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guia de referência do Apache Airflow v2.6.3	Arquivo de restrições do Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guia de referência do Apache Airflow v2.5.1	Arquivo de restrições do Apache Airflow v2.5.1	Python 3.10
v2.4.3	Guia de referência do Apache Airflow v2.4.3	Arquivo de restrições do Apache Airflow v2.4.3	Python 3.10

Versão do Apache Airflow	Guia do Apache Airflow	Restrições do Apache Airflow	Versão do Python
v2.2.2	Guia de referência do Apache Airflow v2.2.2	Arquivo de restrições do Apache Airflow v2.2.2	Python 3.7
v2.0.2	Guia de referência do Apache Airflow v2.0.2	Arquivo de restrições do Apache Airflow v2.0.2	Python 3.7

Para obter mais informações sobre como migrar suas implantações autogerenciadas do Apache Airflow ou migrar um ambiente Amazon MWAA existente, incluindo instruções para fazer backup do seu banco de dados de metadados, consulte Guia de migração do Amazon MWAA.

Componentes do Apache Airflow

Esta seção descreve o número de agendadores e operadores do Apache Airflow disponíveis para cada versão do Apache Airflow no Amazon MWAA e fornece uma lista dos principais atributos do Apache Airflow, indicando a versão que oferece suporte a cada atributo.

Programadores

Versão do Apache Airflow	Agendador (padrão)	Agendador (min)	Agendador (máx.)	
Apache Airflow v2 e superior	2	2	5	

Operadores

Versão Airflow	Operadores (min)	Operadores (máximo)	Operadores (padrão)	
Apache Airflow v2	1	25	10	

Atualizando a versão do Apache Airflow

O Amazon MWAA é compatível com atualizações de versões anteriores. Isso significa que você pode atualizar seu ambiente de uma versão x**.1.**z para x**.2.**z, mas não para uma nova versão principal, por exemplo, de **1.**y.z para**2.**y.z.

> **Note**
>
> Você não pode fazer regredir a versão do Apache Airflow para seu ambiente.

Para obter mais informações e instruções detalhadas sobre como atualizar seus recursos de fluxo de trabalho e atualizar o ambiente para uma nova versão, consulte the section called “Como atualizar a versão”.

Versões obsoletas do Apache Airflow

A tabela a seguir lista as versões obsoletas do Apache Airflow no Amazon MWAA, junto com as datas de lançamento inicial e término do suporte para cada versão. Para obter mais informações sobre a migração para uma nova versão do Apache Airflow, consulte o Guia de migração do Amazon MWAA.

Versão do Apache Airflow	Data de lançamento do Apache Airflow	Data de disponibilidade do Amazon MWAA	Data de suporte limitado do Amazon MWAA	Data de fim de suporte do Amazon MWAA
v1.10.12	25 de agosto de 2020	24 de novembro de 2020	21 de agosto de 2023	21 de fevereiro de 2024
v2.0.2	19 de abril de 2021	25 de maio de 2021	23 de novembro de 2023	29 de abril de 2024
v2.2.2	15 de novembro de 2021	27 de janeiro de 2022	25 de janeiro de 2024	27 de junho de 2024

Suporte à versão do Apache Airflow e perguntas frequentes

De acordo com o processo de lançamento e a política de versões da comunidade Apache Airflow, o Amazon MWAA está comprometido em oferecer suporte a pelo menos três versões menores do Apache Airflow a qualquer momento. Anunciaremos a data de encerramento do suporte de determinada versão secundária do Apache Airflow pelo menos 90 dias antes da data de término do suporte.

Perguntas frequentes

P: Por quanto tempo o Amazon MWAA oferece suporte a uma versão do Apache Airflow?

R: O Amazon MWAA oferece suporte a uma versão secundária do Apache Airflow por no mínimo 12 meses após a primeira disponibilização.

P: Receberei notificação quando o suporte estiver terminando para uma versão do Apache Airflow no Amazon MWAA?

R: Sim. Se algum ambiente Amazon MWAA em sua conta executar a versão próxima ao fim do suporte, o Amazon MWAA enviará um aviso até o dia do fim do AWS Health Dashboard suporte.

P: O que acontece na data de fim do suporte?

R: Na data de suporte limitado, não será mais possível criar novos ambientes do Amazon MWAA com a versão associada. Seus ambientes existentes continuarão disponíveis até a data de término do suporte.

P: O que acontece na data de fim do suporte?

R: No final da data de suporte, você continuará sendo capaz de acessar seus ambientes Amazon MWAA existentes que executam a versão associada e obsoleta do Apache Airflow por sua conta e risco. Para obter instruções sobre a atualização para uma versão mais recente do Apache Airflow no Amazon MWAA, consulte Guia de migração do Amazon MWAA.

Important

Você é responsável por manter suas versões do Amazon MWAA atualizadas. AWS exorta todos os clientes a atualizarem seus ambientes Amazon MWAA para a versão mais recente, a fim de se beneficiarem das mais atuais salvaguardas de segurança, privacidade e disponibilidade. Se você operar seu ambiente em uma versão ou software sem suporte após a data de descontinuação, chamada de versão legada, enfrentará uma maior probabilidade de riscos operacionais, de segurança e de privacidade, incluindo eventos de inatividade. Ao operar seu ambiente Amazon MWAA em uma versão legada, você confirma que compreende e assume conscientemente esses riscos e concorda em concluir sua atualização para a versão mais recente o mais rápido possível. A operação contínua de seu ambiente em uma versão antiga está sujeita ao contrato que rege o uso dos AWS serviços.
As versões antigas não são consideradas disponíveis ao público em geral e AWS não oferecem mais suporte para a versão antiga. Como resultado, AWS pode limitar o acesso ou o uso de qualquer versão antiga a qualquer momento, se AWS determinar que a versão antiga representa um risco de segurança ou responsabilidade, ou um risco de danos, aos serviços AWS, às suas afiliadas ou a qualquer outro terceiro. Sua decisão de continuar executando suas cargas de trabalho em uma versão antiga pode fazer com que seu conteúdo fique indisponível, corrompido ou irrecuperável. Ambientes executados em uma versão antiga estão sujeitos às exceções do Acordo de Nível de Serviço (SLA).
Ambientes e softwares relacionados executados em uma versão antiga podem conter bugs, erros, defeitos e componentes nocivos. Consequentemente, e não obstante qualquer informação em contrário no contrato ou nos termos de serviço, AWS fornece a versão antiga como está.
Para obter mais informações sobre o modelo de responsabilidade compartilhada AWS da, consulte Responsabilidade compartilhada no AWS Well-Architected Framework.

Endpoints e cotas de serviços do Amazon Managed Workflows for Apache Airflow

O Amazon Managed Workflows for Apache Airflow tem as seguintes service quotas e endpoints. As cotas de serviço, também chamadas de limites, são o número máximo de recursos ou operações de serviço da sua AWS conta.

Sumário

Service endpoints

Para ver uma lista de endpoints do Amazon MWAA, consulte Amazon Managed Workflows para endpoints e cotas do Apache Airflow.

Cotas de serviço

Nome da cota	Descrição	Cota padrão	Ajustável
Ambientes do	O número máximo de ambientes do Amazon MWAA por conta por região.	10	Sim
Operadores por ambiente	O número máximo de operadores por ambiente do Amazon MWAA.	25	Sim
Servidores Web por ambiente	O número máximo de servidores web por ambiente Amazon MWAA.	5	Sim

Aumento de cotas

Você pode solicitar um aumento para uma cota ajustável enviando uma solicitação de aumento de cota.

Perguntas frequentes sobre o Amazon MWAA

Esta página descreve perguntas comuns que você pode encontrar ao usar o Amazon Managed Workflows for Apache Airflow.

Sumário

- Quanto tempo o Amazon MWAA leva para reconhecer um novo arquivo DAG?
- Por que meu arquivo DAG não é captado pelo Apache Airflow?
- Posso remover plugins.zip ou requirements.txt de um ambiente?
- Por que não vejo meus plug-ins no menu Plugins de administração do Apache Airflow v2.0.2?
- Posso usar operadores do AWS Database Migration Service (DMS)?

Versões compatíveis

O que o Amazon MWAA oferece suporte para o Apache Airflow v2?

Para saber o que o Amazon MWAA oferece suporte, consulte Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow..

Por que as versões mais antigas do Apache Airflow não oferecem suporte?

Estamos oferecendo suporte apenas à versão mais recente (a partir do lançamento) do Apache Airflow, Apache Airflow v1.10.12, devido a questões de segurança com versões mais antigas.

Qual versão do Python devo usar?

As seguintes versões do Apache Airflow são compatíveis no Amazon Managed Workflows for Apache Airflow.

> **Note**
>
> - A partir do Apache Airflow v2.2.2, o Amazon MWAA oferece suporte à instalação de requisitos de Python, pacotes de provedores e plug-ins personalizados diretamente no servidor web Apache Airflow.
> - A partir do Apache Airflow v2.7.2, seu arquivo de requisitos deve incluir uma declaração `--constraint`. Se você não fornecer uma restrição, o Amazon MWAA especificará uma para garantir que os pacotes listados em seus requisitos sejam compatíveis com a versão do Apache Airflow que você está usando.
>
>   Para obter mais informações sobre como configurar restrições em seu arquivo de requisitos, consulte Instalando dependências do Python.

Versão do Apache Airflow	Guia do Apache Airflow	Restrições do Apache Airflow	Versão do Python
v2.8.1	Guia de referência do Apache Airflow v2.8.1	Arquivo de restrições do Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guia de referência do Apache Airflow v2.7.2	Arquivo de restrições do Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guia de referência do Apache Airflow v2.6.3	Arquivo de restrições do Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guia de referência do Apache Airflow v2.5.1	Arquivo de restrições do Apache Airflow v2.5.1	Python 3.10
v2.4.3	Guia de referência do Apache Airflow v2.4.3	Arquivo de restrições do Apache Airflow v2.4.3	Python 3.10
v2.2.2	Guia de referência do Apache Airflow v2.2.2	Arquivo de restrições do Apache Airflow v2.2.2	Python 3.7
v2.0.2	Guia de referência do Apache Airflow v2.0.2	Arquivo de restrições do Apache Airflow v2.0.2	Python 3.7

Para obter mais informações sobre como migrar suas implantações autogerenciadas do Apache Airflow ou migrar um ambiente Amazon MWAA existente, incluindo instruções para fazer backup do seu banco de dados de metadados, consulte Guia de migração do Amazon MWAA.

Qual versão de **`pip`** o Amazon MWAA usa?

Para ambientes que executam o Apache Airflow v1.10.12, o Amazon MWAA instala a versão de `pip` 21.1.2.

> **Note**
>
> O Amazon MWAA não atualiza `pip` para ambientes Apache Airflow v1.10.12.

Para ambientes que executam o Apache Airflow v2, o Amazon MWAA instala a versão de `pip` 21.3.1.

Casos de uso

Quando devo usar AWS Step Functions vs. Amazon MAA?

1. Você pode usar o Step Functions para processar pedidos individuais de clientes, já que o Step Functions pode ser escalado para atender à demanda de um ou de um milhão de pedidos.
2. Se você estiver executando um fluxo de trabalho noturno que processa os pedidos do dia anterior, poderá usar o Step Functions ou o Amazon MWAA. O Amazon MWAA permite que você tenha uma opção de código aberto para abstrair o fluxo de trabalho dos AWS recursos que você está usando.

Especificações do ambiente

Quanto armazenamento de tarefas está disponível para cada ambiente?

O armazenamento de tarefas é limitado a 10 GB e é especificado pelo Amazon ECS Fargate 1.3. A quantidade de RAM é determinada pela classe de ambiente especificada. Para obter mais informações sobre classes do ambiente, consulte Como configurar a classe de ambiente do Amazon MWAA.

Qual é o sistema operacional padrão usado para ambientes Amazon MWAA?

Os ambientes do Amazon MWAA são criados em instâncias que executam o Amazon Linux AMI.

Posso usar uma imagem personalizada para meu ambiente Amazon MWAA?

Imagens personalizadas não são compatíveis. O Amazon MWAA usa imagens criadas no Amazon Linux AMI. O Amazon MWAA instala os requisitos adicionais executando `pip3 -r install` para os requisitos especificados no arquivo requirements.txt que você adiciona ao bucket do Amazon S3 para o ambiente.

O Amazon MWAA está em conformidade com a HIPAA?

Amazon MWAA é elegível para a Lei de Portabilidade e Responsabilidade de Provedores de Saúde (HIPAA) dos EUA. Se você tiver um Adendo de Associado Comercial (BAA) da HIPAA em vigor com, AWS você pode usar o Amazon MWAA para fluxos de trabalho que lidam com Informações de Saúde Protegidas (PHI) em ambientes criados em ou após 14 de novembro de 2022.

O Amazon MWAA é compatível com instâncias spot?

Atualmente, o Amazon MWAA não é compatível com os tipos de instância spot sob demanda do Amazon EC2 para o Apache Airflow. No entanto, um ambiente Amazon MWAA pode acionar instâncias spot, por exemplo, no Amazon EMR e no Amazon EC2.

O Amazon MWAA é compatível com um domínio personalizado?

Para poder usar um domínio personalizado para seu nome de host do Amazon MWAA, faça uma das seguintes opções:

- Para implantações do Amazon MWAA com acesso ao servidor web público, você pode usar o Amazon com CloudFront Lambda @Edge para direcionar o tráfego para o seu ambiente e mapear um nome de domínio personalizado para. CloudFront Para obter mais informações e um exemplo de configuração de um domínio personalizado para um ambiente público, consulte o exemplo de domínio personalizado do Amazon MWAA para servidor web público no repositório de exemplos do Amazon MWAA. GitHub
- Para implantações do Amazon MWAA com acesso privado ao servidor web, você pode usar um Application Load Balancer (ALB) para direcionar o tráfego para o Amazon MWAA e mapear um nome de domínio personalizado para o ALB. Para ter mais informações, consulte the section called “Como usar um balanceador de carga (avançado)”.

Posso usar SSH no meu ambiente?

Embora o SSH não seja compatível em um ambiente Amazon MWAA, é possível usar um DAG para executar comandos bash usando o `BashOperator`. Por exemplo: .

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago
with DAG(dag_id="any_bash_command_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
      cli_command = BashOperator(
          task_id="bash_command",
          bash_command="{{ dag_run.conf['command'] }}"
      )
```

Para acionar o DAG na IU do Apache Airflow, use:

```
{ "command" : "your bash command"}
```

Por que uma regra de autorreferência é necessária no grupo de segurança da VPC?

Ao criar uma regra de autorreferência, você está restringindo a origem ao mesmo grupo de segurança na VPC e fechá-la para todas as redes. Para saber mais, consulte the section called “Segurança em suas VPC”.

Posso ocultar ambientes de grupos diferentes no IAM?

Você pode limitar o acesso especificando um nome de ambiente em AWS Identity and Access Management, no entanto, a filtragem de visibilidade não está disponível no AWS console. Se um usuário puder ver um ambiente, ele poderá ver todos os ambientes.

Posso armazenar dados temporários no operador do Apache Airflow?

Seus operadores do Apache Airflow podem armazenar dados temporários sobre os operadores. Os operadores do Apache Airflow podem acessar arquivos temporários no `/tmp` nos contêineres Fargate do seu ambiente.

> **Note**
>
> O armazenamento total de tarefas é limitado a 10 GB, de acordo com o Amazon ECS Fargate 1.3. Não há garantia de que as tarefas subsequentes serão executadas na mesma instância de contêiner Fargate, que pode usar uma pasta `/tmp` diferente.

Posso especificar mais de 25 operadores do Apache Airflow?

Sim. Embora você possa especificar até 25 operadores do Apache Airflow no console do Amazon MWAA, você pode configurar até 50 em um ambiente solicitando um aumento de cota. Consulte Requesting a quota increase (Como solicitar um aumento de cota) para obter mais informações.

O Amazon MWAA é compatível ao Amazon VPCs compartilhadas ou sub-redes compartilhadas?

O Amazon MWAA não é compatível ao Amazon VPCs compartilhadas ou sub-redes compartilhadas. A Amazon VPC que você seleciona ao criar um ambiente deve pertencer à conta que está tentando criar o ambiente. No entanto, é possível pode rotear o tráfego de um Amazon VPC na conta do Amazon MWAA para uma VPC compartilhada. Para obter mais informações e ver um exemplo de roteamento de tráfego para um Amazon VPC compartilhado, consulte Roteamento de saída centralizado para a Internet no Guia de gateways de trânsito do Amazon VPC.

Indicadores

Quais métricas são usadas para determinar se os operadores devem ser escalados?

O Amazon MWAA monitora o QueuedTaskse a entrada RunningTasks CloudWatch para determinar se o Apache Airflow Workers deve ser escalado em seu ambiente. Para saber mais, consulte Monitoramento e métricas.

Posso criar métricas personalizadas no CloudWatch?

Não no CloudWatch console. No entanto, você pode criar um DAG que grava métricas personalizadas em CloudWatch. Para ter mais informações, consulte the section called “Como usar um DAG para gravar métricas personalizadas”.

DAGs, operadores, conexões e outras perguntas

Posso usar `PythonVirtualenvOperator`?

O `PythonVirtualenvOperator` não é explicitamente compatível com o Amazon MWAA, mas você pode criar um plug-in personalizado que use `PythonVirtualenvOperator`. Para obter o código de exemplo, consulte the section called “Plugin personalizado para corrigir PythonVirtualEnvOperator ”.

Quanto tempo o Amazon MWAA leva para reconhecer um novo arquivo DAG?

Os DAGs são sincronizados periodicamente do bucket do Amazon S3 para seu ambiente. Se você adicionar um novo arquivo DAG, levará cerca de 300 segundos para que o Amazon MWAA comece a usar o novo arquivo. Se você atualizar um DAG existente, o Amazon MWAA levará cerca de 30 segundos para reconhecer suas atualizações.

Esses valores de 300 segundos para novos DAGs e 30 segundos para atualizações para DAGs existentes correspondem às opções de configuração do Apache Airflow `dag_dir_list_interval` e `min_file_process_interval`, respectivamente.

Por que meu arquivo DAG não é captado pelo Apache Airflow?

Algumas possíveis soluções são:

1. Verifique se seu perfil de execução tem permissões suficientes para seu bucket do Amazon S3. Para saber mais, consulte Perfil de execução do Amazon MWAA.
2. Verifique se o bucket do Amazon S3 tem o acesso público bloqueado configurado e o versionamento habilitado. Para saber mais, consulte Criar um bucket do Amazon S3 para o Amazon MWAA.
3. Verifique o arquivo DAG em si. Por exemplo, certifique-se de que cada DAG tenha uma ID de DAG exclusiva.

Posso remover `plugins.zip` ou `requirements.txt` de um ambiente?

Atualmente, não há como remover um plugins.zip ou requirements.txt de um ambiente depois de adicionados, mas estamos trabalhando para resolver o problema. Nesse ínterim, uma solução

alternativa é apontar para um texto vazio ou arquivo zip, respectivamente. Para saber mais, consulte Como excluir arquivos do Amazon S3.

Por que não vejo meus plug-ins no menu Plugins de administração do Apache Airflow v2.0.2?

Por motivos de segurança, o servidor Web do Apache Airflow no Amazon MWAA tem saída de rede limitada e não instala plug-ins nem dependências do Python diretamente no servidor web Apache Airflow para ambientes da versão 2.0.2. O plug-in exibido permite que o Amazon MWAA autentique seus usuários do Apache Airflow no (IAM). AWS Identity and Access Management

Para conseguir instalar plug-ins e dependências do Python diretamente no servidor web, recomendamos criar um novo ambiente com o Apache Airflow v2.2 e superior. O Amazon MWAA instala dependências e plug-ins personalizados do Python diretamente no servidor web para o Apache Airflow v2.2 e superior.

Posso usar operadores do AWS Database Migration Service (DMS)?

O Amazon MWAA é compatível a operadores de DMS. No entanto, esse operador não pode ser usado para realizar ações no banco de dados de metadados Amazon Aurora PostgreSQL associado a um ambiente Amazon MWAA.

Solução de problemas para Amazon Managed Workflows para Apache Airflow

Este tópico descreve problemas e erros comuns que você pode encontrar ao usar o Apache Airflow no Amazon Managed Workflows para Apache Airflow e as etapas recomendadas para resolver esses erros.

Sumário

Solução de problemas: DAGs, operadores, conexões e outros problemas no Apache Airflow v2

Os tópicos desta página descrevem resoluções para dependências do Apache Airflow v2 Python, plug-ins personalizados, DAGs, operadores, conexões, tarefas e problemas do servidor Web que você pode encontrar em um ambiente Amazon Managed Workflows for Apache Airflow.

Sumário

Conexões

O tópico a seguir descreve os erros que você pode receber ao usar uma conexão Apache Airflow ou ao usar outro banco de dados AWS.

Não consigo me conectar ao Secrets Manager

Recomendamos as seguintes etapas:

1. Aprenda a criar chaves secretas para sua conexão e variáveis do Apache Airflow em the section called “Como configurar o Secrets Manager”.
2. Saiba como usar a chave secreta para uma variável do Apache Airflow (`test-variable`) em Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow.
3. Saiba como usar a chave secreta para uma conexão Apache Airflow (`myconn`) em Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow.

Como configuro as condições do gerenciador de segredos **`secretsmanager:ResourceTag/<tag-key>`** ou uma restrição de recursos na minha política de perfil de execução?

> **Note**
>
> Aplica-se ao Apache Airflow versão 2.0 e anterior.

Atualmente, você não pode limitar o acesso aos segredos do Secrets Manager usando chaves de condição ou outras restrições de recursos no perfil de execução do seu ambiente, devido a um problema conhecido no Apache Airflow.

Não consigo me conectar ao Snowflake

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione as seguintes entradas ao requirements.txt para seu ambiente.

```
apache-airflow-providers-snowflake==1.3.0
```

3. Adicione as seguintes importações ao seu DAG:

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

Certifique-se de que o objeto de conexão Apache Airflow inclui os seguintes pares de chave-valor:

1. ID de conexão: snowflake_conn

2. Tipo de conexão: Snowflake
3. Host: <my account>.<my region if not us-west-2>.snowflakecomputing.com
4. Esquema: <my schema>
5. Login: <my user name>
6. Senha: ********
7. Porta: <port, if any>
8. Extra:

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

Por exemplo:

```
>>> import json
>>> from airflow.models.connection import Connection
>>> myconn = Connection(
...    conn_id='snowflake_conn',
...    conn_type='Snowflake',
...    host='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...    schema='YOUR_SCHEMA'
...    login='YOUR_USERNAME',
...    password='YOUR_PASSWORD',
...    port='YOUR_PORT'
...    extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Não consigo ver minha conexão na IU do Airflow

O Apache Airflow fornece modelos de conexão na IU do Apache Airflow. Isto é usado para gerar a string do URI de conexão, independentemente do tipo de conexão. Se um modelo de conexão não estiver disponível na IU do Apache Airflow, um modelo de conexão alternativo poderá ser usado para gerar uma string de URI de conexão, como o uso do modelo de conexão HTTP.

Recomendamos as seguintes etapas:

1. Veja os tipos de conexão que o Amazon MWAA fornece na IU do Apache Airflow em Pacotes do provedor Apache Airflow instalados em ambientes Amazon MWAA.
2. Veja os comandos para criar uma conexão Apache Airflow na CLI em Referência de comandos da CLI do Apache Airflow.
3. Aprenda a usar modelos de conexão na IU do Apache Airflow de forma intercambiável para tipos de conexão que não estão disponíveis na IU do Apache Airflow no Amazon MWAA em Visão geral dos tipos de conexão.

Servidor web

O tópico a seguir descreve os erros que você pode receber do seu servidor Web Apache Airflow no Amazon MWAA.

Eu vejo um erro 5xx ao acessar o servidor web

Recomendamos as seguintes etapas:

1. Verifique as opções de configuração do Apache Airflow. Verifique se os pares de chave-valor que você especificou como uma opção de configuração do Apache Airflow, como AWS Secrets Manager, foram configurados corretamente. Para saber mais, consulte the section called “Não consigo me conectar ao Secrets Manager”.
2. Verifique `requirements.txt`. Verifique se o pacote “extras” do Airflow e outras bibliotecas listadas no seu `requirements.txt` são compatíveis com sua versão do Apache Airflow.
3. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Eu vejo um erro “O agendador não parece estar em execução”

Se o programador não parecer estar em execução ou se o último “batimento” tiver sido recebido muitas horas antes, seus DAGs podem não aparecer no Apache Airflow e novas tarefas não serão programadas.

Recomendamos as seguintes etapas:

1. Confirme se seu grupo de segurança da VPC permite acesso de entrada à porta `5432`. Essa porta é necessária para se conectar ao banco de dados de metadados PostgreSQL do Amazon

Aurora para seu ambiente. Depois que essa regra for adicionada, aguarde alguns minutos para o Amazon MWAA e o erro deverá desaparecer. Para saber mais, consulte the section called "Segurança em suas VPC".

> **Note**
>
> - O banco de dados Aurora PostgreSQL faz parte da arquitetura de serviços do Amazon MWAA e não está visível em seu Conta da AWS.
> - Os erros relacionados ao banco de dados geralmente são um sintoma de falha do programador e não a causa raiz.

2. Se o programador não estiver em execução, pode ser devido a vários fatores, como falhas na instalação de dependências ou um programador sobrecarregado. Confirme se seus DAGs, plug-ins e requisitos estão funcionando corretamente ao visualizar os grupos de log correspondentes no CloudWatch Logs. Para saber mais, consulte Monitoramento e métricas.

Tarefas

O tópico a seguir descreve os erros que você pode receber nas tarefas do Apache Airflow em um ambiente.

Vejo minhas tarefas travadas ou não concluídas

Se suas tarefas do Apache Airflow estiverem "travadas" ou não estiverem sendo concluídas, recomendamos as seguintes etapas:

1. Pode haver um grande número de DAGs definidos. Reduza o número de DAGs e realize uma atualização do ambiente (como alterar um nível de log) para forçar uma reinicialização.

    a. O Airflow analisa os DAGs, estejam eles habilitados ou não. Se você estiver usando mais de 50% da capacidade do seu ambiente, você pode começar a sobrecarregar o programador do Apache Airflow. Isso leva a um grande tempo total de análise no CloudWatch Metrics ou a longos tempos de processamento do DAG no CloudWatch Logs. Há outras maneiras de otimizar as configurações do Apache Airflow que estão fora do escopo deste guia.

    b. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called "Ajuste de desempenho para o Apache Airflow".

2. Pode haver um grande número de tarefas na fila. Isso geralmente aparece como um grande (e crescente) número de tarefas no estado “Nenhum” ou como um grande número em Tarefas na fila e/ou Tarefas pendentes no CloudWatch. Este erro pode ocorrer pelos seguintes motivos:

   a. Se houver mais tarefas a serem executadas do que o ambiente tem a capacidade de executar e/ou um grande número de tarefas que foram colocadas em fila antes do ajuste de escala automático, você terá tempo para detectar as tarefas e implantar mais operadores.

   b. Se houver mais tarefas para executar do que um ambiente tem a capacidade de executar, recomendamos reduzir o número de tarefas que seus DAGs executam simultaneamente e/ou aumentar o número mínimo de operadores no Apache Airflow.

   c. Se houver um grande número de tarefas que foram colocadas em fila antes que o ajuste de escala automático tivesse tempo de detectar e implantar operadores adicionais, recomendamos intercalar a implantação de tarefas e/ou aumentar o número mínimo de operadores no Apache Airflow.

   d. Você pode usar o comando update-environment no AWS Command Line Interface (AWS CLI) para alterar o número mínimo ou máximo de Operadores que são executados em seu ambiente.

   ```
   aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-workers 10
   ```

   e. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called “Ajuste de desempenho para o Apache Airflow”.

3. Pode haver tarefas sendo excluídas no meio da execução que aparecem como logs de tarefas que param sem nenhuma indicação adicional no Apache Airflow. Este erro pode ocorrer pelos seguintes motivos:

   a. Se houver um breve momento em que 1) as tarefas atuais excedam a capacidade atual do ambiente, seguido por 2) alguns minutos sem nenhuma tarefa em execução ou na fila e 3) novas tarefas sendo colocadas na fila.

   b. O escalonamento automático do Amazon MWAA reage ao primeiro cenário adicionando mais operadores. No segundo cenário, ele remove os operadores adicionais. Algumas das tarefas que estão sendo colocadas na fila podem resultar na remoção dos operadores e terminarão quando o contêiner for excluído.

   c. Recomendamos aumentar o número mínimo de operadores em seu ambiente. Outra opção é ajustar o tempo de seus DAGs e tarefas para garantir que esses cenários não ocorram.

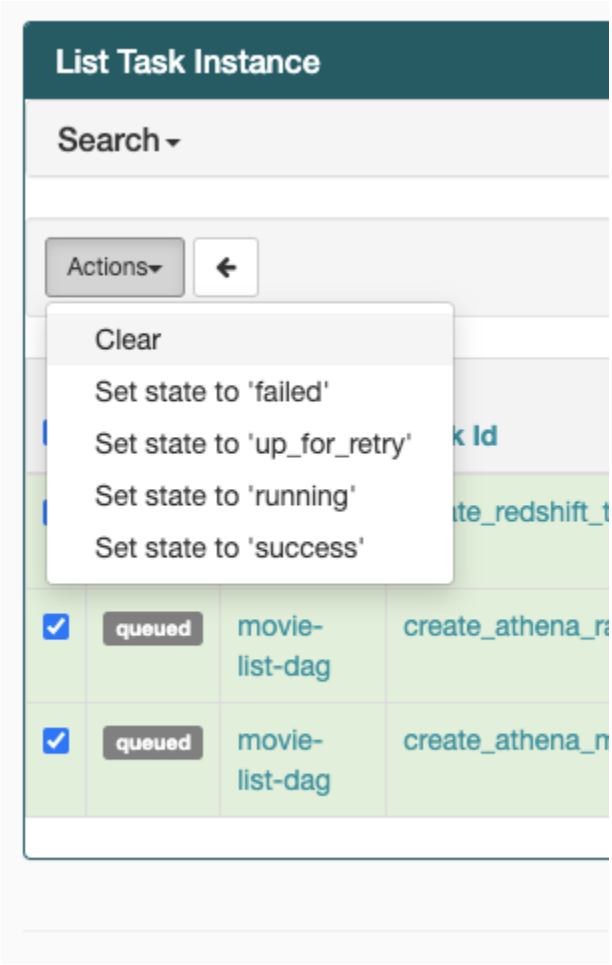
d. Você também pode definir o mínimo de operadores igual ao máximo de operadores em seu ambiente, desativando de maneira eficaz o ajuste de escala automático. Use o comando update-environment no AWS Command Line Interface (AWS CLI) para desabilitar o escalonamento automático definindo o número mínimo e máximo de operadores como sendo o mesmo.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-
workers 5
```

e. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called “Ajuste de desempenho para o Apache Airflow”.

4. Se suas tarefas estiverem travadas no estado “em execução”, você também poderá limpá-las ou marcá-las como bem-sucedidas ou malsucedidas. Isso permite que o componente de ajuste de escala automático do seu ambiente reduza a escala verticalmente do número de operadores em execução no seu ambiente. A imagem a seguir mostra um exemplo de uma tarefa perdida.

- Escolha o círculo para a tarefa perdida e selecione Limpar (conforme mostrado). Isso permite ao Amazon MWAA reduzir a escala verticalmente dos operadores; caso contrário, o Amazon MWAA não poderá determinar quais DAGs estão ativados ou desativados e não poderá reduzir a escala verticalmente se ainda houver tarefas na fila.



5. Saiba mais sobre o ciclo de vida das tarefas do Apache Airflow em Conceitos no guia de referência do Apache Airflow.

CLI

O tópico a seguir descreve os erros que você pode receber ao executar comandos da CLI do Airflow no AWS Command Line Interface.

Eu vejo um erro “503” ao acionar um DAG na CLI

A CLI do Airflow é executada no servidor Web do Apache Airflow, que tem simultaneidade limitada. Normalmente, no máximo 4 comandos da CLI podem ser executados simultaneamente.

Por que o comando da CLI **`dags backfill`** do Apache Airflow falha? Existe uma solução alternativa?

> **Note**
>
> O seguinte se aplica somente aos ambientes Apache Airflow v2.0.2.

O comando `backfill`, como outros comandos da CLI do Apache Airflow, analisa todos os DAGs localmente antes que qualquer DAG seja processado, independentemente do DAG ao qual a operação da CLI se aplica. Em ambientes Amazon MWAA usando o Apache Airflow v2.0.2, como os plug-ins e os requisitos ainda não estão instalados no servidor web no momento em que o comando CLI é executado, a operação de análise falha e a operação `backfill` não é invocada. Se você não tivesse nenhum requisito nem plug-ins em seu ambiente, a operação `backfill` seria bem-sucedida.

Para poder executar o comando da CLI `backfill`, recomendamos invocá-lo em um operador bash. Em um operador bash, `backfill` é iniciado pelo operador, permitindo que os DAGs sejam analisados com sucesso, pois todos os requisitos e plug-ins necessários estão disponíveis e instalados. O exemplo a seguir mostra como criar um DAG com `BashOperator` para executar `backfill`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="backfill_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="airflow dags backfill my_dag_id"
    )
```

Operadores

O tópico a seguir descreve os erros que você pode receber ao usar Operadores.

Recebi um erro **`PermissionError: [Errno 13] Permission denied`** ao usar o operador S3Transform

Recomendamos as etapas a seguir se você estiver tentando executar um script de shell com o operador S3Transform e estiver recebendo um erro `PermissionError: [Errno 13] Permission denied`. As etapas a seguir pressupõem que você tenha um arquivo plugins.zip existente. Se você estiver criando um novo plugins.zip, consulte Instalando plug-ins personalizados.

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Crie seu script de “transform” (transformação).

```
#!/bin/bash
cp $1 $2
```

3. (opcional) os usuários do macOS e do Linux podem precisar executar o comando a seguir para garantir que o script seja executável.

```
chmod 777 transform_test.sh
```

4. Adicione o script ao seu arquivo plugins.zip.

```
zip plugins.zip transform_test.sh
```

5. Siga as etapas em Carregar o plugins.zip para o Amazon S3.
6. Siga as etapas em Especificação da versão do plugins.zip no console do Amazon MWAA.
7. Crie o seguinte DAG.

```
from airflow import DAG
from airflow.providers.amazon.aws.operators.s3_file_transform import
 S3FileTransformOperator
from airflow.utils.dates import days_ago
import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    file_transform = S3FileTransformOperator(
        task_id='file_transform',
```

```
        transform_script='/usr/local/airflow/plugins/transform_test.sh',
        source_s3_key='s3://YOUR_S3_BUCKET/files/input.txt',
        dest_s3_key='s3://YOUR_S3_BUCKET/files/output.txt'
    )
```

8. Siga as etapas em Como fazer upload do código DAG para o Amazon S3.

Solução de problemas: DAGs, Operadores, Conexões e outros problemas no Apache Airflow v1

Os tópicos desta página contêm resoluções para dependências do v1.10.12 Python Apache Airflow, plug-ins personalizados, DAGs, Operadores, Conexões, tarefas e problemas de servidor Web que você pode encontrar em um ambiente do Amazon Managed Workflows for Apache Airflow.

Sumário

Como atualizar requirements.txt

O tópico a seguir descreve os erros que você pode receber ao atualizar seu `requirements.txt`.

Adicionar **`apache-airflow-providers-amazon`** faz com que meu ambiente falhe

`apache-airflow-providers-`**`xyz`** é compatível apenas com o Apache Airflow v2. `apache-airflow-backport-providers-`**`xyz`** é compatível com o Apache Airflow 1.10.12.

DAG quebrado

O tópico a seguir descreve os erros que você pode receber ao executar DAGs.

Recebi um erro “Broken DAG” (DAG quebrado) ao usar operadores do Amazon DynamoDB

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione o seguinte pacote ao seu `requirements.txt`.

   ```
   boto
   ```

3. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Recebi o erro “Broken DAG: No module named psycopg2” (DAG quebrado: Nenhum módulo chamado psycopg2)

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione o seguinte ao `requirements.txt` com sua versão do Apache Airflow. Por exemplo:

```
apache-airflow[postgres]==1.10.12
```

3. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Recebi um erro “Broken DAG” (DAG quebrado) ao usar os operadores do Slack

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione o seguinte pacote ao seu `requirements.txt` e especifique sua versão do Apache Airflow. Por exemplo:

```
apache-airflow[slack]==1.10.12
```

3. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Recebi vários erros ao instalar o Google/GCP/BigQuery

O Amazon MWAA usa o Amazon Linux, que exige uma versão específica do Cython e das bibliotecas de criptografia. Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione o seguinte pacote ao seu `requirements.txt`.

```
grpcio==1.27.2
```

```
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow-backport-providers-amazon[google]
```

3. Se não estiver usando provedores de backport, você pode usar:

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow[gcp]==1.10.12
```

4. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Recebi o erro “Broken DAG: No module named Cython” (DAG quebrado: nenhum módulo chamado Cython)

O Amazon MWAA usa o Amazon Linux, que exige uma versão específica do Cython. Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione o seguinte pacote ao seu `requirements.txt`.

```
cython==0.29.21
```

3. As bibliotecas Cython têm várias versões de dependência de pip necessárias. Por exemplo, usar `awswrangler==2.4.0` exige `pyarrow<3.1.0,>=2.0.0`, então o pip3 tenta instalar `pyarrow==3.0.0`, o que leva a um erro de DAG quebrado. Recomendamos especificar de maneira explícita a versão mais antiga aceitável. Por exemplo, se você especificar o valor mínimo `pyarrow==2.0.0` antes de `awswrangler==2.4.0`, o erro desaparecerá e `requirements.txt` será instalado corretamente. Os requisitos finais devem ser semelhantes aos seguintes:

```
cython==0.29.21
pyarrow==2.0.0
awswrangler==2.4.0
```

4. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Operadores

O tópico a seguir descreve os erros que você pode receber ao usar Operadores.

Recebi um erro ao usar o operador do BigQuery

O Amazon MWAA não oferece suporte a operadores com extensões de IU. Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Uma solução alternativa é substituir a extensão adicionando uma linha no DAG para definir `<operator name>.operator_extra_links = None` após a importação dos operadores com problemas. Por exemplo:

```
from airflow.contrib.operators.bigquery_operator import BigQueryOperator
BigQueryOperator.operator_extra_links = None
```

3. Você pode usar essa abordagem para todos os DAGs adicionando a opção acima a um plug-in. Para ver um exemplo, consulte the section called “Plugin personalizado para corrigir PythonVirtualEnvOperator ”.

Conexões

O tópico a seguir descreve os erros que você pode receber ao usar uma conexão Apache Airflow ou ao usar outro banco de dados AWS.

Não consigo me conectar ao Snowflake

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Adicione as seguintes entradas ao requirements.txt para seu ambiente.

```
asn1crypto == 0.24.0
```

```
snowflake-connector-python == 1.7.2
```

3. Adicione as seguintes importações ao seu DAG:

```
from airflow.contrib.hooks.snowflake_hook import SnowflakeHook
from airflow.contrib.operators.snowflake_operator import SnowflakeOperator
```

Certifique-se de que o objeto de conexão Apache Airflow inclui os seguintes pares de chave-valor:

1. ID de conexão: snowflake_conn
2. Tipo de conexão: Snowflake
3. Host: <my account>.<my region if not us-west-2>.snowflakecomputing.com
4. Esquema: <my schema>
5. Login: <my user name>
6. Senha: ********
7. Porta: <port, if any>
8. Extra:

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

Por exemplo:

```
>>> import json
>>> from airflow.models.connection import Connection
>>> myconn = Connection(
...    conn_id='snowflake_conn',
...    conn_type='Snowflake',
...    host='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...    schema='YOUR_SCHEMA'
...    login='YOUR_USERNAME',
...    password='YOUR_PASSWORD',
...    port='YOUR_PORT'
```

```
...    extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Não consigo me conectar ao Secrets Manager

Recomendamos as seguintes etapas:

1. Aprenda a criar chaves secretas para sua conexão e variáveis do Apache Airflow em the section called “Como configurar o Secrets Manager”.
2. Saiba como usar a chave secreta para uma variável do Apache Airflow (`test-variable`) em Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow.
3. Saiba como usar a chave secreta para uma conexão Apache Airflow (`myconn`) em Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow.

Não consigo me conectar ao meu servidor MySQL em “<DB-identifier-name>.cluster-id.<region>.rds.amazonaws.com”

O grupo de segurança do Amazon MWAA e o grupo de segurança do RDS precisam de uma regra de entrada para permitir o tráfego de e para o outro. Recomendamos as seguintes etapas:

1. Modifique o grupo de segurança do RDS para permitir todo o tráfego do grupo de segurança VPC do Amazon MWAA.
2. Modifique o grupo de segurança VPC do Amazon MWAA para permitir todo o tráfego do grupo de segurança RDS.
3. Execute novamente suas tarefas e verifique se o problema com o SQL foi bem-sucedido verificando os logs do Apache Airflow no CloudWatch Logs.

Servidor web

O tópico a seguir descreve os erros que você pode receber do seu servidor Web Apache Airflow no Amazon MWAA.

Estou usando o BigQueryOperator e isso está causando uma falha no meu servidor web

Recomendamos as seguintes etapas:

1. Operadores do Apache Airflow, como o `BigQueryOperator` e `QuboleOperator` que contêm `operator_extra_links`, podem fazer com que seu servidor web Apache Airflow falhe. Esses operadores tentam carregar código em seu servidor web, o que não é permitido por motivos de segurança. Recomendamos corrigir os operadores em seu DAG adicionando o seguinte código após suas instruções de importação:

   ```
   BigQueryOperator.operator_extra_links = None
   ```

2. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.

Eu vejo um erro 5xx ao acessar o servidor web

Recomendamos as seguintes etapas:

1. Verifique as opções de configuração do Apache Airflow. Verifique se os pares de chave-valor que você especificou como uma opção de configuração do Apache Airflow, como AWS Secrets Manager, foram configurados corretamente. Para saber mais, consulte the section called “Não consigo me conectar ao Secrets Manager”.
2. Verifique `requirements.txt`. Verifique se o pacote “extras” do Airflow e outras bibliotecas listadas no seu `requirements.txt` são compatíveis com sua versão do Apache Airflow.
3. Para explorar formas de especificar dependências do Python em um arquivo `requirements.txt`, consulte Como gerenciar dependências do Python em requirements.txt.

Eu vejo um erro “O agendador não parece estar em execução”

Se o programador não parecer estar em execução ou se o último “batimento” tiver sido recebido muitas horas antes, seus DAGs podem não aparecer no Apache Airflow e novas tarefas não serão programadas.

Recomendamos as seguintes etapas:

1. Confirme se seu grupo de segurança da VPC permite acesso de entrada à porta `5432`. Essa porta é necessária para se conectar ao banco de dados de metadados PostgreSQL do Amazon Aurora para seu ambiente. Depois que essa regra for adicionada, aguarde alguns minutos para o Amazon MWAA e o erro deverá desaparecer. Para saber mais, consulte the section called “Segurança em suas VPC”.

> **Note**
>
> - O banco de dados Aurora PostgreSQL faz parte da arquitetura de serviços do Amazon MWAA e não está visível em seu Conta da AWS.
> - Os erros relacionados ao banco de dados geralmente são um sintoma de falha do programador e não a causa raiz.

2. Se o programador não estiver em execução, pode ser devido a vários fatores, como falhas na instalação de dependências ou um programador sobrecarregado. Confirme se seus DAGs, plug-ins e requisitos estão funcionando corretamente ao visualizar os grupos de log correspondentes no CloudWatch Logs. Para saber mais, consulte Monitoramento e métricas.

Tarefas

O tópico a seguir descreve os erros que você pode receber nas tarefas do Apache Airflow em um ambiente.

Vejo minhas tarefas travadas ou não concluídas

Se suas tarefas do Apache Airflow estiverem “travadas” ou não estiverem sendo concluídas, recomendamos as seguintes etapas:

1. Pode haver um grande número de DAGs definidos. Reduza o número de DAGs e realize uma atualização do ambiente (como alterar um nível de log) para forçar uma reinicialização.

   a. O Airflow analisa os DAGs, estejam eles habilitados ou não. Se você estiver usando mais de 50% da capacidade do seu ambiente, você pode começar a sobrecarregar o programador do Apache Airflow. Isso leva a um grande tempo total de análise no CloudWatch Metrics ou a longos tempos de processamento do DAG no CloudWatch Logs. Há outras maneiras de otimizar as configurações do Apache Airflow que estão fora do escopo deste guia.

   b. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called “Ajuste de desempenho para o Apache Airflow”.

2. Pode haver um grande número de tarefas na fila. Isso geralmente aparece como um grande (e crescente) número de tarefas no estado “Nenhum” ou como um grande número em Tarefas na fila e/ou Tarefas pendentes no CloudWatch. Este erro pode ocorrer pelos seguintes motivos:

a. Se houver mais tarefas a serem executadas do que o ambiente tem a capacidade de executar e/ou um grande número de tarefas que foram colocadas em fila antes do ajuste de escala automático, você terá tempo para detectar as tarefas e implantar mais operadores.

b. Se houver mais tarefas para executar do que um ambiente tem a capacidade de executar, recomendamos reduzir o número de tarefas que seus DAGs executam simultaneamente e/ou aumentar o número mínimo de operadores no Apache Airflow.

c. Se houver um grande número de tarefas que foram colocadas em fila antes que o ajuste de escala automático tivesse tempo de detectar e implantar operadores adicionais, recomendamos intercalar a implantação de tarefas e/ou aumentar o número mínimo de operadores no Apache Airflow.

d. Você pode usar o comando update-environment no AWS Command Line Interface (AWS CLI) para alterar o número mínimo ou máximo de Operadores que são executados em seu ambiente.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-
workers 10
```

e. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called “Ajuste de desempenho para o Apache Airflow”.

3. Pode haver tarefas sendo excluídas no meio da execução que aparecem como logs de tarefas que param sem nenhuma indicação adicional no Apache Airflow. Este erro pode ocorrer pelos seguintes motivos:

a. Se houver um breve momento em que 1) as tarefas atuais excedam a capacidade atual do ambiente, seguido por 2) alguns minutos sem nenhuma tarefa em execução ou na fila e 3) novas tarefas sendo colocadas na fila.

b. O escalonamento automático do Amazon MWAA reage ao primeiro cenário adicionando mais operadores. No segundo cenário, ele remove os operadores adicionais. Algumas das tarefas que estão sendo colocadas na fila podem resultar na remoção dos operadores e terminarão quando o contêiner for excluído.

c. Recomendamos aumentar o número mínimo de operadores em seu ambiente. Outra opção é ajustar o tempo de seus DAGs e tarefas para garantir que esses cenários não ocorram.

d. Você também pode definir o mínimo de operadores igual ao máximo de operadores em seu ambiente, desativando de maneira eficaz o ajuste de escala automático. Use o comando update-environment no AWS Command Line Interface (AWS CLI) para desabilitar

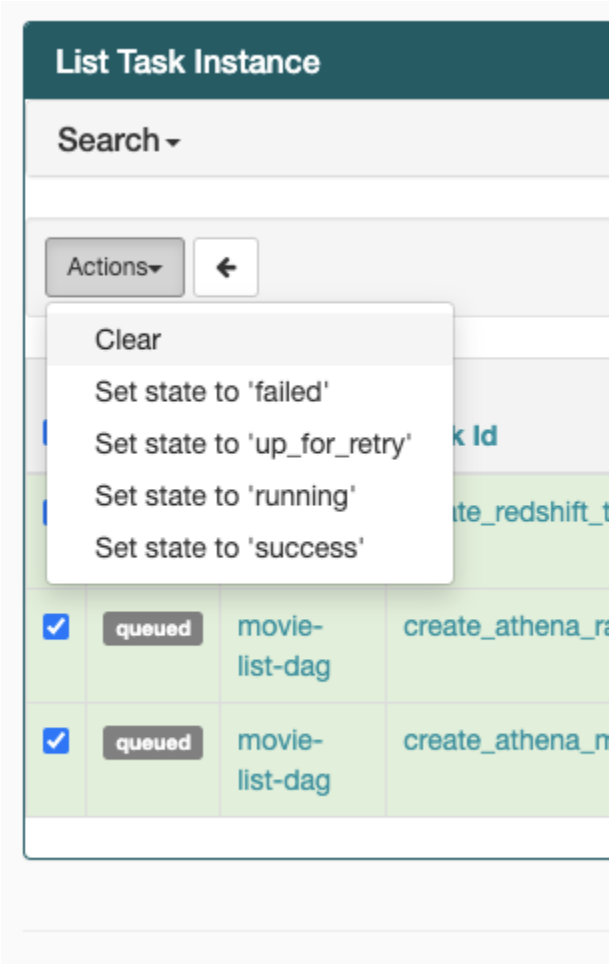
o escalonamento automático definindo o número mínimo e máximo de operadores como sendo o mesmo.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-
workers 5
```

e. Para saber mais sobre as práticas recomendadas para ajustar o desempenho do seu ambiente, consulte the section called “Ajuste de desempenho para o Apache Airflow”.

4. Se suas tarefas estiverem travadas no estado “em execução”, você também poderá limpá-las ou marcá-las como bem-sucedidas ou malsucedidas. Isso permite que o componente de ajuste de escala automático do seu ambiente reduza a escala verticalmente do número de operadores em execução no seu ambiente. A imagem a seguir mostra um exemplo de uma tarefa perdida.

- Escolha o círculo para a tarefa perdida e selecione Limpar (conforme mostrado). Isso permite ao Amazon MWAA reduzir a escala verticalmente dos operadores; caso contrário, o Amazon MWAA não poderá determinar quais DAGs estão ativados ou desativados e não poderá reduzir a escala verticalmente se ainda houver tarefas na fila.



5. Saiba mais sobre o ciclo de vida das tarefas do Apache Airflow em Conceitos no Guia de referência do Apache Airflow.

CLI

O tópico a seguir descreve os erros que você pode receber ao executar comandos da CLI do Airflow no AWS Command Line Interface.

Eu vejo um erro “503” ao acionar um DAG na CLI

A CLI do Airflow é executada no servidor Web do Apache Airflow, que tem simultaneidade limitada. Normalmente, no máximo 4 comandos da CLI podem ser executados simultaneamente.

Solução de problemas: Como criar e atualizar um ambiente do Amazon MWAA

Os tópicos desta página contêm erros que podem ser encontrados ao criar e atualizar um ambiente do Amazon Managed Workflows for Apache Airflow, e como resolver esses erros.

Sumário

Atualizar o `requirements.txt`

O tópico a seguir descreve os erros que você pode receber ao atualizar seu `requirements.txt`.

Especifiquei uma nova versão do meu `requirements.txt` e está demorando mais de 20 minutos para atualizar meu ambiente

Se o ambiente demorar mais de vinte minutos para instalar uma nova versão de um arquivo `requirements.txt`, a atualização do ambiente falhou e o Amazon MWAA está voltando para a última versão estável da imagem do contêiner.

1. Verifique as versões do pacote. Recomendamos sempre especificar uma versão específica (==) ou uma versão máxima (>=) para as dependências do Python em seu `requirements.txt`.
2. Verifique os logs do Apache Airflow. Se você habilitou os logs do Apache Airflow, verifique se seus grupos de logs foram criados com sucesso na página Grupos de logs no console do CloudWatch. Caso veja logs em branco, o motivo mais comum é a falta de permissões em seu perfil de execução para o CloudWatch ou o Amazon S3, onde os logs são gravados. Para saber mais, consulte Perfil de execução.
3. Verifique as opções de configuração do Apache Airflow. Caso esteja usando o Secrets Manager, verifique se os pares de chave-valor que você especificou como uma opção de configuração do Apache Airflow foram configurados corretamente. Para saber mais, consulte the section called “Como configurar o Secrets Manager”.
4. Verifique a configuração de rede da VPC. Para saber mais, consulte the section called “Ambiente travado”.
5. Verifique as permissões do perfil de execução. Um perfil de execução é um perfil do IAM AWS Identity and Access Management com uma política de permissões que concede permissão ao Amazon MWAA para invocar os recursos de outros serviços AWS (como Amazon S3, CloudWatch, Amazon SQS, Amazon ECR) em seu nome. Sua chave gerenciada pelo cliente ou AWS chave própria também precisa ter acesso permitido. Para saber mais, consulte Perfil de execução.
6. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.

Plug-ins

O tópico a seguir descreve os problemas que você pode encontrar ao configurar ou atualizar os plug-ins do Apache Airflow.

O Amazon MWAA oferece suporte à implementação de uma IU personalizada?

A partir do Apache Airflow v2.2.2, o Amazon MWAA oferece suporte à instalação de plug-ins no servidor web Apache Airflow e à implementação de uma IU personalizada. Se o seu ambiente Amazon MWAA estiver executando o Apache Airflow v2.0.2 ou anterior, você não poderá implementar uma IU personalizada.

Para obter mais informações sobre gerenciamento de versões e atualização de seus ambientes existentes, consulte Versões.

Consigo implementar alterações personalizadas na IU no executor local do Amazon MWAA por meio de plug-ins, mas quando tento fazer o mesmo no Amazon MWAA, não vejo minhas alterações nem erros. Por que isso está acontecendo?

O executor local do Amazon MWAA tem todos os componentes do Apache Airflow empacotados em uma imagem, permitindo que você aplique alterações personalizadas no plug-in de IU.

Criar bucket

O tópico a seguir descreve os erros que você pode receber ao criar um bucket do Amazon S3.

Não consigo selecionar a opção para configurações de bloqueio de acesso público do S3

O perfil de execução do seu ambiente Amazon MWAA precisa de permissão para a ação `GetBucketPublicAccessBlock` no bucket do Amazon S3 para verificar se o bucket bloqueou o acesso público. Recomendamos as seguintes etapas:

1. Siga as etapas para anexar uma política JSON ao seu perfil de execução.
2. Anexe a seguinte política do JSON:

```
{
   "Effect":"Allow",
   "Action":[
```

```
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*"
        ],
        "Resource":[
            "arn:aws:s3:::YOUR_S3_BUCKET_NAME",
            "arn:aws:s3:::YOUR_S3_BUCKET_NAME/*"
        ]
    }
```

Substitua os espaços reservados de amostra em *YOUR_S3_BUCKET_NAME* pelo nome do seu bucket Amazon S3, como *my-mwaa-unique-s3-bucket-name*.

3. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.

Criar o ambiente do

O tópico a seguir descreve os erros que você pode receber ao criar um ambiente.

Tentei criar um ambiente e ele está travado no estado “Criando”

Recomendamos as seguintes etapas:

1. Verifique a rede VPC com roteamento público. Se você estiver usando um Amazon VPC com acesso à Internet, verifique:

    - se seu Amazon VPC está configurado para permitir tráfego de rede entre os diferentes recursos da AWS usados pelo seu ambiente Amazon MWAA, conforme definido em the section called “Sobre a rede”. Por exemplo, seu grupo de segurança da VPC deve permitir todo o tráfego em uma regra de autorreferência ou, opcionalmente, especificar o intervalo de portas para o intervalo de portas HTTPS 443 e um intervalo de portas TCP 5432.
2. Verifique a rede VPC com roteamento privado. Se você estiver usando um Amazon VPC sem acesso à Internet, verifique:

    - se seu Amazon VPC está configurado para permitir tráfego de rede entre os diferentes recursos da AWS para seu ambiente Amazon MWAA, conforme definido em the section called “Sobre a rede”. Por exemplo, suas duas sub-redes privadas não devem ter uma tabela de rotas para um gateway NAT (ou instância NAT), nem um gateway da Internet.

3. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.

Tentei criar um ambiente, mas ele mostra o status como “Falha na criação”

Recomendamos as seguintes etapas:

1. Verifique a configuração de rede da VPC. Para saber mais, consulte the section called “Ambiente travado”.
2. Verifique as permissões de usuário. O Amazon MWAA executa um teste com base nas credenciais do usuário antes de criar um ambiente. Sua conta da AWS pode não ter permissão no AWS Identity and Access Management (IAM) para criar alguns dos recursos para um ambiente. Por exemplo, se você escolheu o modo de acesso do Apache Airflow à rede privada, sua conta AWS deve ter o acesso concedido por seu administrador para a política de controle de acesso AmazonMWAAFullConsoleAccess para seu ambiente, o que permite que sua conta crie endpoints da VPC .
3. Verifique as permissões do perfil de execução. Um perfil de execução é um perfil do IAM AWS Identity and Access Management com uma política de permissões que concede permissão ao Amazon MWAA para invocar os recursos de outros serviços AWS (como Amazon S3, CloudWatch, Amazon SQS, Amazon ECR) em seu nome. Sua chave gerenciada pelo cliente ou AWS chave própria também precisa ter acesso permitido. Para saber mais, consulte Perfil de execução.
4. Verifique os logs do Apache Airflow. Se você habilitou os logs do Apache Airflow, verifique se seus grupos de logs foram criados com sucesso na página Grupos de logs no console do CloudWatch. Caso veja logs em branco, o motivo mais comum é a falta de permissões em seu perfil de execução para o CloudWatch ou o Amazon S3, onde os logs são gravados. Para saber mais, consulte Perfil de execução.
5. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.
6. Se você estiver usando um Amazon VPC sem acesso à Internet, certifique-se de ter criado um endpoint de gateway do Amazon S3 e concedido as permissões mínimas necessárias ao Amazon ECR para acessar o Amazon S3. Para saber mais sobre como criar um endpoint do gateway do Amazon S3, consulte:

- Criação de uma rede Amazon VPC sem acesso à Internet
- Criação de endpoint de gateway para o Amazon S3 no Guia do usuário do Amazon Elastic Container Registry

Tentei selecionar uma VPC e recebi o erro “Falha de rede”

Recomendamos as seguintes etapas:

- Se você vir um erro de “Falha de rede” ao tentar selecionar um Amazon VPC ao criar seu ambiente, desative todos os proxies no navegador que estejam em execução e tente novamente.

Tentei criar um ambiente e recebi um erro de serviço, partição ou recurso “deve ser transferido”

Recomendamos as seguintes etapas:

- É possível que esteja recebendo este erro porque o URI que você especificou para seu bucket do Amazon S3 inclui um “/” no final do URI. Recomendamos remover o “/” no caminho. O valor deve estar no seguinte formato:

```
s3://your-bucket-name
```

Tentei criar um ambiente e ele mostra o status como “Disponível”, mas quando tento acessar a IU do Airflow, aparece um erro de “Resposta vazia do servidor” ou “502 Bad Gateway”

Recomendamos as seguintes etapas:

1. Verifique a configuração do grupo de segurança da VPC. Para saber mais, consulte the section called “Ambiente travado”.
2. Confirme se todos os pacotes do Apache Airflow que você listou em `requirements.txt` correspondem à versão do Apache Airflow que você está executando no Amazon MWAA. Para saber mais, consulte Como instalar dependências do Python.

3. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.

Tentei criar um ambiente e meu nome de usuário é um monte de caracteres aleatórios

- O Apache Airflow tem um máximo de 64 caracteres para nomes de usuário. Se seu perfil do IAM AWS Identity and Access Management exceder esse tamanho, um algoritmo de hash será usado para reduzi-lo, permanecendo exclusivo.

Atualizar ambiente

O tópico a seguir descreve os erros que você pode receber ao atualizar um ambiente.

Tentei mudar a classe do ambiente, mas a atualização falhou

Se você atualizar seu ambiente para uma classe de ambiente diferente (como alterar um `mw1.medium` para um `mw1.small`) e a solicitação de atualização do ambiente falhar, o status do ambiente entrará em um estado `UPDATE_FAILED` e o ambiente será revertido e cobrado de acordo com a versão estável anterior de um ambiente.

Recomendamos as seguintes etapas:

1. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.
2. Para executar um script de solução de problemas que verifique a configuração da rede Amazon VPC e a configuração para seu ambiente Amazon MWAA, consulte o script Verificar ambiente nas AWSferramentas de suporte no GitHub.

Ambiente de acesso

O tópico a seguir descreve os erros que você pode receber ao acessar um ambiente.

Não consigo acessar a IU do Apache Airflow

Recomendamos as seguintes etapas:

1. Verifique as permissões de usuário. Talvez você não tenha recebido acesso a uma política de permissões que permite visualizar a IU do Apache Airflow. Para saber mais, consulte the section called “Como acessar um ambiente do Amazon MWAA”.
2. Verifique o acesso à rede. A causa pode ser porque você selecionou o modo de acesso à rede privada. Se a URL da sua IU do Apache Airflow estiver no formato `387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-west-2.airflow.amazonaws.com`, significa que você está usando roteamento privado para o seu servidor Web Apache Airflow. Você pode atualizar o modo de acesso do Apache Airflow para o modo de acesso à rede pública ou criar um mecanismo para acessar o endpoint da VPC do seu servidor Web Apache Airflow. Para saber mais, consulte the section called “Como gerenciar o acesso às endpoints da VPC”.

Solução de problemas: erros do CloudWatch Logs e do CloudTrail

Os tópicos desta página contêm resoluções para o Amazon CloudWatch Logs e erros AWS CloudTrail que você pode encontrar em um ambiente Amazon Managed Workflows for Apache Airflow.

Sumário

Logs

O tópico a seguir descreve os erros que você pode receber ao visualizar os logs do Apache Airflow.

Não consigo ver meus registros de tarefas ou recebi o erro “Lendo o log remoto do Cloudwatch log_group”

O Amazon MWAA configurou o Apache Airflow para ler e gravar logs diretamente de e para o Amazon CloudWatch Logs. Se um operador falhar ao iniciar uma tarefa ou não conseguir gravar nenhum log, você verá o erro:

```
*** Reading remote log from Cloudwatch log_group: airflow-environmentName-Task
 log_stream: DAG_ID/TASK_ID/timestamp/n.log.Could not read remote logs from log_group:
 airflow-environmentName-Task log_stream: DAG_ID/TASK_ID/time/n.log.
```

- Recomendamos as seguintes etapas:

  a. Verifique se você ativou os logs de tarefas no nível `INFO` do seu ambiente. Para obter mais informações, consulte Visualizando registros de fluxo de ar na Amazon CloudWatch.

  b. Verifique se o perfil de execução do ambiente tem as políticas de permissão corretas.

  c. Verifique se seu operador ou tarefa está funcionando corretamente, tem recursos suficientes para analisar o DAG e tem as bibliotecas Python apropriadas para carregar. Para verificar se você tem as dependências corretas, tente eliminar as importações até encontrar a que está causando o problema. Recomendamos testar suas dependências do Python usando a ferramenta Amazon MWAA local-runner.

As tarefas estão falhando sem nenhum log

Se as tarefas estiverem falhando em um fluxo de trabalho e você não conseguir localizar nenhum log das tarefas com falha, verifique se você está definindo o parâmetro `queue` em seus argumentos padrão, conforme mostrado a seguir.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

# Setting queue argument to default.
default_args = {
 "start_date": days_ago(1),
```

```
 "queue": "default"
}

with DAG(dag_id="any_command_dag", schedule_interval=None, catchup=False,
 default_args=default_args) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="{{ dag_run.conf['command'] }}"
    )
```

Para resolver o problema, remova `queue` do seu código e invoque o DAG novamente.

Vejo um erro “ResourceAlreadyExistsException” no CloudTrail

```
"errorCode": "ResourceAlreadyExistsException",
    "errorMessage": "The specified log stream already exists",
    "requestParameters": {
        "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
        "logStreamName": "scheduler_cross-account-eks.py.log"
    }
```

Certos requisitos do Python, como `apache-airflow-backport-providers-amazon` reverter a biblioteca `watchtower` que o Amazon MWAA usa para se comunicar com o CloudWatch para uma versão mais antiga. Recomendamos as seguintes etapas:

- Adicione a seguinte biblioteca ao seu `requirements.txt`.

  ```
  watchtower==1.0.6
  ```

Eu vejo um erro de “Solicitação inválida” no CloudTrail

```
Invalid request provided: Provided role does not have sufficient permissions for s3
 location airflow-xxx-xxx/dags
```

Se você estiver criando um ambiente Amazon MWAA e um bucket Amazon S3 usando o mesmo modelo AWS CloudFormation, você precisa adicionar uma seção `DependsOn` dentro do seu modelo AWS CloudFormation. Os dois recursos (MWAA Environment e MWAA Execution Policy) têm uma dependência em AWS CloudFormation. Recomendamos as seguintes etapas:

- Adicione a seguinte declaração **`DependsOn`** ao seu modelo AWS CloudFormation.

```
...
      MaxWorkers: 5
      NetworkConfiguration:
        SecurityGroupIds:
          - !GetAtt SecurityGroup.GroupId
        SubnetIds: !Ref subnetIds
      WebserverAccessMode: PUBLIC_ONLY
    DependsOn: MwaaExecutionPolicy

    MwaaExecutionPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
        - !Ref MwaaExecutionRole
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action: airflow:PublishMetrics
            Resource:
...
```

Para ver um exemplo, consulte Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow.

Eu vejo uma mensagem “Não é possível localizar uma biblioteca Oracle Client de 64 bits: “libclntsh.so: não é possível abrir o arquivo de objeto compartilhado: esse arquivo ou diretório não existe” nos logs do Apache Airflow

- Recomendamos as seguintes etapas:
  - Se você estiver usando o Apache Airflow v2, adicione `core.lazy_load_plugins : False` como uma opção de configuração do Apache Airflow. Para saber mais, consulte Usando opções de configuração para carregar plug-ins em 2.

Vejo psycopg2 'o servidor fechou a conexão inesperadamente' nos meus logs do Agendador

Se você vir um erro semelhante a esse, seu agendador do Apache Airflow pode ter ficado sem recursos.

```
2021-06-14T10:20:24.581-05:00    sqlalchemy.exc.OperationalError:
 (psycopg2.OperationalError) server closed the connection unexpectedly
2021-06-14T10:20:24.633-05:00    This probably means the server terminated abnormally
2021-06-14T10:20:24.686-05:00    before or while processing the request.
```

Recomendamos as seguintes etapas:

- Considere a atualização para o Apache Airflow v2.0.2, que permite especificar até 5 agendadores.

Vejo “O executor relata a instância da tarefa %s concluída (%s), embora a tarefa diga que é %s” nos meus registros de processamento do DAG

Se você ver um erro semelhante ao seguinte, suas tarefas de longa execução podem ter atingido o limite de tempo da tarefa no Amazon MWAA. O Amazon MWAA tem um limite de 12 horas para qualquer tarefa do Airflow, para evitar que as tarefas fiquem presas na fila e bloqueiem atividades como escalonamento automático.

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
 %s) Was the task killed externally
```

Recomendamos as seguintes etapas:

- Considere dividir a tarefa em várias tarefas de execução mais curtas. O Airflow normalmente tem um modelo em que os operadores são assíncronos. Ele invoca atividades em sistemas externos e a pesquisa Apache Airflow Sensors para ver quando está concluída. Se um sensor falhar, ele poderá ser repetido com segurança sem afetar a funcionalidade do operador.

Eu vejo 'Não foi possível ler os logs remotos do log_group: airflow-* {*environmentName} -Task log_stream: * {*DAG_ID} /* {*TASK_ID} /* {*time} /* {*n} .log. ' nos meus logs de tarefas

Se você ver um erro semelhante ao que se segue, o perfil de execução do seu ambiente pode não conter uma política de permissões para criar fluxos de log para logs de tarefas.

```
Could not read remote logs from log_group: airflow-*{*environmentName}-Task
 log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

Recomendamos as seguintes etapas:

- Modifique o perfil de execução do seu ambiente usando um dos exemplos de políticas em the section called “Perfil de execução”.

Você também pode ter especificado um pacote de provedor em seu `requirements.txt` arquivo que é incompatível com sua versão do Apache Airflow. Por exemplo, se você estiver usando o Apache Airflow v2.0.2, você pode ter especificado um pacote, como o pacote apache-airflow-providers-databricks, que só é compatível com o Airflow 2.1+.

Recomendamos as seguintes etapas:

1. Se você estiver usando o Apache Airflow v2.0.2, modifique o `requirements.txt` arquivo e adicione `apache-airflow[databricks]`. Isso instala a versão correta do pacote Databricks que é compatível com o Apache Airflow v2.0.2.
2. Teste seus DAGs, plug-ins personalizados e dependências do Python localmente usando aws-mwaa-local-runner no GitHub.

Histórico de documentos do Amazon MWAA

A tabela a seguir descreve adições importantes feitas na documentação do serviço do Amazon MWAA, a partir de novembro de 2020. Para receber notificações sobre atualizações dessa documentação, inscreva-se no feed RSS.

Alteração	Descrição	Data
O Amazon MWAA oferece suporte à escalabilidade automática de servidores web e à API REST do Apache Airflow	O Amazon MWAA agora oferece suporte à escalabilidade automática de servidores web, bem como à capacidade de acessar e usar a API REST do Apache Airflow. • the section called “Configurando o escalonamento automático do servidor web” • the section called “Usando a API REST do Apache Airflow”	16 de maio de 2024
Descrição aprimorada do comportamento de escalonamento automático	O tópico a seguir foi atualizado para refletir o novo comportamento de escalabilidade automática do Amazon MWAA quando os trabalhadores assumem novas tarefas à medida que os trabalhadores da Fargate reduzem a escala. • the section called “Configurando o escalonamento automático do trabalhador”	10 de maio de 2024
Support para tamanhos de instância maiores	O Amazon MWAA agora oferece suporte a duas	16 de abril de 2024

	opções maiores de tamanho de instância para cargas de trabalho maiores:, e `mw1.xlarge mw1.2xlarge` • the section called “Funcionalidades do ambiente”	
Nova versão do Apache Airflow	O Amazon MWAA agora oferece suporte ao Apache Airflow v2.8.1. Essa atualização inclui informações sobre pacotes de provedores atualizados e detalhes sobre o uso do Apache Airflow v2.8.1 no Amazon MWAA. • Versões • the section called “Pacotes de provedores para conexões do Apache Airflow v2.8.1”	22 de fevereiro de 2024

Suporte à Amazon VPC compartilhada	O Amazon MWAA oferece suporte à criação de ambientes entre contas para organizações que usam o Amazon OpenSearch Service para gerenciar recursos do Amazon MWAA usando uma Amazon VPC central compartilhada em uma conta de proprietário. Como parte desse lançamento, o Amazon MWAA permite que você escolha criar e gerenciar seus próprios endpoints da VPC do Amazon. • the section called “Gerenciando seus próprios endpoints Amazon VPC”	15 de novembro de 2023
Nova versão do Apache Airflow	O Amazon MWAA agora oferece suporte ao Apache Airflow v2.7.2. Essa atualização inclui informações sobre pacotes de provedores atualizados e detalhes sobre o uso do Apache Airflow v2.7.2 no Amazon MWAA. • Versões • the section called “Pacotes de provedores para conexões Apache Airflow v2.7.2”	6 de novembro de 2023

Nova versão do Apache Airflow	O Amazon MWAA agora oferece suporte ao Apache Airflow v2.6.3. Essa atualizaç ão inclui informações sobre pacotes de provedores atualizados e detalhes sobre o uso do Apache Airflow v2.6.3 no Amazon MWAA. • Versões • the section called “Pacotes de provedores para conexões Apache Airflow v2.6.3”	9 de agosto de 2023
Informações sobre descontin uação de versões	Atualizado o tópico sobre a descontinuação da versão para incluir avisos de descontinuação e cronogram as para o Apache Airflow v2.0.2 e o Apache Airflow v2.2.2. • the section called “Versões obsoletas do Apache Airflow”	31 de julho de 2023

Novos tópicos e casos de uso	O Amazon MWAA é compatível com atualizaç ões de versões anteriores. Essas atualizações incluem o seguinte novo tópico que descreve como atualizar o ambiente e garantir que seus recursos de fluxo de trabalho sejam compatíveis com a versão do Apache Airflow para a qual você está atualizando: • the section called “Como atualizar a versão”	5 de junho de 2023
Tópico atualizado	Atualizadas as políticas do IAM gerenciadas pelo cliente que concedem ao usuário acesso completo ao console e à API do Amazon MWAA. A atualização descreve por que você deve fornecer permissão para `iam:PassRole` para permitir que um usuário transfira perfis para o Amazon MWAA. O Amazon MWAA usa essas permissões para realizar ações em nome do usuário. • the section called “Como acessar um ambiente do Amazon MWAA”	12 de abril de 2023

Nova orientação	Tópico atualizado sobre a configuração AWS Secrets Manager como back-end do Amazon MWAA para fornecer orientação sobre o uso de padrões de pesquisa. O uso de padrões de pesquisa restringe os segredos que o Apache Airflow pesquisa e reduz o número de chamadas de API que o Amazon MWAA faz ao Secrets Manager para recuperar conexões e variáveis. Isso reduz os custos associados ao uso do Secrets Manager como back-end. • Criar o back-end do Secrets Manager como uma opção de configuração do Apache Airflow	12 de abril de 2023
Nova versão do Apache Airflow	O Amazon MWAA agora oferece suporte ao Apache Airflow v2.5.1. Essa atualizaç ão inclui informações sobre pacotes de provedores atualizados e detalhes sobre o uso do Apache Airflow v2.5.1 no Amazon MWAA. • Versões • the section called “Pacotes de provedores para conexões Apache Airflow v2.5.1”	11 de abril de 2023

Novos tópicos e casos de uso	Adicionado um novo tópico sobre o uso de um script de startup com um ambiente do Amazon MWAA. Este tópico descreve como configurar um script de startup para um ambiente existente, usando-o para instalar runtimes do Linux e definir variáveis de ambiente. • the section called “Como usar um script de startup”	3 de abril de 2023
Atualizada a seção sobre acesso a servidores web privados	Atualizado o seguinte tópico sobre acesso a servidores web privados. A atualização esclarece que, em ambientes com acesso privado ao servidor web, você deve usar um repositório wheel do Python (`.whl`) para empacotar e instalar dependências. • Modo de acesso privado ao servidor web	24 de fevereiro de 2023

Adicionadas informações sobre versões obsoletas do Apache Airflow	Atualizado o tópico Versões com novas informações sobre como o Amazon MWAA gerenciou versões obsoletas do Apache Airflow. Removida seção sobre a atualização para uma versão mais recente do Apache Airflow e seção que descrevia as mudanças entre o Apache Airflow v1 e o Apache Airflow v2. Para obter mais informações sobre a migração para uma nova versão do Apache Airflow, consulte o Guia de migração do Amazon MWAA. • the section called “Versões obsoletas do Apache Airflow” • the section called “Suporte à versão do Apache Airflow e perguntas frequentes”	17 de fevereiro de 2023

Correções nas métricas de contêiner do Amazon MWAA	Atualizado o tópico de métricas de contêiner e removido um conjunto de métricas erradas que não existiam na dimensão `Cluster`. Adicionada uma seção adicional que descreve como você pode avaliar o número de operadores adicionais que um ambiente está utilizando em um determinado momento ao representar graficamente a métrica `CPUUtilization` ou a `MemoryUtilization` do componente `AdditionalWorker` e definir o tipo de estatística como `SampleCount`. • the section called “Avaliando o número de contêineres adicionais de trabalhadores e servidores web”	20 de janeiro de 2023

Nova versão do Apache Airflow	O Amazon MWAA agora oferece suporte ao Apache Airflow v2.4.3. Essa atualizaç ão inclui informações sobre pacotes de fornecedores atualizados, detalhes sobre o uso do Apache Airflow v2.4.3 no Amazon MWAA e informações consolidadas sobre quais recursos são compatíveis com cada versão do Apache Airflow no Amazon MWAA. • Versões • the section called “Pacotes de provedores para conexões Apache Airflow v2.4.3”	5 de janeiro de 2023

Tópico atualizado sobre perfil vinculado a serviços	Informações atualizadas sobre a função vinculada ao serviço que a Amazon MWAA usa para criar e gerenciar AWS recursos em seu nome, incluindo informações sobre como você pode excluir a função vinculada ao serviço quando não precisar mais dela. Isso inclui uma política atualizada de permissão de função vinculada ao serviço que permite que a Amazon MWAA publique métricas adicionais CloudWatch sob o namespace. `AWS/MWAA` • the section called “Perfil vinculado a serviço”	18 de novembro de 2022

Novo tópico sobre métricas de serviço	Adicionado um novo tópico que descreve as métricas de serviço emitidas pelo Amazon MWAA no namespace `AWS/MWAA`. São incluídos agendadores, operadores e servidores web de métricas de cluster do Amazon ECS, métricas do Amazon SQS para as filas que permitem ao Amazon MWAA separar agendadores e operadore s, bem como métricas do Amazon RDS para o banco de dados de metadados. • the section called “Métricas de contêiner, fila e banco de dados”	18 de novembro de 2022
Novo tópico	Adicionadas novas orientaçõ es sobre a modificação de um arquivo de restrições para especificar novas versões dos pacotes do provedor para usar com seu ambiente Amazon MWAA. • the section called “Como especificar pacotes de fornecedores mais novos”	18 de novembro de 2022

Entrada de perguntas frequentes atualizada	Atualizadas as informações relacionadas à elegibilidade do Amazon MWAA para a HIPAA. • the section called “Conformidade com a HIPAA”	15 de novembro de 2022
Novo tópico	Adicionado um novo tópico sobre o uso de chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em uma política de confiança di perfil de execução do Amazon MWAA, a fim de evitar a confusão entre serviços. • the section called “Prevenção contra o ataque do “substituto confuso” em todos os serviços”	21 de outubro de 2022
Novo código de exemplo	Foram adicionadas instruções atualizadas e um exemplo de código DAG que grava métricas personalizadas no nível do sistema operacional em. CloudWatch • the section called “Como usar um DAG para gravar métricas personalizadas”	13 de setembro de 2022

Novo código de exemplo	Foram adicionadas instruções atualizadas e um novo exemplo de código AWS Lambda Python que recupera um token CLI do Apache Airflow e, em seguida, invoca um DAG em um ambiente específico do Amazon MWAA. • the section called “Como invocar DAGs com Lambda”	12 de setembro de 2022
Novos diagramas arquitetônicos	Adicionados novos diagramas de arquitetura que demonstram um ambiente Amazon MWAA com um servidor web público e privado. • the section called “Modos de acesso do Apache Airflow”	12 de setembro de 2022
Novo código de exemplo	Adicionadas instruções atualizadas e um novo exemplo de código do DAG que recupera um token CLI do Apache Airflow e, em seguida, invoca outro DAG em um ambiente diferente do Amazon MWAA. • the section called “Como invocar DAGs em diferentes ambientes”	16 de agosto de 2022

Novo código de exemplo	Adicionadas instruções atualizadas e um novo DAG que consulta o Aurora PostgreSQL de um ambiente para obter informações de metadados, grava o resultado em arquivos CSV e armazena os arquivos no Amazon S3. • the section called “Como exportar metadados do ambiente para Amazon S3”	12 de agosto de 2022
Novo código de exemplo	Foram adicionadas instruções atualizadas e um novo DAG que atualiza um AWS CodeArtifact token em tempo de execução e armazena o resultado no Amazon S3. • the section called “Como atualizar um token AWS CodeArtifact em runtime”	3 de agosto de 2022
Novo código de exemplo	Adicionadas instruções atualizadas e exemplo de código DAG para uso de `ECSOperator` no Amazon MWAA. • the section called “Usar a `ECSOperator` ”	26 de julho de 2022

Novo código de exemplo	Adicionadas instruções atualizadas e exemplo de código DAG para uso de `SSHOperator` no Amazon MWAA. • the section called “Como usar o `SSHOperator` ”	15 de julho de 2022
Novo código de exemplo	Adicionadas novas instruções e exemplo de código DAG para usar o dbt Postgres com o Amazon MWAA. • the section called “Como usar DBT com o Amazon MWAA”	17 de junho de 2022
Novos tópicos e casos de uso	Adicionadas novas instruções e um exemplo de código do DAG para instalar dependências usando arquivos wheel do Python para ambientes Amazon MWAA com acesso público e privado. • Coomo gerenciar dependências usando wheels do Python	13 de maio de 2022

Novos tópicos e casos de uso	Foram adicionadas novas orientações sobre como escolher para quais métricas do Apache Airflow o Amazon MWAA envia. CloudWatch • Como escolher quais métricas do Apache Airflow relatar	19 de abril de 2022
Novos guias	O Amazon MWAA oferece um guia de migração para migrar fluxos de trabalho do Apache Airflow de implantaç ões autogerenciadas, bem como de ambientes Amazon MWAA existentes. • Guia de migração do Amazon MWAA	7 de março de 2022
Novos tópicos e casos de uso	Adicionadas novas práticas recomendadas de segurança para trabalhar com o Apache Airflow, incluindo uma solução para detectar alterações nos privilégios de usuário do Apache Airflow. • the section called “Práticas recomendadas de segurança no Apache Airflow”	18 de fevereiro de 2022

Novo código de exemplo	Adicionada um novo exemplo de código para criar DAGs com reconhecimento de fuso horário usando o Pendulum e esclarecido como usar um plug-in personalizado para alterar o fuso horário no qual os registros do Apache Airflow são criados. • the section called “Como alterar o fuso horário de um DAG”	11 de fevereiro de 2022

Lançamento do Apache Airflow v2.2.2	O Amazon Managed Workflows para Apache Airflow agora oferece suporte ao Apache Airflow v2.2.2. A partir da versão 2.2, o Amazon MWAA instalará pacotes e plug-ins personalizados do Python diretamente no servidor web Apache Airflow, permitindo maior flexibilidade para gerenciar seus ambientes . Para obter mais informações, consulte. • Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow.. • the section called “Pacotes de provedores para conexões Apache Airflow v2.2.2”. • Log de alterações do Apache Airflow v2.2.2 no site de documentação do Apache Airflow.	27 de janeiro de 2022

Novos tutoriais	Adicionado um novo tutorial que demonstra a criação de um novo perfil personalizado do Apache Airflow e a atribuição do perfil a um usuário do Apache Airflow mapeado do IAM para limitar o acesso do usuário a um subconjunto de DAGs especificados. • the section called “Tutorial : Restringir usuários a um subconjunto de DAGs”	8 de dezembro de 2021

Correções	Corrigida uma recomendação de melhores práticas para definir o valor de `scheduler.min_file_process_interval` para otimizar o uso da CPU. Adicionado um exemplo de política do IAM concedendo acesso aos recursos do Secrets Manager no perfil de execução. Adicionado tópico de solução de problemas sobre o uso das chaves de condição do Secrets Manager. • Ajuste de desempenho: como o agendador analisa os DAGs • Forneça ao Amazon MWAA permissão para acessar as chaves secretas do Secrets Manager • Como configurar chaves de condição no perfil de execução do Amazon MWAA para Secrets Manager	22 de novembro de 2021

Novo código de exemplo	Adicionado o seguinte novo exemplo de código para modificar o fuso horário no qual os DAGs são processados usando um plug-in personalizado e um novo tópico de solução de problemas para invocar o comando CLI do Apache Airflow `dags backfill` de dentro de um operador bash. • the section called “Como alterar o fuso horário de um DAG” • O comando da CLI Backfill usando um operador bash	1º de novembro de 2023
Correções	Corrigiu problemas na amostra de código do operador do Amazon ECS e esclarece u as permissões adicionai s necessárias na função de execução do Amazon MWAA para permitir que o ambiente acesse o grupo de registros de tarefas do Amazon ECS em Logs. CloudWatch • Permissões de operador do Amazon ECS.	26 de outubro de 2021

Novo código de exemplo	Adicionado novo exemplo de código que consulta o banco de dados PostgreSQL do Aurora em busca de informaçõ es relevantes às execuções do DAG e grava os resultados em um arquivo CSV armazenado no Amazon S3. • the section called “Como exportar metadados do ambiente para Amazon S3”.	1.º de outubro de 2021
Correções	Corrigidas as informações sobre como o Amazon MWAA sincroniza automaticamente objetos novos e alterados do seu bucket de destino do Amazon S3 com seus agendadores e operadores. • Como a pasta DAG funciona.	1.º de outubro de 2021
Atual compatibilidade	O Amazon MWAA agora oferece suporte a pacotes de fornecedores adicionais para o Apache Airflow 2.0+. Para saber mais sobre os pacotes com suporte, consulte: • the section called “Pacotes de provedores para conexões Apache Airflow v2.0.2”.	24 de setembro de 2021

Novos comandos e procedimentos	Foram adicionados mais exemplos de orientação e AWS CLI comando para criar um endpoint de gateway Amazon S3 ao usar uma Amazon VPC sem acesso à Internet: • Como criar uma rede Amazon VPC sem acesso à Internet.	24 de setembro de 2021
Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada um novo exemplo de código que usa um operador do Amazon Elastic Container Service em the section called “Usar a `ECSOperator`”. • Adicionados novos tópicos de solução de problemas para problemas na configuração dos plug-ins do Apache Airflow em the section called “Plug-ins”.	19 de setembro de 2021

Nova região compatível	O Amazon MWAA já está disponível nas seguintes regiões: • Ásia-Pacífico (Mumbai): ap-south-1 • Ásia-Pacífico (Seul): ap-northeast-2 • Europa (Londres): eu-west-2 • Europa (Paris): eu-west-3 • Canadá (Central): ca-central-1 • América do Sul (São Paulo): sa-east-1 Para obter mais informações sobre endpoints de serviços e disponibilidade de região, consulte: • Endpoints e cotas do Amazon MWAA em Referência geral da AWS.	31 de agosto de 2021
Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Atualizadas as políticas de amostra para permitir que o Amazon MWAA busque as configurações do Amazon S3 em nível de conta (`s3:GetAccountPublicAccessBlock`) em Perfil de execução do Amazon MWAA.	27 de agosto de 2021

Correções	Adicionadas as seguintes alterações: • Corrigido o AWS CloudFormation modelo para usar uma regra de entrada de autorreferência para o grupo de segurança em. Criar a rede VPC • Corrigido o AWS CloudFormation modelo para usar uma regra de entrada de autorreferência para o grupo de segurança em. Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow	27 de agosto de 2021
Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado decorador DAG à lista do que é compatível com o Apache Airflow v2.0.2. Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow..	20 de agosto de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado caso de uso `celery.sync_parallelism` para Ajuste de desempenho para o Apache Airflow no Amazon MWAA. • Adicionados endpoints de serviço à página de cotas e alterdo o nome para Endpoints e cotas de serviços do Amazon Managed Workflows for Apache Airflow. • Esclarecidos os pré-requisitos de rede com base no feedback do usuário em Comece a usar o Amazon Managed Workflows for Apache Airflow. • Movido `dags list-runs` e `dags next-execution` para comandos CLI do Airflow não compatíveis em Referência de comandos da CLI do Apache Airflow.	13 de agosto de 2021

Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de bash para definir, obter ou excluir uma variável do Apache Airflow v2.0.2 em Referência de comandos da CLI do Apache Airflow. • Adicionadas dependências do Apache Airflow v2.0.2 e o exemplo de conexão do Airflow para Como usar o Amazon MWAA com Amazon RDS para Microsoft SQL Server.	13 de agosto de 2021
Correções	Adicionadas as seguintes alterações: • Corrigido o exemplo de código Python com base no feedback em Como criar uma conexão SSH usando o `SSHOperator` .	13 de agosto de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Movido `variables set` para comandos CLI do Airflow compatíveis em Referência de comandos da CLI do Apache Airflow. • Adicionado o resumo sobre O que mudou na v2.0.2 da página de versões do Airflow para Como instalar dependências do Python com base no feedback do usuário. • Adicionado o resumo sobre O que mudou na v2.0.2 da página de versões do Airflow para Referência de comandos da CLI do Apache Airflow com base no feedback do usuário. • Adicionado o resumo sobre O que mudou na v2.0.2 da página de versões do Airflow para Visão geral dos tipos de conexão com base no feedback do usuário. • Adicionado o resumo sobre O que mudou na v2.0.2 da página de versões do Airflow para Instalando plug-ins personalizados com base no feedback do usuário.	6 de agosto de 2021

	• Adicionado o resumo sobre O que mudou na v2.0.2 da página de versões do Airflow para Como adicionar ou atualizar DAGs com base no feedback do usuário.	
Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código do Apache Airflow v2.0.2 para Como usar um DAG para importar variáveis na CLI. • Adicionado exemplo de código do Apache Airflow v2.0.2 para Como invocar DAGs com uma função do Lambda.	6 de agosto de 2021
Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado tópico de solução de problemas para “Não consigo ver minha conexão na IU do Airflow” em. Solução de problemas para Amazon Managed Workflows para Apache Airflow • Adicionada uma lista de Amazon VPCs às quais o Amazon MWAA oferece suporte para Sobre a rede do Amazon MWAA.	29 de julho de 2021

Correções	Adicionadas as seguintes alterações: • Corrigido o exemplo de código Python com base no feedback do usuário para imprimir o token de login na web em Crie um token de acesso ao servidor web Apache Airflow. • Corrigido o tópico de conexão do Snowflake com base no feedback do usuário para usar uma única cota para o parâmetro do armazém em Solução de problemas para Amazon Managed Workflows para Apache Airflow.	29 de julho de 2021
Tópicos removidos ou movidos	Adicionadas as seguintes alterações: • Reestruturada a página existente para incluir todas as páginas de documenta ção de monitoramento e métricas Monitoramento e métricas para o Amazon Managed Workflows for Apache Airflow. • Movido Métricas do ambiente Apache Airflow v2 em CloudWatch para o menu de navegação de monitoramento e métricas.	23 de julho de 2021

Novos guias	Adicionadas as seguintes alterações: • Criado Pacotes do provedor Apache Airflow instalados em ambientes Amazon MWAA. • Criado Visão geral do monitoramento no Amazon MWAA. • Criado Visualizando registros de auditoria em AWS CloudTrail. • Criado Visualizando registros de fluxo de ar na Amazon CloudWatch.	23 de julho de 2021

Correções	Adicionadas as seguintes alterações: • Corrigido o exemplo de código do Python com base no feedback do usuário para gerar uma string de conexão do Airflow na sequência correta e adicionado o parâmetro de porta em Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager. • Adicionada etapa para instalar um pacote de descompactação localment e com base no feedback do usuário em Como criar um plugin personalizado com a Oracle.	23 de julho de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Tópico adicionado para operadores AWS de DMS emPerguntas frequentes sobre o Amazon MWAA. • Adicionado tópico de solução de problemas para um erro de logs remotos em Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Movido `variables set` para comandos CLI do Airflow não compatíveis em Referência de comandos da CLI do Apache Airflow.	16 de julho de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionadas etapas sequenciais para criar um arquivo requirements.txt com base no feedback do usuário em Como instalar dependências do Python. • Adicionadas etapas sequenciais para criar um arquivo plugins.zip com base no feedback do usuário em Instalando plug-ins personalizados. • Adicionados links de referência cruzada em todo o guia do usuário para o guia de referência da API em Guia de referência da API do Amazon Managed Workflows for Apache Airflow. • Adicionado tópico sobre por que os plug-ins não são exibidos no menu Administrador > Plugins do Airflow 2.0 em Perguntas frequentes sobre o Amazon MWAA.	09 de julho de 2021
Novos guias	Adicionadas as seguintes alterações: • Criado Como excluir arquivos do Amazon S3.	09 de julho de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada uma lista de valores compatíveis em Como usar chaves gerenciadas pelo cliente para criptografia. • Atualizado e esclarecido o exemplo de um URL de repositório privado com base no feedback do usuário em Como gerenciar dependências do Python em requirements.txt.	2 de julho de 2021
Novo código de exemplo	Adicionadas as seguintes alterações: • Foi adicionado o código de amostra do Apache Airflow v1.10.12 para usar uma chave privada AWS Secrets Manager para uma conexão SSH em. Como criar uma conexão SSH usando o `SSHOperator`	2 de julho de 2021
Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado StartedTaskInstances e FinishedTaskInstances métricas paraMétricas do ambiente Apache Airflow v2 em CloudWatch.	25 de junho de 2021

Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código do Apache Airflow v2.0.2 em Como usar o Amazon MWAA com o Amazon EKS.	25 de junho de 2021
Novos guias	Adicionadas as seguintes alterações: • Criado Ajuste de desempenho para o Apache Airflow no Amazon MWAA.	25 de junho de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionados `connections add` e `connections delete` aos comandos CLI compatíveis do Apache Airflow v2.0.2 em Referência de comandos da CLI do Apache Airflow. • Adicionado que a versão mais recente disponível em AWS CloudFormation é o Apache Airflow v2.0.2 em. Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow • Adicionada pergunta sobre armazenamento de dados temporários nos operadores do Apache Airflow para Perguntas frequentes sobre o Amazon MWAA. • Adicionado tópico para o erro “Executor reporta a instância da tarefa %s concluída” em Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionado tópico para o log “Servidor fechou a conexão inesperadamente” para Solução de problemas para Amazon Managed	18 de junho de 2021

Workflows para Apache Airflow.

- Adicionado exemplo para executar comandos CLI em um túnel SSH para um bastion host para Como criar um token da CLI do Apache Airflow.
- Adicionado tópico para nomes de usuário gerados aleatoriamente para Solução de problemas para Amazon Managed Workflows para Apache Airflow.
- Adicionado tópico para um erro 503 ao executar um DAG na CLI para Solução de problemas para Amazon Managed Workflows para Apache Airflow.
- Adicionado tópico para plug-ins personalizados no Apache Airflow v2.0.2, os quais precisam de uma opção de configuração do Airflow de `core.lazy_load_plugins : False` para fazer upload de plug-ins no início de cada processo do Airflow para substituir a configura ção padrão da versão para Como usar opções de configuração do Apache Airflow no Amazon MWAA.

- Adicionada etapa de opções de configuração do Airflow para o exemplo de código dos plug-ins Apache Airflow v2.0.2 em Criação de um plug-in personalizado com o Apache Hive e o Hadoop.
- Adicionada etapa de opções de configuração do Airflow para o exemplo de código dos plug-ins Apache Airflow v2.0.2 em Criar um plug-in personalizado que gera variáveis de ambiente de runtime.
- Adicionada etapa de opções de configuração do Airflow para o exemplo de código dos plug-ins Apache Airflow v2.0.2 em Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow.
- Adicionada etapa de opções de configuração do Airflow para o exemplo de código dos plug-ins Apache Airflow v2.0.2 em Como criar um plugin personalizado com a Oracle.

Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código para uma conexão Snowflake do Apache Airflow em Como usar uma chave secreta em AWS Secrets Manager para uma conexão Snowflake do Apache Airflow.	18 de junho de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada orientação de criptografia no lado do servidor para Criar um bucket do Amazon S3 para o Amazon MWAA. • Adicionado o back-end de segredos do Apache Airflow v2.0.2 para Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager. • Adicionada pergunta para solicitações de aumento de cota de operadores do Apache Airflow para Perguntas frequentes sobre o Amazon MWAA. • Adicionada pergunta sobre quais métricas são usadas para determinar se os operadores do Apache Airflow devem ser escalados para Perguntas frequentes sobre o Amazon MWAA. • Foi adicionada uma pergunta para criar métricas personalizadas em CloudWatch Perguntas frequentes sobre o Amazon MWAA. • Adicionadas etapas para habilitar endereços IP	2 de junho de 2021

	privados para um endpoint de interface VPC do Amazon S3 para uma VPC com roteamento privado em Como criar endpoints de serviço de VPC necessários em um Amazon VPC com roteamento privado. • Adicionada uma opção para configurar um túnel SSH usando o encaminhamento de portas locais em Tutorial: Como configurar o acesso à rede privada usando um Linux Bastion Host.	
Novo código de exemplo	Adicionadas as seguintes alterações: • Foi adicionado um exemplo de código para um DAG que consulta o banco de dados de metadados PostgreSQL do Amazon Aurora e publica métricas personalizadas na Amazon em. CloudWatch Como usar um DAG para gravar métricas personalizadas no CloudWatch	2 de junho de 2021

Novos guias	Adicionadas as seguintes alterações: • Criado um guia sobre como usar modelos de conexão de forma intercambiável na IU do Apache Airflow em Visão geral dos tipos de conexão.	2 de junho de 2021
Correções	Adicionadas as seguintes alterações: • Foram adicionados endpoints VPC Apache Airflow ao AWS CloudFormation modelo na Opção três: criar uma rede VPC sem acesso à Internet a. Criar a rede VPC	2 de junho de 2021

Lançamento do Apache Airflow v2.0.2	Lançamento da disponibilidade geral do Apache Airflow v2.0.2. • Criado Versões do Apache Airflow no Amazon Managed Workflows for Apache Airflow.. • Criado Métricas do ambiente Apache Airflow v2 em CloudWatch. • Adicionados links específicos da versão do Apache Airflow v2.0.2 para Como usar opções de configuração do Apache Airflow no Amazon MWAA. • Adicionada orientação específica da versão do Apache Airflow v2.0.2 para Como instalar dependências do Python. • Adicionada orientação específica da versão do Apache Airflow v2.0.2 para Como gerenciar dependências do Python em requirements.txt. • Adicionados exemplo de plug-ins do Apache Airflow v2.0.2 para Instalando plug-ins personalizados • Adicionado exemplo de código do Apache Airflow v2.0.2 para Limpeza	26 de maio de 2021

do banco de dados do Aurora PostgreSQL em um ambiente do Amazon MWAA.

- Adicionado exemplo de código do Apache Airflow v2.0.2 para Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow.
- Adicionado exemplo de código do Apache Airflow v2.0.2 para Como criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow.
- Adicionados os comandos do Apache Airflow v2.0.2 para Referência de comandos da CLI do Apache Airflow.
- Adicionados scrits do Apache Airflow v2.0.2 para Como criar um token da CLI do Apache Airflow.
- Adicionada uma observaçã o de que o Amazon MWAA usa a versão mais recente do Apache Airflow por padrão para Criar um ambiente do Amazon MWAA.

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada orientação para solução de problemas de tarefas do Airflow que estão travadas ou não estão funcionando para Solução de problemas para Amazon Managed Workflows para Apache Airflow.	14 de maio de 2021
Correções	Adicionadas as seguintes alterações: • Atualizamos o exemplo de código de plug-ins para usar a versão mais recente do Java em Criação de um plug-in personalizado com o Apache Hive e o Hadoop. Anteriormente, era `os.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"` .	12 de maio de 2021
Tópicos removidos ou movidos	Adicionadas as seguintes alterações: • Movidos os tópicos em Solução de problemas para Amazon Managed Workflows para Apache Airflow para novas páginas por categoria.	10 de maio de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada visão geral do bucket do Amazon S3 ao. Como trabalhar com DAGs no Amazon MWAA	10 de maio de 2021
Tópicos removidos ou movidos	Adicionadas as seguintes alterações: • Movido Acessando o Apache Airflow para a navegação de nível superior e adicionou páginas para Crie um token de acesso ao servidor web Apache Airflow, Como criar um token da CLI do Apache Airflow e Referência de comandos da CLI do Apache Airflow.	7 de maio de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionados links específicos da versão ao Guia de referência do Apache Airflow para todos os comandos da CLI do Airflow compatíveis e não suportados em Referência de comandos da CLI do Apache Airflow. • Adicionados links específicos da versão ao Guia de referência do Apache Airflow para todas as opções de configuração em Como usar opções de configuração do Apache Airflow no Amazon MWAA. • Adicionado o utilitário Amazon MAA CLI para Como gerenciar dependências do Python em requirements.txt.	7 de maio de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionados exemplos simples e aninhados de como estruturar um plugins.zip em Instalando plug-ins personalizados. • Adicionado utilitário Amazon MAA CLI às páginas Como adicionar ou atualizar DAGs, Instalando plug-ins personalizados e Como instalar dependências do Python. • Reestruturado o conteúdo em uma visão geral, feito o upload para o Amazon S3 e instalação das seções do Amazon MWAA com base nos comentários dos usuários nas páginas Instalando plug-ins personalizados e Como instalar dependências do Python. • Adicionado um exemplo de caso de uso para criar e conectar os endpoints da VPC necessários para um Amazon VPC existente sem acesso à Internet em Sobre a rede do Amazon MWAA.	30 de abril de 2021

Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código que usa uma chave secreta no Secrets Manager para uma variável com o Apache Airflow em Como usar uma chave secreta em AWS Secrets Manager para uma variável do Apache Airflow.	30 de abril de 2021
Novos guias	Adicionadas as seguintes alterações: • Criado Como criar endpoints de serviço de VPC necessários em um Amazon VPC com roteamento privado.	30 de abril de 2021
Correções	Adicionadas as seguintes alterações: • Opa! Nós atualizamos `core.default_ui_timezone` para `webserver.default_ui_timezone` em Como usar opções de configuração do Apache Airflow no Amazon MWAA.	30 de abril de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionadas etapas do Windows (PuTTY) para o túnel SSH para Tutorial: Como configurar o acesso à rede privada usando um Linux Bastion Host. • Adicionado tópico para `apache-airflow-providers-amazon`, que é compatível somente com o Apache Airflow 2.0 para Solução de problemas para Amazon Managed Workflows para Apache Airflow.	23 de abril de 2021
Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código que usa uma chave secreta no Secrets Manager para uma conexão com o Apache Airflow em Como usar uma chave secreta em AWS Secrets Manager para uma conexão do Apache Airflow.	23 de abril de 2021

Novos guias	Adicionadas as seguintes alterações: • Criado Sobre a rede do Amazon MWAA. • Criado Segurança em sua VPC no Amazon MWAA. • Criado Gerenciando o acesso a endpoints Amazon VPC específicos de serviços no Amazon MWAA.	23 de abril de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Foi adicionado um novo AWS CloudFormation modelo para criar uma rede Amazon VPC sem acesso à Internet. Criar a rede VPC • Foi adicionado um novo tutorial para criar uma AWS Client VPN entradaTutorial: Como configurar o acesso à rede privada usando um AWS Client VPN. • Alterado o nome da página Acesso à rede para Modos de acesso do Apache Airflow com base no feedback do usuário e simplificou a documentação em Modos de acesso do Apache Airflow. • Documentos simplific ados para incluir apenas informações e modelos de introdução do Amazon VPC com base no feedback do usuário em Criar a rede VPC. • Solução alternativa BigQuery do operador adicionada ao. Solução de problemas para Amazon Managed Workflows para Apache Airflow	16 de abril de 2021

	• Adicionada uma prática recomendada de arquivo de restrições do Apache Airflow v1.10.12 para Como instalar dependências do Python.	
Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado exemplo de código para criar um plug-in personalizado usando o Oracle em Como criar um plugin personalizado com a Oracle. • Adicionado exemplo de código para criar um plug-in personalizado que gera variáveis de ambiente de runtime em Criar um plug-in personalizado que gera variáveis de ambiente de runtime. •	16 de abril de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado tópico para o requisito de regra de autorreferência em um grupo de segurança de VPC para Perguntas frequentes sobre o Amazon MWAA. • Adicionado diretório de plug-ins personalizados e limites de tamanho para Instalando plug-ins personalizados. • Adicionados diretório de requisitos e limites de tamanho para Como instalar dependências do Python. • Esclarecidas as opções de configuração do Apache Airflow para `foo.user` e `foo.pass` em Como gerenciar dependências do Python em requirements.txt. • Adicionada visão geral das opções de configuração para Como usar opções de configuração do Apache Airflow no Amazon MWAA.	9 de abril de 2021

Novo código de exemplo	Adicionadas as seguintes alterações: • Código de amostra adicionado para criar um plug-in personalizado usando PythonVirtualenvOperator inComo criar um plug-in personalizado para PythonVirtualEnvOperator do Apache Airflow. • Adicionado exemplo de código para criar um plug-in personalizado com o Apache Hive e o Hadoop em Criação de um plug-in personalizado com o Apache Hive e o Hadoop.	9 de abril de 2021
Correções	Adicionadas as seguintes alterações: • Opa! Atualizamos o formato de um requirements.txt e adicionamos um exemplo compatível com o Apache Airflow v1.10.12 in. Como instalar dependências do Python	31 de março de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada solução alternativa para remover um requirements.txt ou plugins.zip para Perguntas frequentes sobre o Amazon MWAA • Adicionada uma solução alternativa de bash para SSH em um ambiente para Perguntas frequentes sobre o Amazon MWAA. • Tópico adicionado para CloudTrail ResourceA lreadyExistsException erro emSolução de problemas para Amazon Managed Workflows para Apache Airflow.	26 de março de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Foi adicionada uma lista de AWS serviços usados paraPerfil de execução do Amazon MWAA. • Foi adicionada uma lista de AWS serviços usados paraPerfil vinculado a serviços para o Amazon MWAA. • Adicionada pergunta para a versão Python 3.7 para Amazon MWAA para Perguntas frequentes sobre o Amazon MWAA. • Pergunta adicionada PythonVirtualenvOperator para Perguntas frequentes sobre o Amazon MWAA o. • Adicionado o script de solução de problemas como próximas etapas para todos os tópicos relacionados à VPC e à configuração do ambiente em Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Esclarecida documentação de que um linux bastion deve estar na mesma região que um ambiente em Tutorial: Como configura	19 de março de 2021

	r o acesso à rede privada usando um Linux Bastion Host.	
Novos guias	Adicionadas as seguintes alterações: • Guia de conexões do Apache Airflow criado para AWS Secrets Manager at. Configurando uma conexão Apache Airflow usando um segredo AWS Secrets Manager • Criei um tutorial de início rápido usando um AWS CloudFormation modelo para criar a infraestrutura Amazon VPC, o bucket Amazon S3 e o ambiente Amazon MWAA em. Tutoriais de início rápido para Amazon Managed Workflows for Apache Airflow	19 de março de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado o tópico de solução de problemas de criação de bucket do Amazon S3 para Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionadas etapas para criar e anexar uma política JSON para Perfil de execução do Amazon MWAA.	12 de março de 2021
Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionado código de exemplo para adicionar uma configuração ao acionar um DAG para Acessando o Apache Airflow.	12 de março de 2021
Novos guias	Adicionadas as seguintes alterações: • Criado guia de práticas recomendadas em Como gerenciar dependências do Python em requirements.txt.	12 de março de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Foi adicionado o tópico Google/GCP/solução BigQuery de problemas ao. Solução de problemas para Amazon Managed Workflows para Apache Airflow • Adicionado tópico de solução de problemas do Cython para Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionado tópico de solução de problemas do MySQL para Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionado tópico de solução de problemas de erros do servidor web 5xx para Solução de problemas para Amazon Managed Workflows para Apache Airflow.	5 de março de 2021

Atual compatibilidade	Adicionadas as seguintes alterações: • Anteriormente, não `backend_kwargs` era suportado AWS Secrets Manager e você precisava de uma solução alternativa para substituir a chamada da função Secrets Manager. Agora, `backend_kwargs` é compatível. Consulte o tópico AWS Secrets Manager de solução de problemas em Solução de problemas para Amazon Managed Workflows para Apache Airflow.	4 de março de 2021
Correções	Adicionadas as seguintes alterações: • Opa! Atualizamos o tamanho de cada classe de ambiente para refletir o GB real em Como configurar a classe de ambiente do Amazon MWAA.	4 de março de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado acesso à rede privada usando uma política de endpoint da VPC para Modos de acesso do Apache Airflow. • Adicionadas verificações adicionais para o tópico de criação de um ambiente para Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionadas etapas para visualizar os logs para `requirements.txt` para Como instalar dependências do Python.	26 de fevereiro de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado o caso de uso do Apache Hive para Como instalar dependências do Python. • Esclarecida a documentação de que as dependências necessárias para um pacote Apache Airflow precisam ser incluídas no arquivo `requirements.txt` em Como instalar dependências do Python. • Adicionado tópico de solução de problemas Atualização de requisitos.txt para Solução de problemas para Amazon Managed Workflows para Apache Airflow.	25 de fevereiro de 2021
Novos tutoriais	Adicionadas as seguintes alterações: • Adicionado tutorial de rede privada para Tutorial: Como configurar o acesso à rede privada usando um Linux Bastion Host.	22 de fevereiro de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionadas configurações de rede pública e privada para Modos de acesso do Apache Airflow. • Adicionados casos de uso e cenários de usuário do grupo de desenvolvimento para Perfil de execução do Amazon MWAA.	22 de fevereiro de 2021
Novo código de exemplo	Adicionadas as seguintes alterações: • Adicionados exemplos de scripts Python para token de login na web e token CLI para Acessando o Apache Airflow. • Adicionado um exemplo de código para acionar o DAG em outro ambiente para Exemplos de código para o Amazon Managed Workflows for Apache Airflow. • Adicionado um exemplo de código para acionar o DAG usando uma função do Lambda para Como invocar DAGs com uma função do Lambda.	22 de fevereiro de 2021

Novos comandos e procedimentos	Adicionadas as seguintes alterações: • Adicionados procedimentos passo a passo a todos os scripts em Acessando o Apache Airflow.	22 de fevereiro de 2021
Novo código de exemplo	Adicionadas as seguintes alterações: • Exemplo de cURL atualizado para token de login na web em Acessando o Apache Airflow. • Adicionado o código de amostra para conectar-se a um Microsoft SQL Server do Amazon RDS para Como usar o Amazon MWAA com Amazon RDS para Microsoft SQL Server.	17 de fevereiro de 2021

Novos comandos e procedimentos	Adicionadas as seguintes alterações: • AWS CLI Comandos adicionados às Como trabalhar com DAGs no Amazon MWAA páginas. • O Apache Airflow não oferece suporte a DAGs serializados nos comandos da CLI. Como a CLI é executada no servidor web, que não tem plug-ins ou requisitos por motivos de segurança, qualquer ambiente MWAA com plugins.zip ou requirements.txt não suportará esses comandos. Movido os comandos `list_dags` e `backfill` do Apache Airflow para comandos não compatíveis em Acessando o Apache Airflow.	17 de fevereiro de 2021
GitHub lançar	Os documentos do guia do usuário agora são de código aberto em. GitHub Escolha “Editar esta página em GitHub” em qualquer página.	17 de fevereiro de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionada pergunta para o caso de uso de perfis de etapa v. Amazon MWAA para Solução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionada política de acesso da CLI para Como acessar um ambiente do Amazon MWAA. • Esclarecida a documenta ção de que qualquer opção de configuração compatível do Apache Airflow pode ser especificada em Como usar opções de configuração do Apache Airflow no Amazon MWAA. • Esclarecida a documenta ção de que, se um contêiner Fargate em uma zona de disponibilidade falhar, o MWAA mudará para outro contêiner em uma zona de disponibilidade diferente em Criar a rede VPC.	12 de fevereiro de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adição do Como configurar a classe de ambiente do Amazon MWAA.	5 de fevereiro de 2021
Tópicos removidos ou movidos	Adicionadas as seguintes alterações: • Removido o requisito para que o nome do bucket do Amazon S3 começar com `airflow-` em Comece a usar o Amazon Managed Workflows for Apache Airflow. • Movido Como acessar um ambiente do Amazon MWAA e Perfil de execução do Amazon MWAA para Gerenciamento de acesso a um ambiente do Amazon MWAA.	4 de fevereiro de 2021

Amazon MAA CloudFormation	Atualize os parâmetros para criar um ambiente no Amazon MWAA CloudFormation. • Remover SubnetList. • Remover TagList. • Adicionar NetworkConfiguration. • Adicionar TagMap. • Adicionar exemplos de solicitação de criação de ambiente.	4 de fevereiro de 2021

Novos tópicos e casos de uso	Adicionadas as seguintes alterações: • Adicionado exemplo de configuração de e-mail para Como usar opções de configuração do Apache Airflow no Amazon MWAA. • Tópico PostgresHook de solução de problemas adicionado aoSolução de problemas para Amazon Managed Workflows para Apache Airflow. • Tópico AWS Secrets Manager de solução de problemas adicionado aoSolução de problemas para Amazon Managed Workflows para Apache Airflow. • Adicionado caso de uso de alto desempenho para Configurando a escalabilidade automática do funcionário do Amazon MWAA.	29 de janeiro de 2021

Lançamento do Amazon MWAA	Lançamento da disponibilidade geral do Amazon Managed Workflows for Apache Airflow. • Documentação do guia do usuário • AWS CloudFormation documentação	24 de novembro de 2020

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.