aws

Guía del usuario

Amazon Managed Workflows para Apache Airflow

Amazon Managed Workflows para Apache Airflow: Guía del usuario

Table of Contents

¿Qué es Amazon Managed Workflows para Apache Airflow?

Use Amazon Managed Workflows para Apache Airflow, un servicio administrado para Apache Airflow, para configurar y operar canalizaciones de datos en la nube a escala. Apache Airflow es una herramienta de código abierto que se usa para crear, programar y supervisar flujos de trabajo.

Con Amazon MWAA, puede usar Apache Airflow y Python para crear flujos de trabajo sin tener que administrar la infraestructura subyacente para conseguir escalabilidad, disponibilidad y seguridad. Amazon MWAA se escala automáticamente para adaptarse a sus necesidades de flujo de trabajo. También se integra con los servicios de seguridad de AWS para permitir un acceso rápido y seguro a los datos.

Contenidos

Características

Consulte las siguientes características para obtener información sobre cómo Amazon MWAA puede simplificar la administración de sus flujos de trabajo de Apache Airflow.

- Configuración automática de Airflow: configure rápidamente Apache Airflow eligiendo una versión de Apache Airflow al crear un entorno de Amazon MWAA. Amazon MWAA configura Apache Airflow mediante la misma interfaz de usuario de Apache Airflow y el mismo código abierto que puede descargar de Internet.
- Escalado automático: escale automáticamente los procesos de trabajo de Apache Airflow (los recursos informáticos que ejecutan sus tareas) fijando límites mínimos y máximos. Amazon MWAA supervisa los procesos de trabajo de su entorno y usa su componente de escalado automático para añadir procesos de trabajo con el objetivo de satisfacer la demanda, hasta alcanzar el número máximo de procesos de trabajo que se haya definido.
- Autenticación integrada: habilite la autenticación y la autorización basadas en roles para el servidor web de Apache Airflow definiendo las políticas de control de acceso en AWS Identity and Access

Management (IAM). Los procesos de trabajo de Apache Airflow asumen estas políticas para brindar un acceso seguro a los servicios de AWS.

- Seguridad integrada: los programadores y procesos de trabajo de Apache Airflow se ejecutan en Amazon VPC de Amazon MWAA. Los datos también se cifran automáticamente mediante AWS Key Management Service, por lo que su entorno es seguro de forma predeterminada.
- Modos de acceso público o privado: acceda a su servidor web de Apache Airflow mediante un modo de acceso público o privado. El modo de acceso a la red pública usa un punto de conexión de VPC para el servidor web de Apache Airflow al que se puede acceder a través de Internet. El modo de acceso a la red privada usa un punto de conexión de VPC para el servidor web de Apache Airflow al que se puede acceder a través de su VPC. En ambos casos, el acceso de los usuarios a Apache Airflow se controla mediante la política de control de acceso que defina en AWS Identity and Access Management (IAM) y en el SSO de AWS.
- Actualizaciones y revisiones simplificadas: Amazon MWAA proporciona nuevas versiones de Apache Airflow periódicamente. El equipo de Amazon MWAA actualizará y revisará las imágenes de estas versiones.
- Supervisión del flujo de trabajo: consulte los registros de Apache Airflow y las métricas de Apache Airflow de Amazon CloudWatch para identificar los retrasos en las tareas de Apache Airflow o los errores de flujo de trabajo sin tener que usar otras herramientas externas. Amazon MWAA envía automáticamente las métricas del entorno y, si están habilitados, los registros de Apache Airflow a CloudWatch.
- Integración de AWS: Amazon MWAA admite integraciones de código abierto con Amazon Athena, AWS Batch, Amazon CloudWatch, Amazon DynamoDB, AWS DataSync, Amazon EMR, AWS Fargate, Amazon EKS, Amazon Data Firehose, AWS Glue, AWS Lambda, Amazon Redshift, Amazon SQS, Amazon SNS, Amazon SageMaker AI y Amazon S3, así como cientos de operadores y sensores integrados y creados por la comunidad.
- Flotas de procesos de trabajo: Amazon MWAA ofrece soporte para el uso de contenedores para ampliar la flota de procesos de trabajo bajo demanda y reducir la caída de programadores mediante Amazon ECS en AWS Fargate. Se admiten operadores que invoquen tareas en los contenedores de Amazon ECS y operadores de Kubernetes que creen y ejecuten pods en un clúster de Kubernetes.

Arquitectura

Todos los componentes incluidos en el cuadro exterior (en la imagen siguiente) aparecen como un único entorno de Amazon MWAA en su cuenta. El programador y los procesos de trabajo de Apache

Airflow son contenedores de AWS Fargate que se conectan a las subredes privadas de la Amazon VPC de su entorno. Cada entorno tiene su propia base de metadatos de Apache Airflow administrada por AWS, a la que pueden acceder los contenedores de Fargate de programador y procesos de trabajo a través de un punto de conexión de VPC protegido de forma privada.

Amazon CloudWatch, Amazon S3, Amazon SQS y AWS KMS son independientes de Amazon MWAA y se debe poder acceder a ellos desde los programadores y procesos de trabajo de Apache Airflow en los contenedores de Fargate. Hay disponibles varios programadores de Apache Airflow solo en Apache Airflow v2 y versiones posteriores. Para más información sobre el ciclo de vida de las tareas de Apache Airflow en Conceptos, consulte la guía de referencia de Apache Airflow.

Se puede acceder al servidor web de Apache Airflow a través de Internet seleccionando el modo de acceso red pública de Apache Airflow o desde dentro de su VPC seleccionando el modo de acceso red privada de Apache Airflow. En ambos casos, el acceso de los usuarios a Apache Airflow se controla mediante la política de control de acceso que defina en AWS Identity and Access Management (IAM).

> **Note**
>
> A partir de Apache Airflow v3, el servidor web Amazon MWAA también aloja el servidor API de ejecución de Apache Airflow.

Integración

La activa y creciente comunidad de código abierto de Apache Airflow proporciona operadores (complementos que simplifican las conexiones a los servicios) para que Apache Airflow se integre con los servicios de AWS. Esto incluye servicios como Amazon S3, Amazon Redshift, Amazon EMR, AWS Batch y Amazon SageMaker AI, así como servicios en otras plataformas en la nube.

El uso de Apache Airflow con Amazon MWAA es totalmente compatible con servicios de AWS y herramientas populares de terceros, como Apache Hadoop, Presto, Hive y Spark, para llevar a cabo tareas de procesamiento de datos. Amazon MWAA se compromete a mantener la compatibilidad con la API de Apache Airflow y pretende proporcionar integraciones fiables a los servicios de AWS y ponerlos a disposición de la comunidad. Además, se compromete a participar en el desarrollo de características para la comunidad.

Para ver un ejemplo de código, consulte Códigos de ejemplo de Amazon Managed Workflows para Apache Airflow.

Versiones compatibles

Amazon MWAA admite varias versiones de Apache Airflow. Para obtener más información sobre las versiones de Apache Airflow que admitimos y los componentes de Apache Airflow incluidos en cada versión, consulte Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow.

Siguientes pasos

- Introducción al uso de una plantilla única de CloudFormation que cree un bucket de Amazon S3 para los DAG y archivos auxiliares de Airflow, una Amazon VPC con enrutamiento público y un entorno de Amazon MWAA en Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow.
- Introducción a la creación gradual de un bucket de Amazon S3 para sus DAG y archivos auxiliares de Airflow, eligiendo una de las tres opciones de red de Amazon VPC, y de un entorno de Amazon MWAA en Introducción a Amazon Managed Workflows para Apache Airflow.

Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow

En este tutorial de inicio rápido, se usa una plantilla de AWS CloudFormation que crea la infraestructura de Amazon VPC, un bucket de Amazon S3 con una carpeta `dags` y un entorno de Amazon Managed Workflows para Apache Airflow al mismo tiempo.

Temas

En este tutorial:

En este tutorial, se explican tres comandos de la AWS Command Line Interface (AWS CLI) para cargar un DAG en Amazon S3, ejecutarlo en Apache Airflow y ver los registros en CloudWatch. Por último, aprenderá a crear una política de IAM para un equipo de desarrollo de Apache Airflow.

> **Note**
>
> La plantilla de CloudFormation de esta página crea un entorno de Amazon Managed Workflows para Apache Airflow para la última versión de Apache Airflow disponible en CloudFormation. La última versión disponible es Apache Airflow v3.0.6.

La plantilla de CloudFormation crea los siguientes recursos:

- Infraestructura de VPC. La plantilla utiliza Enrutamiento público a través de Internet. Usa el Modo de acceso mediante red pública para el servidor web Apache Airflow en `WebserverAccessMode: PUBLIC_ONLY`.

- Bucket de Amazon S. La plantilla crea un bucket de Amazon S3 con una carpeta `dags`. Está configurada para bloquear todo el acceso público, con el control de versiones de buckets activado, tal y como se define en Creación de un bucket de Amazon S3 para Amazon MWAA.
- Entorno de Amazon MWAA. La plantilla crea un entorno de Amazon MWAA asociado a la carpeta `dags` del bucket de Amazon S3, un rol de ejecución con permiso para los servicios de AWS usados por Amazon MWAA y el cifrado predeterminado mediante una clave de AWS propia, tal y como se define en Creación de entornos de Amazon MWAA.
- Registros de CloudWatch. La plantilla permite registros de Apache Airflow en CloudWatch en el nivel INFO y superior para el grupo de registros del programador de Airflow, el grupo de registros del servidor web de Airflow, el grupo de registros del proceso de trabajo de Airflow, el grupo de registros de procesamiento del DAG de Airflow y el grupo de registros de tareas de Airflow, tal como se define en Visualización de registros en Amazon CloudWatch.

Este tutorial guía por los siguientes pasos:

- Carga y ejecutar un DAG. Cargue el DAG tutorial de Apache Airflow de la última versión de Apache Airflow compatible con Amazon MWAA en Amazon S3 y, a continuación, ejecútelo en la interfaz de usuario de Apache Airflow, tal y como se define en Cómo añadir o actualizar DAG.
- Registros de acceso. Vea el grupo de registros del servidor web de Airflow en CloudWatch Logs, tal y como se define en Visualización de registros en Amazon CloudWatch.
- Crear una política de control de acceso. Cree una política de control de acceso en IAM para su equipo de desarrollo de Apache Airflow, tal como se define en Acceso a un entorno de Amazon MWAA.

> **Note**
>
> En la VPC que aloja el entorno de Amazon MWAA, establezca `assignIpv6AddressOnCreation` en `true` para todas las subredes conectadas. Esta configuración garantiza la asignación automática de direcciones del protocolo de Internet versión 6 (IPv6) a los recursos de estas subredes.

Requisitos previos

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Paso uno: guarde la plantilla de CloudFormation localmente

- Copie el contenido de la siguiente plantilla y guárdelo localmente como `mwaa-public-network.yml`. También puede descargar la plantilla.

```
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

  MwaaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17&TCX5-2025-waiver;
        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&TCX5-2025-waiver;
        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}"
```

Paso dos: cree la pila con la AWS CLI

1. En el símbolo del sistema, vaya hasta el directorio en el que está almacenado `mwaa-public-network.yml`. Por ejemplo:

```
cd mwaaproject
```

2. Utilice el comando `aws cloudformation create-stack` para crear la pila con la AWS CLI.

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

> **Note**
>
> Toma más de 30 minutos crear la infraestructura de Amazon VPC, el bucket de Amazon S3 y el entorno de Amazon MWAA.

Paso tres: cargue un DAG en Amazon S3 y ejecútelo en la interfaz de usuario de Apache Airflow

1. Copie el contenido del archivo `tutorial.py` de la última versión compatible de Apache Airflow y guárdelo localmente como `tutorial.py`.
2. En el símbolo del sistema, vaya hasta el directorio en el que está almacenado `tutorial.py`. Por ejemplo:

```
cd mwaaproject
```

3. Use el siguiente comando para obtener una lista de todos los buckets de Amazon S3.

```
aws s3 ls
```

4. Utilice el comando siguiente para enumerar los archivos y las carpetas del bucket de Amazon S3 para su entorno.

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

5. Utilice el siguiente script para cargar el archivo `tutorial.py` en su carpeta `dags`. Reemplace el valor de muestra en *amzn-s3-demo-bucket*.

```
aws s3 cp tutorial.py s3://amzn-s3-demo-bucket/dags/
```

6. Abra la página Entornos en la consola de Amazon MWAA.
7. Seleccione un entorno.
8. Elija Abrir interfaz de usuario de Airflow.
9. En la interfaz de usuario de Apache Airflow, de la lista de DAG disponibles, elija el DAG tutorial.
10. En la página de detalles del DAG, seleccione la opción Pausar/Reanudar DAG que aparece junto al nombre del DAG para reanudar el DAG.
11. Elija Desencadenar DAG.

Paso cuatro: vea los registros en CloudWatch Logs

Puede ver los registros de Apache Airflow en la consola de CloudWatch para todos los registros de Apache Airflow que estaban habilitados por la pila CloudFormation. En la siguiente sección, se muestra cómo ver los registros del grupo de registros del servidor web de Airflow.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija el grupo de registro del servidor web de Airflow en el panel de monitoreo.
4. Seleccione el registro `webserver_console_ip` en los flujos de registro.

Siguientes pasos

- Obtenga más información sobre cómo cargar los DAG, especificar las dependencias de Python en `requirements.txt` y personalizar plugins en un `plugins.zip` en Trabajo con DAG en Amazon MWAA.
- Obtenga más información sobre las mejores prácticas que recomendamos para ajustar el rendimiento de su entorno Ajuste del desempeño de Apache Airflow en Amazon MWAA.
- Cree un panel de supervisión para su entorno en Monitorización de paneles y alarmas en Amazon MWAA.
- Ejecute algunos de los ejemplos de código de DAG en Códigos de ejemplo de Amazon Managed Workflows para Apache Airflow.

Introducción a Amazon Managed Workflows para Apache Airflow

Amazon Managed Workflows para Apache Airflow usa la VPC de Amazon, archivos de DAG y los archivos auxiliares de su bucket de almacenamiento de Amazon S3 para crear un entorno. En esta guía, se describen los requisitos previos y los recursos de AWS necesarios para empezar a usar Amazon MWAA.

Temas

Requisitos previos

Para crear un entorno de Amazon MWAA, procure tener permiso para acceder a los recursos de AWS que necesita para crearlo.

- Cuenta de AWS: una Cuenta de AWS con permiso para usar Amazon MWAA y los servicios y recursos de AWS que use su entorno.

Acerca de esta guía

En esta sección, se describen la infraestructura y los recursos de AWS que creará en esta guía.

- Amazon VPC: componentes de red de Amazon VPC que necesita un entorno de Amazon MWAA. Puede configurar una VPC existente que cumpla estos requisitos (avanzados), como se indica en Acerca de las redes en Amazon MWAA, o crear la VPC y los componentes de red, como se define en the section called “Creación de la red de VPC”.

- Bucket de Amazon S3: bucket de Amazon S3 para almacenar sus DAG y los archivos asociados, como `plugins.zip` y `requirements.txt`. Su bucket de Amazon S3 debe estar configurado para bloquear todo el acceso público, con el control de versiones del bucket activado, tal y como se define en Creación de un bucket de Amazon S3 para Amazon MWAA.
- Entorno de Amazon MWAA: entorno de Amazon MWAA configurado con la ubicación de su bucket de Amazon S3, la ruta al código DAG y todos los complementos personalizados o dependencias de Python, y su Amazon VPC y su grupo de seguridad, tal y como se define en Creación de entornos de Amazon MWAA.

Antes de empezar

Para crear un entorno de Amazon MWAA, puede crear y configurar otros recursos de AWS antes de crear el entorno.

Para crear un entorno, necesitará lo siguiente:

- Clave AWS KMS: clave AWS KMS para el cifrado de datos en su entorno. Puede elegir la opción predeterminada en la consola de Amazon MWAA para crear una clave de AWS propia al crear un entorno o especificar una clave existente administrada por el cliente con permisos configurados para otros servicios de AWS usados en su entorno (avanzados). Consulte Uso de claves maestras de cliente para el cifrado para obtener más información.
- Rol de ejecución: rol de ejecución que permite a Amazon MWAA acceder a los recursos de AWS de su entorno. Puede elegir la opción predeterminada en la consola de Amazon MWAA para crear un rol de ejecución al crear un entorno. Consulte Rol de ejecución de Amazon MWAA para obtener más información.
- Grupo de seguridad de VPC: grupo de seguridad de VPC que permite a Amazon MWAA acceder a otros recursos de AWS de la red de VPC. Puede elegir la opción predeterminada de la consola de Amazon MWAA para crear un grupo de seguridad al crear un entorno o proporcionar a un grupo de seguridad las reglas de entrada y salida adecuadas (avanzadas). Consulte Seguridad en la VPC en Amazon MWAA para obtener más información.

Regiones disponibles

Amazon MWAA ya está disponible en las siguientes Regiones de AWS. Para obtener más información sobre cada región, por ejemplo, cuáles están habilitadas o deshabilitadas de forma predeterminada, consulte Regiones de AWS.

Code	Nombre
us-east-1	Este de EE. UU. (Norte de Virginia)
us-east-2	Este de EE. UU. (Ohio)
us-west-1	Oeste de EE. UU. (Norte de California)
us-west-2	Oeste de EE. UU. (Oregón)
af-south-1	África (Ciudad del Cabo)
ap-east-1	Asia-Pacífico (Hong Kong)
ap-south-2	Asia-Pacífico (Hyderabad)
ap-southeast-3	Asia-Pacífico (Yakarta)
ap-southeast-5	Asia-Pacífico (Malasia)
ap-southeast-4	Asia-Pacífico (Melbourne)
ap-south-1	Asia-Pacífico (Mumbai)
ap-northeast-3	Asia-Pacífico (Osaka)
ap-northeast-2	Asia-Pacífico (Seúl)
ap-southeast-1	Asia-Pacífico (Singapur)
ap-southeast-2	Asia-Pacífico (Sídney)
ap-northeast-1	Asia-Pacífico (Tokio)
ca-central-1	Canadá (centro)
ca-west-1	Oeste de Canadá (Calgary)
eu-central-1	Europa (Fráncfort)
eu-west-1	Europa (Irlanda)

Code	Nombre
eu-west-2	Europa (Londres)
eu-south-1	Europa (Milán)
eu-west-3	Europa (París)
eu-south-2	Europa (España)
eu-north-1	Europa (Estocolmo)
eu-central-2	Europa (Zúrich)
il-central-1	Israel (Tel Aviv)
me-south-1	Medio Oriente (Baréin)
me-central-1	Medio Oriente (EAU)
sa-east-1	América del Sur (São Paulo)

Creación de un bucket de Amazon S3 para Amazon MWAA

Esta guía describe los pasos para crear un bucket de Amazon S3 para almacenar los gráficos acíclicos dirigidos (DAG) de Apache Airflow, los complementos personalizados en un archivo `plugins.zip` y las dependencias de Python en un archivo `requirements.txt`.

Contenido

Antes de empezar

- No puede cambiar el nombre de un bucket de Amazon S3 después de crearlo. Para obtener más información, consulte Reglas para nombrar buckets en la guía del usuario de Amazon Simple Storage Service.

- De debe configurar un bucket de Amazon S3 para un entorno de Amazon MWAA para Bloquear todo el acceso público, con Control de versiones de buckets activado.
- Los buckets de Amazon S3 usados para un entorno de Amazon MWAA deben ubicarse en la misma Región de AWS que los entornos de Amazon MWAA. Para ver una lista de las Regiones de AWS de Amazon MWAA, consulte los puntos de conexión y las cuotas de Amazon MWAA en Referencia general de AWS.

Creación de buckets

En esta sección se describen los pasos para crear el bucket de Amazon S3 para su entorno.

Creación de un bucket

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. Elija Crear bucket.
3. En Nombre del bucket, escriba un nombre compatible con DNS para el bucket.

   El nombre del bucket debe:

   - Ser único en todo Amazon S3.
   - Tener entre 3 y 63 caracteres.
   - No contiene caracteres en mayúsculas.
   - Comenzar por una letra minúscula o un número.

   > **Important**
   >
   > Evite incluir información confidencial, como números de cuenta, en el nombre del bucket. El nombre del bucket será visible en las URL que señalan a los objetos almacenados en él.

4. Elija una Región de AWS en Region (Región). Debe ser la misma Región de AWS que su entorno de Amazon MWAA.

   - Recomendamos que se elija una región cercana para minimizar la latencia y los costos, así como para satisfacer los requisitos reglamentarios.

5. Elija Bloquear todo el acceso público.
6. Seleccione Enable (Habilitar) en Bucket versioning (Control de versiones de buckets).
7. Opcional: etiquetas. Añada pares de etiquetas clave-valor para identificar su bucket de Amazon S3 en Etiquetas. Por ejemplo, `Bucket: Staging`.
8. Opcional: cifrado del lado del servidor. Si lo desea, puede habilitar una de las siguientes opciones de cifrado en su bucket de Amazon S3.

    a. Elija Amazon S3 (SSE-S3) en el cifrado del servidor para activar el cifrado del bucket en el servidor.

    b. Elija la clave AWS Key Management Service (SSE-KMS) para usar una clave de AWS KMS para el cifrado en su bucket de Amazon S3:

        i. Clave administrada de AWS (aws/s3): si elige esta opción, podrá usar una clave propia de AWS administrada por Amazon MWAA o especificar una clave administrada por el cliente para el cifrado de su entorno de Amazon MWAA.

        ii. Elija una de sus claves de AWS KMS o introduzca el ARN de clave de AWS KMS: si decide especificar una clave administrada por el cliente en este paso, debe especificar un identificador de clave de AWS KMS o un ARN. Amazon MWAA no admite los alias de AWS KMS ni las claves multirregionales. La clave de AWS KMS que especifique también debe usarse para el cifrado en su entorno de Amazon MWAA.
9. Opcional: Configuración avanzada. Si desea habilitar Amazon S3 Object Lock:

    a. Elija Ajustes avanzados y, luego, Habilitar.

    > **Important**
    >
    > Al habilitar Object Lock, se bloquearán permanentemente los objetos de este bucket. Para obtener más información, consulte Bloqueo de objetos mediante Amazon S3 Object Lock en la guía del usuario de Amazon Simple Storage Service.

    b. Elija la confirmación.
10. Elija Crear bucket.

Siguientes pasos

- Aprenda a crear la red de Amazon VPC necesaria para un entorno en Creación de la red de VPC.

- Obtenga información sobre cómo administrar los permisos de acceso en ¿Cómo se configuran los permisos de los buckets de ACL?
- Aprenda a eliminar un bucket de almacenamiento en ¿Cómo se elimina un bucket de S3?.

Creación de la red de VPC

Los flujos de trabajo de Amazon para Apache Airflow requieren una VPC de Amazon y componentes de red específicos para dar soporte a un entorno. En esta guía se describen las diferentes opciones para crear la red de Amazon VPC para un entorno de Amazon Managed Workflows para Apache Airflow.

> **Note**
>
> Apache Airflow funciona mejor en un entorno de red de baja latencia. Si utiliza una Amazon VPC existente que enruta el tráfico a otra región o a un entorno local, le recomendamos que agregue puntos de conexión de AWS PrivateLink para Amazon SQS, CloudWatch, Amazon S3 y AWS KMS. Para más información sobre la configuración de AWS PrivateLink para Amazon MWAA, consulte Creación de una red VPC de Amazon sin acceso a Internet.

Contenido

Requisitos previos

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Antes de empezar

- La red de VPC que especifique para su entorno no se puede cambiar después crearlo.
- Puede usar el enrutamiento público o privado para su servidor web de Amazon VPC y Apache Airflow. Para acceder a una lista de opciones, consulte the section called “Ejemplos de casos de uso para un modo de acceso a Amazon VPC y Apache Airflow”.

Opciones para crear la red de Amazon VPC

En la sección siguiente se describen las opciones disponibles para crear la red de Amazon VPC para un entorno.

> **Note**
>
> Amazon MWAA no admite el uso de la zona de disponibilidad (AZ) `use1-az3` en la región Este de EE. UU. (Norte de Virginia). Cuando crea la VPC para Amazon MWAA en la región Este de EE. UU. (Norte de Virginia), debe asignar la `AvailabilityZone` de forma explícita en la plantilla CloudFormation (CFN). No se debe asignar a `use1-az3` el nombre de la zona de disponibilidad asignada. Puede obtener la asignación detallada de los nombres de las zonas de disponibilidad (AZ) a sus correspondientes ID de zona de disponibilidad mediante la ejecución del siguiente comando:
>
> ```
> aws ec2 describe-availability-zones --region us-east-1
> ```

Opción 1: crear la red de VPC en la consola de Amazon MWAA

En la siguiente sección, se muestra cómo crear una red de Amazon VPC en la consola de Amazon MWAA. Esta opción usa Enrutamiento público a través de Internet. Se puede usar para un servidor web de Apache Airflow con el modo de acceso de Red privada o Red pública.

En la siguiente imagen, se muestra dónde puede encontrar el botón Create MWAA VPC (Crear VPC de MWAA) en la consola de Amazon MWAA.

Opción 2: Creación de una red Amazon VPC con acceso a Internet

La siguiente plantilla de CloudFormation crea una red de Amazon VPC con acceso a Internet en su región predeterminada de Región de AWS. Esta opción usa Enrutamiento público a través de Internet. Esta plantilla se puede usar para un servidor web de Apache Airflow con el modo de acceso de Red privada o Red pública.

1. Copie el contenido de la siguiente plantilla y guárdelo localmente como `cfn-vpc-public-private.yaml`. También puede descargar la plantilla.

```
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. En el símbolo del sistema, vaya hasta el directorio en el que está almacenado `cfn-vpc-public-private.yaml`. Por ejemplo:

```
cd mwaaproject
```

3. Utilice el comando `aws cloudformation create-stack` para crear la pila con la AWS CLI.

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

> **Note**
>
> Se tardan unos 30 minutos en crear la infraestructura de Amazon VPC.

Opción 3: Creación de una red de Amazon VPC sin acceso a Internet

La siguiente plantilla de CloudFormation crea una red de Amazon VPC sin acceso a Internet en su Región de AWS predeterminada.

Esta opción usa Enrutamiento privado sin acceso a Internet. Esta plantilla se puede usar solo para un servidor web de Apache Airflow con el modo de acceso a la red privada. Crea los puntos de conexión de VPC necesarios para los servicios de AWS que utiliza un entorno.

1. Copie el contenido de la siguiente plantilla y guárdelo localmente como `cfn-vpc-private.yaml`. También puede descargar la plantilla.

```
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


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. En el símbolo del sistema, vaya hasta el directorio en el que está almacenado `cfn-vpc-private.yml`. Por ejemplo:

```
cd mwaaproject
```

3. Utilice el comando `aws cloudformation create-stack` para crear la pila con la AWS CLI.

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

> **Note**
>
> Se tardan unos 30 minutos en crear la infraestructura de Amazon VPC.

4. Deberá crear un mecanismo para acceder a estos puntos de conexión de VPC desde su ordenador. Consulte Administración del acceso a los puntos de conexión de Amazon VPC específicos del servicio en Amazon MWAA para obtener más información.

> **Note**
>
> Puede restringir aún más el acceso saliente en el CIDR de su grupo de seguridad de Amazon MWAA. Por ejemplo, puede restringirse a sí mismo añadiendo una regla de salida autorreferenciada, la lista de prefijos de Amazon S3 y el CIDR de su Amazon VPC.

Siguientes pasos

- Obtenga información sobre cómo crear un entorno Amazon MWAA en Creación de entornos de Amazon MWAA.
- Aprenda a crear un túnel VPN desde su ordenador a su Amazon VPC con enrutamiento privado en Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN.

Creación de entornos de Amazon MWAA

Amazon Managed Workflows para Apache Airflow configura Apache Airflow en un entorno de la versión que elija utilizando el mismo Airflow de código abierto y la misma interfaz de usuario que Apache. En esta guía se describen los pasos para crear entornos de Amazon MWAA.

Contenido

Antes de empezar

- La red de VPC que especifica para su entorno no se puede cambiar después crearlo.
- Necesita un bucket de Amazon S3 configurado para bloquear todo el acceso público, con el control de versiones del bucket activado.

- Necesita una cuenta de Cuenta de AWS con permisos para usar Amazon MWAA y un permiso en AWS Identity and Access Management (IAM) para crear roles de IAM. Si elige el modo de acceso de red privada para el servidor web de Apache Airflow, que limita el acceso de Apache Airflow dentro de su Amazon VPC, necesitará permiso en IAM para crear puntos de conexión de Amazon VPC.

> **Note**
>
> Amazon MWAA determina la red de forma dinámica durante la creación. Si usa subredes IPv6, Amazon MWAA crea una conectividad de enlace privado IPv6 con la base de datos y el servidor web. Amazon MWAA no admite la transición entre tipos de red y no puede actualizar los entornos existentes a IPv6.

Versiones de Apache Airflow

Las siguientes versiones de Apache Airflow son compatibles con Amazon Managed Workflows para Apache Airflow.

> **Note**
>
> - A partir del 30 de diciembre de 2025, Amazon MWAA dejará de ser compatible con las versiones v2.4.3, v2.5.1 y v2.6.3 de Apache Airflow. Para obtener más información, consulta Preguntas frecuentes y compatibilidad de Apache Airflow.
> - A partir de Apache Airflow v2.2.2, Amazon MWAA admite la instalación de requisitos de Python, paquetes de proveedores y complementos personalizados directamente en el servidor web Apache Airflow.
> - A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que se está usando.
>
>   Para obtener más información sobre la configuración de restricciones en su archivo de requisitos, consulte cómo instalar dependencias de Python.

Versión de Apache Airflow	Fecha de la versión de Apache Airflow	Fecha de disponibilidad en Amazon MWAA	Restricciones de Apache Airflow	Versión de Python
v3.0.6	de agosto de 29, 2025	1 de octubre de 2025	archivo de restricciones v3.0.6	Python 3.12
v2.10.3	de noviembre de 4, 2024	18 de diciembre de 2024	archivo de restricciones v2.10.3	Python 3.11
v2.10.1	de septiembre de 5, 2024	26 de septiembre de 2024	Archivo de restricciones v2.10.1	Python 3.11
v2.9.2	de junio de 10, 2024	9 de julio de 2024	Archivo de restricciones v2.9.2	Python 3.11
v2.8.1	de enero de 19, 2024	23 de febrero de 2024	Archivo de restricciones v2.8.1	Python 3.11
v2.7.2	de octubre de 12, 2023	6 de noviembre de 2023	Archivo de restricciones v2.7.2	Python 3.11

Para obtener más información sobre la migración de sus implementaciones autoadministradas de Apache Airflow o la migración de un entorno Amazon MWAA existente, incluidas las instrucciones para realizar copias de seguridad de su base de datos de metadatos, consulte la guía de migración a Amazon MWAA.

Creación de un entorno

En la siguiente sección se describen los pasos para crear entornos de Amazon MWAA.

Paso 1: especificar los detalles

Pasos para especificar los detalles del entorno

1. Abra la consola de Amazon MWAA.
2. Seleccione su Región de AWS.
3. Seleccione Crear entorno.
4. Siga los pasos que se detallan a continuación en la página Especificar detalles, en Detalles del entorno:
   a. Escriba un nombre para el entorno en Name (Nombre).
   b. Elija la versión Apache Airflow en versión de Airflow.

   > **Note**
   >
   > Si no se especifica ningún valor, el valor predeterminado será la última versión de Apache Airflow. La última versión disponible es Apache Airflow v3.0.6.

5. En Código DAG de Amazon S3, especifique lo siguiente:
   a. Un bucket de S. Elija Explorar S3 y seleccione su bucket de Amazon S3 o introduzca el URI de Amazon S3.
   b. Una carpeta DAG. Elija Explorar S3 y seleccione la carpeta `dags` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.
   c. Un archivo de complementos (opcional). Elija Explorar S3 y seleccione el archivo `plugins.zip` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.
   d. Un archivo de requisitos (opcional). Elija Explorar S3 y seleccione el archivo `requirements.txt` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.
   e. Un archivo de script de inicio (opcional). Elija Explorar S3 y seleccione el archivo de script en su bucket de Amazon S3 o introduzca el URI de Amazon S3.
6. Elija Siguiente.

Paso 2: configurar los ajustes avanzados

Configuración de opciones avanzadas

1. En la página Configurar los ajustes avanzados, en Redes,

   - Elija su Amazon VPC.

     Este paso rellena dos de las subredes privadas de su Amazon VPC.

2. En Webserver access (Acceso al servidor web), seleccione el modo de acceso de Apache Airflow preferido:

   a. Una red privada. Esto limita el acceso a la interfaz de usuario de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno. Para este paso, necesita permiso para crear puntos de conexión de VPC de Amazon.

   > **Note**
   >
   > Elija la opción red privada si solo se puede acceder a la UI de Apache Airflow desde una red corporativa y no necesita acceder a repositorios públicos para cumplir con los requisitos de instalación del servidor web. Si elige este modo de acceso, deberá crear un mecanismo para acceder al servidor web de Apache Airflow en su VPC de Amazon. Para obtener más información, consulta Acceso al punto de conexión de VPC del servidor web Apache Airflow (acceso mediante red privada).

   b. Red pública. Esto permite que los usuarios con acceso a la política de IAM de su entorno accedan a la UI de Apache Airflow a través de Internet.

3. En Security groups (Grupos de seguridad), elija el grupo de seguridad que se haya usado para proteger su VPC de Amazon:

   a. Por defecto, Amazon MWAA crea un grupo de seguridad en su VPC de Amazon con reglas de entrada y salida específicas en Crear un nuevo grupo de seguridad.

   b. Opcional. Desactive la casilla de verificación de Crear nuevo grupo de seguridad para seleccionar hasta 5 grupos de seguridad.

> **Note**
>
> Debe configurarse un grupo de seguridad de Amazon VPC existente con reglas de entrada y salida específicas para permitir el tráfico de red. Consulte Seguridad en la VPC en Amazon MWAA para obtener más información.

4. En Clase de entorno, elija una clase de entorno.

   Le recomendamos que elija el tamaño más pequeño necesario para soportar su carga de trabajo. Puede cambiar la clase de entorno en cualquier momento.

5. En Número máximo de procesos de trabajo, especifique el número máximo de procesos de trabajo de Apache Airflow que se ejecutarán en el entorno.

   Para obtener más información, consulta Ejemplo de caso de uso de alto rendimiento.

6. Especifique el número máximo de servidores web y el número mínimo de servidores web para configurar la forma en la que Amazon MWAA escala los servidores web Apache Airflow en su entorno.

   Para obtener más información sobre el escalado automático del servidor web, consulte the section called “Configuración del escalado automático del servidor web”.

7. En Cifrado, elija una opción de cifrado de datos:

   a. De forma predeterminada, Amazon MWAA usa una clave de AWS propia para cifrar los datos.

   b. Opcional. Seleccione Personalizar la configuración de cifrado (avanzada) para elegir una clave de AWS KMS diferente. Si decide especificar una clave administrada por el cliente en este paso, debe especificar un identificador de clave de AWS KMS o un ARN. Amazon MWAA no admite alias ni claves multirregionales de AWS KMS. Si especificó una clave de Amazon S3 para el cifrado del servidor en su bucket de Amazon S3, debe especificar la misma clave para su entorno de Amazon MWAA.

> **Note**
>
> Debe tener permisos sobre la clave para seleccionarla en la consola de Amazon MWAA. También debe conceder permisos para que Amazon MWAA utilice la clave adjuntando la política descrita en Asociación de políticas de claves.

8. Recomendado. En Monitorización, elija una o más categorías de registro para Configuración del registro de Airflow para enviar los registros de Apache Airflow a Registros de CloudWatch:

    a. Registros de tareas de Airflow. En Nivel de registro, elija qué tipo de registro de tareas de Apache Airflow se enviará a CloudWatch Logs.

    b. Registros del servidor web de Airflow. En Log level (Nivel de registro), elija qué tipo de registro del servidor web de Apache Airflow se enviará a los registros de CloudWatch.

    c. Registros del programador de Airflow. En Nivel de registro, elija qué tipo de registro del programador de Apache Airflow se enviará a CloudWatch Logs.

    d. Registros de procesos de trabajo de Airflow. En Nivel de registro, elija qué tipo de registro de procesos de trabajo de Apache Airflow se enviará a CloudWatch Logs.

    e. Registros de procesamiento del DAG de Airflow. En Nivel de registro, elija qué tipo de registro del DAG Apache Airflow se enviará a CloudWatch Logs.

9. Opcional. Para ver las opciones de configuración de Airflow, elija Agregar una opción de configuración personalizada.

    Puede elegir de la lista desplegable sugerida de opciones de configuración de Apache Airflow para su versión de Apache Airflow o especificar opciones de configuración personalizadas. Por ejemplo, `core.default_task_retries`: `3`.

10. Opcional. En Etiquetas, elija Agregar nueva etiqueta para asociar etiquetas a su entorno. Por ejemplo, `Environment`: `Staging`.

11. En Permisos, elija un rol de ejecución:

    a. Por defecto, Amazon MWAA crea un rol de ejecución en Crear un rol nuevo. Para usar esta opción, debe tener permiso para crear roles de IAM.

    b. Opcional. Elija Introduzca el ARN del rol para escribir el nombre de recurso de Amazon (ARN) de un rol de ejecución existente.

12. Elija Siguiente.

Paso 3: consultar y crear

Pasos para consultar un resumen del entorno

- Consulte el resumen del entorno y elija Creación de entorno.

> **Note**
>
> Se tarda entre 20 y 30 minutos en crear un entorno.

Siguientes pasos

- Conozca cómo crear un bucket de Amazon S3 en Creación de un bucket de Amazon S3 para Amazon MWAA.

Administración del acceso a un entorno de Amazon MWAA

Se debe permitir que Amazon Managed Workflows para Apache Airflow utilice otros servicios y recursos de AWS que se utilizan en los entornos. Además, los usuarios deben tener permiso para acceder a un entorno de Amazon MWAA y a su interfaz de usuario de Apache Airflow en AWS Identity and Access Management (IAM). En esta sección, se describe el rol de ejecución que se usa para conceder acceso a su entorno a los recursos de AWS y cómo se añaden los permisos. Asimismo, se indican los permisos de la Cuenta de AWS necesarios para poder acceder a su entorno de Amazon MWAA y a la UI de Apache Airflow.

Temas

Acceso a un entorno de Amazon MWAA

Para usar Amazon Managed Workflows para Apache Airflow, debe usar una cuenta y entidades de IAM que dispongan de los permisos necesarios. En esta página, se describen las políticas de acceso que puede asociar a su equipo de desarrollo y a los usuarios de Apache Airflow en relación con su entorno de Amazon Managed Workflows para Apache Airflow.

Recomendamos usar credenciales temporales y configurar identidades federadas con grupos y roles para acceder a sus recursos de Amazon MWAA. Como práctica recomendada, aconsejamos que no asocie directamente las políticas a los usuarios de IAM. En su lugar, defina grupos o roles para proporcionar acceso temporal a los recursos de AWS.

Un rol de IAM es una identidad de IAM que puede crear en su cuenta y que tiene permisos específicos. Un rol de IAM es similar a un usuario de IAM en que se trata de una identidad de AWS con políticas de permisos que determinan lo que la identidad puede hacer y lo que no en AWS. No obstante, en lugar de asociarse exclusivamente a una persona, la intención es que cualquier usuario pueda asumir un rol que necesite. Además, un rol no tiene asociadas credenciales a largo plazo estándar, como una contraseña o claves de acceso. En su lugar, cuando se asume un rol, este proporciona credenciales de seguridad temporales para la sesión de rol.

Para asignar permisos a identidades federadas, cree un rol y defina permisos para este. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte Crear un rol para un proveedor de identidad de terceros (federación) en la Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center.

Puede utilizar un rol de IAM de su cuenta para conceder otros permisos de Cuenta de AWS de acceso a los recursos de su cuenta. Encontrará un ejemplo de ello en la sección Tutorial: delegar el acceso entre Cuentas de AWS con roles de IAM de la guía del usuario de IAM.

Secciones

Funcionamiento

No todas las entidades de AWS Identity and Access Management (IAM) pueden acceder a los recursos y servicios que se utilizan en un entorno de Amazon MWAA. Deberá crear una política que conceda permiso a los usuarios de Apache Airflow para acceder a estos recursos. Por ejemplo, deberá conceder acceso a su equipo de desarrollo de Apache Airflow.

Amazon MWAA usa estas políticas para validar si un usuario tiene los permisos necesarios para realizar una acción concreta en la consola de AWS o a través de las API que usa un entorno determinado.

Puede utilizar las políticas JSON de este tema para crear una política para sus usuarios de Apache Airflow en IAM, para asociarla después a un usuario, grupo o rol en IAM.

- AmazonMWAAFullConsoleAccess: utilice esta política para conceder permiso para configurar un entorno en la consola de Amazon MWAA.
- AmazonMWAAFullApiAccess: utilice esta política para conceder acceso a todas las API de Amazon MWAA que se utilizan para administrar un entorno.
- AmazonMWAAReadOnlyAccess: use esta política para conceder acceso para ver los recursos que usa un entorno en la consola de Amazon MWAA.
- AmazonMWAAWebServerAccess: use esta política para conceder acceso al servidor web de Apache Airflow.
- AmazonMWAAAirflowCliAccess: utilice esta política para conceder acceso para ejecutar los comandos de la CLI de Apache Airflow.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center.
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones de Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Política de acceso completo a la consola: AmazonMWAAFullConsoleAccess

Es posible que un usuario deba acceder a la política de permisos de `AmazonMWAAFullConsoleAccess` si necesita configurar un entorno en la consola de Amazon MWAA.

> **Note**
>
> Su política de acceso completo a la consola debe incluir permisos para realizar la acción `iam:PassRole`. Esto permitirá al usuario transferir roles vinculados a servicios y roles de ejecución a Amazon MWAA. Amazon MWAA asumirá estos roles para poder llamar a otros servicios de AWS en su nombre. En el ejemplo siguiente se utiliza la clave de condición `iam:PassedToService` para especificar la entidad principal del servicio de Amazon MWAA (`airflow.amazonaws.com`) como servicio al que se le puede transferir un rol.
> Para obtener más información acerca de `iam:PassRole`, consulte cómo conceder permisos a un usuario para transferir un rol a un servicio de AWS en la guía del usuario de IAM.

Utilice la siguiente política si desea crear y administrar sus entornos de Amazon MWAA mediante una Clave propiedad de AWS para el cifrado en reposo.

Uso de Clave propiedad de AWS

JSON

```
{
    "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::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333: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/*"
            ]
        }
    ]
}
```

Utilice la siguiente política si desea crear y administrar sus entornos de Amazon MWAA mediante una clave administrada por el cliente para el cifrado en reposo. Para utilizar una clave administrada por el cliente, la entidad principal de IAM debe tener permiso para acceder a los recursos de AWS KMS mediante la clave almacenada en su cuenta.

Uso de claves administradas por el cliente

JSON

```
{
    "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::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333: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:*:111122223333: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 de acceso completo a la consola y a las API: AmazonMWAAFullApiAccess

Es posible que un usuario deba obtener acceso a la política de permisos `AmazonMWAAFullApiAccess` en caso de tener que acceder a todas las API de Amazon MWAA que se usan para administrar un entorno. Esta política no otorga permisos para acceder a la interfaz de usuario de Apache Airflow.

> **Note**
>
> Las políticas de acceso completo a las API deben incluir permisos para realizar la acción `iam:PassRole`. Esto permitirá al usuario transferir roles vinculados a servicios y roles de

ejecución a Amazon MWAA. Amazon MWAA asumirá estos roles para poder llamar a otros servicios de AWS en su nombre. En el ejemplo siguiente se utiliza la clave de condición `iam:PassedToService` para especificar la entidad principal del servicio de Amazon MWAA (`airflow.amazonaws.com`) como servicio al que se le puede transferir un rol.
Para obtener más información acerca de `iam:PassRole`, consulte cómo conceder permisos a un usuario para transferir un rol a un servicio de AWS en la guía del usuario de IAM.

Utilice la siguiente política si desea crear y administrar sus entornos de Amazon MWAA mediante una Clave propiedad de AWS para el cifrado en reposo.

Uso de Clave propiedad de AWS

JSON

```
{
   "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/*"
         ]
      }
   ]
}
```

Utilice la siguiente política si desea crear y administrar sus entornos de Amazon MWAA mediante una clave administrada por el cliente para el cifrado en reposo. Para utilizar una clave administrada por el cliente, la entidad principal de IAM debe tener permiso para acceder a los recursos de AWS KMS mediante la clave almacenada en su cuenta.

Uso de claves administradas por el cliente

JSON

```
{
    "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:*:111122223333: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 acceso de solo lectura a la consola: AmazonMWAAReadOnlyAccess

Es posible que un usuario deba obtener acceso a la política de permisos `AmazonMWAAReadOnlyAccess` si necesita ver los recursos que usa un entorno en la página de detalles del entorno de la consola de Amazon MWAA. Esta política no permite al usuario crear nuevos entornos, editar los existentes ni ver la UI de Apache Airflow.

JSON

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

Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess

Es posible que un usuario deba obtener acceso a la política de permisos `AmazonMWAAWebServerAccess` si necesita acceder a la UI de Apache Airflow. Esta política no permite al usuario ver los entornos de la consola de Amazon MWAA ni usar las API de Amazon MWAA para realizar acciones. Especifique el rol `Admin`, `Op`, `User`, `Viewer` o `Public` en

`{airflow-role}` para personalizar el nivel de acceso del usuario al token web. Para obtener más información, consulte la sección Roles predeterminados en la guía de referencia de Apache Airflow.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
             "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> - Amazon MWAA permite integrar IAM con los cinco roles predeterminados de control de acceso basado en roles (RBAC) de Apache Airflow. Para obtener más información acerca de cómo usar los roles personalizados de Apache Airflow, consulte the section called "Tutorial: Restricción de usuarios a un subconjunto de DAG".
> - El campo `Resource` en esta política puede usarse para especificar los roles de control de acceso basado en roles de Apache Airflow para el entorno de Amazon MWAA. Sin embargo, no admite el ARN (nombre de recurso de Amazon) del entorno de Amazon MWAA en el campo `Resource` de la política.

Política de acceso a la API de REST de Apache Airflow: AmazonMWAARestAPIAccess.

Para acceder a la API de REST de Apache Airflow, debe otorgar el permiso `airflow:InvokeRestApi` en su política de IAM. En el siguiente ejemplo de política, especifique

el rol de `Admin`, `Op`, `User`, `Viewer` o `Public` en `{airflow-role}` para personalizar el nivel de acceso del usuario. Para obtener más información, consulte la sección Roles predeterminados en la guía de referencia de Apache Airflow.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> - Mientras configura un servidor web privado, la acción `InvokeRestApi` no se puede invocar desde fuera de una nube privada virtual (VPC). Puede utilizar la clave `aws:SourceVpc` para aplicar un control de acceso más detallado para esta operación. Para obtener más información, consulte aws:SourceVpc
> - El campo `Resource` en esta política puede usarse para especificar los roles de control de acceso basado en roles de Apache Airflow para el entorno de Amazon MWAA. Sin embargo, no admite el ARN (nombre de recurso de Amazon) del entorno de Amazon MWAA en el campo `Resource` de la política.

Política de la CLI de Apache Airflow: AmazonMWAAAirflowCliAccess

Es posible que un usuario deba obtener acceso a la política de permisos `AmazonMWAAAirflowCliAccess` si necesita ejecutar comandos de la CLI de Apache Airflow

(por ejemplo `trigger_dag`). Esta política no permite al usuario ver los entornos de la consola de Amazon MWAA ni usar las API de Amazon MWAA para realizar acciones.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
${EnvironmentName}"
        }
    ]
}
```

Creación de una política JSON

Puede crear la política JSON y asociarla a su usuario, rol o grupo en la consola de IAM. Los siguientes pasos describen cómo crear la política JSON en IAM.

Pasos para crear la política JSON

1. Abra la página Políticas en la consola de IAM.
2. Elija Crear política.
3. Seleccione la pestaña JSON.
4. Añada su política JSON.
5. Elija Revisar política.
6. Introduzca un valor en el campo de texto Nombre y Descripción (opcional).

   Por ejemplo, puede asignar a la política el nombre “`AmazonMWAAReadOnlyAccess`”.
7. Elija Crear política.

Ejemplo de caso de uso para asociar políticas a un grupo de desarrolladores

Supongamos que utiliza un grupo de IAM denominado "`AirflowDevelopmentGroup`" para aplicar permisos a todos los desarrolladores de su equipo de desarrollo de Apache Airflow. Estos usuarios deberán obtener acceso a las políticas de permisos de `AmazonMWAAFullConsoleAccess`, `AmazonMWAAAirflowCliAccess` y `AmazonMWAAWebServerAccess`. En esta sección se describe cómo crear un grupo en IAM, cómo crear y asociar estas políticas y cómo vincular el grupo a un usuario de IAM. En los siguientes pasos, se presupone que usa una clave propiedad de AWS.

Pasos para crear la política AmazonMWAAFullConsoleAccess

1. Descargue la política de acceso AmazonMWAAFullConsoleAccess.
2. Abra la página Políticas en la consola de IAM.
3. Elija Crear política.
4. Seleccione la pestaña JSON.
5. Pegue la política JSON de `AmazonMWAAFullConsoleAccess`.
6. Sustituya los valores siguientes:
   a. *123456789012*: la identificación de su Cuenta de AWS (por ejemplo, `0123456789`)
   b. *{your-kms-id}*: el identificador único de una clave administrada por el cliente, solo en caso de que utilice una clave administrada por el cliente para el cifrado en reposo.
7. Elija Consultar política.
8. En Nombre, escriba "**`AmazonMWAAFullConsoleAccess`**".
9. Elija Crear política.

Pasos para crear la política AmazonMWAAWebServerAccess

1. Descargue la política de acceso a AmazonMWAAWebServerAccess.
2. Abra la página Políticas en la consola de IAM.
3. Elija Crear política.
4. Seleccione la pestaña JSON.
5. Pegue la política JSON de `AmazonMWAAWebServerAccess`.
6. Sustituya los valores siguientes:

   a. *us-east-1*: la región de su entorno de Amazon MWAA (por ejemplo, us-east-1)
   b. *123456789012*: la identificación de su Cuenta de AWS (por ejemplo, 0123456789)
   c. *{your-environment-name}*: el nombre de su entorno de Amazon MWAA (por ejemplo, MyAirflowEnvironment).
   d. *{airflow-role}*: el rol predeterminado del Admin en Apache Airflow.
7. Elija Revisar política.
8. En Nombre, escriba "**AmazonMWAAWebServerAccess**".
9. Elija Crear política.

Pasos para crear la política AmazonMWAAAirflowCliAccess

1. Descargue la política de acceso AmazonMWAAAirflowCliAccess.
2. Abra la página Políticas en la consola de IAM.
3. Elija Crear política.
4. Seleccione la pestaña JSON.
5. Pegue la política JSON de AmazonMWAAAirflowCliAccess.
6. Elija Consultar política.
7. En Nombre, escriba "**AmazonMWAAAirflowCliAccess**".
8. Elija Crear política.

Pasos para crear los grupos

1. Abra la Página del grupo de registros en la consola de IAM.
2. Escriba un nombre de AirflowDevelopmentGroup.
3. Elija Paso siguiente.
4. Escriba "AmazonMWAA" en Filtrar para filtrar los resultados.
5. Elija las tres políticas que ha creado.
6. Elija Paso siguiente.
7. Elija Crear grupo.

Pasos para asociarlas a un usuario

1. Abra la página Users en la consola de IAM.
2. Elija un usuario.
3. Elija Grupos.
4. Elija Añadir usuario al grupo o grupos.
5. Elija el AirflowDevelopmentGroup.
6. Elija Añadir a grupos.

Siguientes pasos

- Aprenda a generar un token para acceder a la interfaz de usuario de Apache Airflow en Acceso a Apache Airflow.
- Encontrará más información sobre cómo crear políticas de IAM en Creación de políticas de IAM.

Roles vinculados a servicios para Amazon MWAA

Amazon Managed Workflows para Apache Airflow utiliza los roles vinculados a servicios de AWS Identity and Access Management (IAM). Los roles vinculados a servicios son un tipo único de rol de IAM vinculado directamente a Amazon MWAA. Los roles vinculados a servicios están predefinidos por Amazon MWAA e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre.

Un rol vinculado a servicios simplifica la configuración de Amazon MWAA porque ya no tendrá que agregar de forma manual los permisos necesarios. Amazon MWAA define los permisos de sus roles vinculados al servicio y, a menos que esté definido de otra manera, solo Amazon MWAA puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda adjuntar a ninguna otra entidad de IAM.

Solo es posible eliminar un rol vinculado a un servicio después de eliminar sus recursos relacionados. Así se protegen los recursos de Amazon MWAA, ya que se evita que se puedan eliminar los permisos de acceso a los recursos accidentalmente.

Para obtener información sobre otros servicios que admiten roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Yes (Sí) en la

columna Service-linked roles (Roles vinculados a servicios). Elija una opción Yes (Sí) con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de roles vinculados a un servicio para Amazon MWAA

Amazon MWAA utiliza el rol vinculado al servicio denominado "`AWSServiceRoleForAmazonMWAA`": este rol creado en su cuenta concede a Amazon MWAA acceso a los siguientes servicios de AWS:

- Registros de Amazon CloudWatch (registros de CloudWatch): sirven para crear grupos de registro para los registros de Apache Airflow.
- Amazon CloudWatch (CloudWatch): sirve para publicar métricas relacionadas con su entorno y los componentes subyacentes en su cuenta.
- Amazon Elastic Compute Cloud (Amazon EC2): sirve para crear los siguientes recursos:
  - Un punto de conexión de VPC de Amazon en su VPC para un clúster de base de datos de Amazon Aurora PostgreSQL administrado por AWS que usarán el programador y el proceso de trabajo de Apache Airflow.
  - Un punto de conexión de Amazon VPC adicional para habilitar el acceso de red al servidor web, en caso de que elija la opción de red privada para el servidor web Apache Airflow.
  - Interfaces de red elásticas (ENI) en su VPC de Amazon para habilitar el acceso de red a los recursos de AWS alojados en su VPC de Amazon.

La política de confianza siguiente permite que la entidad principal del servicio asuma el rol vinculado al servicio. La entidad principal del servicio de Amazon MWAA es `airflow.amazonaws.com`, tal y como se refleja en la política.

JSON

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

```
}
```

La política de permisos de roles llamada “AmazonMWAAServiceRolePolicy” permite que Amazon MWAA complete las siguientes acciones en los recursos especificados:

JSON

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Para obtener más información, consulte Permisos de roles vinculados a servicios en la guía del usuario de IAM.

Creación de roles vinculados a servicios para Amazon MWAA

No necesita crear manualmente un rol vinculado a servicios. Cuando crea un entorno de Amazon MWAA environment en la Consola de administración de AWS, la AWS CLI, o la API de AWS, Amazon MWAA environment crea el rol vinculado a servicios en su nombre.

Si elimina este rol vinculado a servicios y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al crear otro entorno de desarrollo, Amazon MWAA environment vuelve a crear el rol vinculado a servicios en su nombre.

Edición roles vinculados a servicios para Amazon MWAA

Amazon MWAA no le permite editar roles vinculados al servicio AWSServiceRoleForAmazonMWAA. Después de crear un rol vinculado al servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al rol. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte cómo editar un rol vinculado a servicios en la guía del usuario de IAM.

Eliminación de roles vinculados a servicios para Amazon MWAA

Si ya no necesita usar una característica o servicio que requieran un rol vinculado a un servicio, le recomendamos que elimine dicho rol. Así no tendrá una entidad no utilizada que no se supervise ni mantenga de forma activa.

Al eliminar un entorno de Amazon MWAA, Amazon MWAA elimina todos los recursos asociados al mismo que se utilizan como parte del servicio. Sin embargo, debe esperar a que Amazon MWAA termine de eliminar su entorno antes de intentar eliminar el rol vinculado al servicio. Si elimina el rol vinculado al servicio antes de que Amazon MWAA haya eliminado el entorno, es posible que Amazon MWAA no pueda eliminar todos los recursos asociados al entorno.

Eliminación manual del rol vinculado a servicios mediante IAM

Utilice la consola de IAM, la AWS CLI o la API de AWS para eliminar el rol vinculado al servicio AWSServiceRoleForAmazonMWAA. Para obtener más información, consulte cómo eliminar un rol vinculado a servicios en la guía del usuario de IAM.

Regiones admitidas para los roles vinculados al servicio de Amazon MWAA

Amazon MWAA permite usar roles vinculados al servicio en todas las regiones en las que se encuentra disponible el servicio. Para obtener más información, consulte Amazon Managed Workflows para puntos de conexión y cuotas de Apache Airflow.

Actualizaciones de políticas

Cambio	Descripción	Fecha
Amazon MWAA actualiza su política de permisos de roles vinculados al servicio	`AmazonMWAAServiceRolePolicy` – Amazon MWAA actualiza la política de permisos relativa a su rol vinculado al servicio para otorgar a Amazon MWAA permiso para publicar métricas adicionales relacionadas con los recursos subyacentes del servicio en las cuentas de	18 de noviembre de 2022

Cambio	Descripción	Fecha
	los clientes. Puede consultar las nuevas métricas en `AWS/MWAA`.	
Amazon MWAA comenzó a realizar el seguimiento de los cambios	Amazon MWAA comenzó a realizar un seguimiento de los cambios en la política de permisos de roles vinculados al servicio administrados por AWS.	18 de noviembre de 2022

Rol de ejecución de Amazon MWAA

Los roles de ejecución son roles de AWS Identity and Access Management (IAM) con una política de permisos que otorgan a Amazon Managed Workflows para Apache Airflow permiso para invocar los recursos de otros servicios de AWS en su nombre. Algunos de los recursos que pueden invocarse son su bucket de Amazon S3, su clave propiedad de AWS y los registros de CloudWatch. Los entornos de Amazon MWAA necesitan un rol de ejecución para cada entorno. En esta página, se describe cómo usar y configurar los roles de ejecución de su entorno para que Amazon MWAA pueda acceder a otros recursos de AWS que se utilizan en su entorno.

Contenido

- Ejemplo de política para una clave administrada por el cliente
- Ejemplo de política para una clave propiedad de AWS
- Siguientes pasos

Información general sobre los roles de ejecución

Amazon MWAA obtiene permiso para usar los demás servicios de AWS que se usan en su entorno gracias a los roles de ejecución. Los roles de ejecución de Amazon MWAA necesitan permiso para utilizar los siguientes servicios de AWS utilizados en un entorno:

- Amazon CloudWatch (CloudWatch): sirve para enviar métricas y registros de Apache Airflow.
- Amazon Simple Storage Service (Amazon S3): sirve para analizar el código de los DAG de su entorno y los archivos auxiliares (por ejemplo, un `requirements.txt`).
- Amazon Simple Queue Service (Amazon SQS): sirve para poner en una cola de Amazon SQS que sea propiedad de Amazon MWAA las tareas de Apache Airflow de su entorno.
- AWS Key Management Service (AWS KMS): sirve para cifrar los datos de su entorno (mediante una clave propiedad de AWS o su clave administrada por el cliente).

> **Note**
>
> Si ha optado por que Amazon MWAA use una clave de KMS propiedad de AWS para cifrar sus datos, deberá definir permisos en una política adjunta a su rol de ejecución de Amazon MWAA que otorguen acceso a claves de KMS arbitrarias almacenadas fuera de su cuenta mediante Amazon SQS. Para que el rol de ejecución de su entorno pueda acceder a claves de KMS arbitrarias, deben darse las dos condiciones siguientes:
>
> - La clave de KMS de una cuenta externa debe permitir el acceso entre cuentas por medio de su política de recursos.
> - El código de sus DAG debe acceder a una cola de Amazon SQS que comienza con `airflow-celery-` en la cuenta de terceros y utiliza la misma clave de KMS para el cifrado.
>
> Para mitigar los riesgos asociados con el acceso entre cuentas a los recursos, recomendamos consultar el código incluido en sus DAG para garantizar que sus flujos de trabajo no accedan a colas arbitrarias de Amazon SQS ajenas a su cuenta. Además, puede usar una clave de KMS administrada por el cliente que se encuentre almacenada en su propia cuenta para administrar el cifrado en Amazon MWAA. Esto limitará el rol de

ejecución de su entorno para que únicamente pueda accederse a la clave de KMS de su cuenta.
Tenga en cuenta que una vez que haya elegido una opción de cifrado, ya no podrá cambiar su elección para los entornos existentes.

Los roles de ejecución también necesitan permiso para realizar las siguientes acciones de IAM:

- `airflow:PublishMetrics`: permite que Amazon MWAA supervise el estado de un entorno.

Permisos que se asocian de forma predeterminada

Puede usar las opciones predeterminadas de la consola de Amazon MWAA para crear un rol de ejecución y una clave propiedad de AWS y, a continuación, seguir los pasos que se indican en esta página para agregar políticas de permisos a su rol de ejecución.

- Si elige la opción Crear un nuevo rol en la consola, Amazon MWAA asociará a su rol de ejecución los permisos mínimos necesarios para un entorno.
- En algunos casos, Amazon MWAA asociará los permisos máximos. Por ejemplo, le recomendamos que elija esta opción en la consola de Amazon MWAA si al crear un entorno desea crear un rol de ejecución. Amazon MWAA añadirá automáticamente las políticas de permisos para todos los grupos de registros de CloudWatch utilizando el patrón de expresiones regulares en el rol de ejecución "`arn:aws:logs:`*`us-east-1`*`:`*`111122223333`*`:log-group:`*`airflow-your-environment-name-*`*"".

Cómo añadir permisos para poder utilizar otros servicios de AWS

Amazon MWAA no puede añadir ni editar políticas de permisos para un rol de ejecución existente una vez creado el entorno. Por ello, deberá actualizar su rol de ejecución con las políticas de permisos adicionales que necesite su entorno. Por ejemplo, si su DAG precisa obtener acceso a AWS Glue, Amazon MWAA no podrá detectar de forma automática que su entorno necesita estos permisos ni añadirlos a su rol de ejecución.

Puede añadir permisos a un rol de ejecución de dos maneras:

- Modificando la política JSON insertada del rol de ejecución. Puede utilizar los ejemplos de documentos de política de JSON de esta página para añadir o sustituir la política JSON de rol de ejecución en la consola de IAM.

- Creando una política JSON para un servicio de AWS y asociándola a su rol de ejecución. Siga los pasos que se indican en esta página para asociar un nuevo documento de política JSON para un servicio de AWS a su rol de ejecución en la consola de IAM.

En caso de que el rol de ejecución ya esté asociado a su entorno, Amazon MWAA podrá comenzar a utilizar las políticas de permisos añadidas de forma inmediata. Del mismo modo, sus DAG podrían fallar si elimina cualquier permiso necesario de un rol de ejecución.

Cómo asociar un nuevo rol de ejecución

Se puede cambiar el rol de ejecución de un entorno en cualquier momento. Si aún no ha asociado un nuevo rol de ejecución a su entorno, siga los pasos que se indican en esta página para crear una política para el nuevo rol de ejecución y asociarlo a su entorno.

Creación de un nuevo rol

Amazon MWAA crea una clave propiedad de AWS para el cifrado de datos y un rol de ejecución en su nombre de forma predeterminada. Al crear un entorno, podrá elegir entre las opciones predeterminadas de la consola de Amazon MWAA. En la siguiente imagen, se muestra la opción predeterminada que debe escogerse para crear un rol de ejecución para un entorno.

> **Important**
>
> Al crear un nuevo rol de ejecución, no vuelva a usar el nombre de un rol de ejecución eliminado. Al elegir nombres únicos, se pueden evitar conflictos y garantizar una administración adecuada de los recursos.

Consulta y actualización de la política de un rol de ejecución

Puede ver el rol de ejecución de su entorno en la consola de Amazon MWAA y actualizar la política JSON del rol en la consola de IAM.

Pasos para asociar la política a un rol de ejecución

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija el rol de ejecución en el panel Permisos para abrir la página de permisos en IAM.

4. Elija el nombre del rol de ejecución para abrir la política de permisos.
5. Elija Editar política.
6. Seleccione la pestaña JSON.
7. Actualice su política JSON.
8. Elija Revisar política.
9. Elija Guardar cambios.

Cómo adjuntar una política JSON para utilizar otros servicios de AWS

Puede crear una política JSON para un servicio de AWS y asociarla a su rol de ejecución. Por ejemplo, puede asociar la siguiente política JSON para conceder acceso de solo lectura a todos los recursos de AWS Secrets Manager.

JSON

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

Pasos para asociar la política a un rol de ejecución

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.

3. Elija su rol de ejecución en el panel Permisos.
4. Seleccione Asociar políticas.
5. Elija Crear política.
6. Elija JSON.
7. Pegue la política JSON.
8. Elija Siguiente: Etiquetas y Siguiente: Consultar.
9. Indique un nombre descriptivo (por ejemplo, “`SecretsManagerReadPolicy`”) y una descripción para la política.
10. Elija Crear política.

Cómo conceder acceso al bucket de Amazon S3 con un bloqueo de acceso público a nivel de cuenta

Es posible que quiera bloquear el acceso a todos los buckets de su cuenta mediante la operación `PutPublicAccessBlock` en Amazon S3. Al bloquear el acceso a todos los buckets de su cuenta, el rol de ejecución de su entorno debe incluir la acción `s3:GetAccountPublicAccessBlock` en una política de permisos.

El ejemplo siguiente muestra la política que debe asociar a su rol de ejecución si bloquea el acceso a todos los buckets de Amazon S3 de su cuenta.

JSON

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

Para obtener más información sobre cómo restringir el acceso a los buckets de Amazon S3, consulte cómo bloquear el acceso público a su almacenamiento de Amazon S3 en la guía del usuario de Amazon Simple Storage Service.

Uso de las conexiones de Apache Airflow

También puede crear una conexión de Apache Airflow y especificar su rol de ejecución y su ARN en el objeto de conexión de Apache Airflow. Consulte Administración de las conexiones a Apache Airflow para obtener más información.

Ejemplos de políticas JSON para un rol de ejecución

En esta sección, se muestran dos ejemplos de políticas de permisos que puede usar para sustituir la que emplea para su rol de ejecución actual o bien para crear un nuevo rol de ejecución y usarlas para su entorno. Estas políticas contienen marcadores de posición del ARN de recursos para los grupos de registro de Apache Airflow, un bucket de Amazon S3 y un entorno de Amazon MWAA.

Recomendamos copiar la política de ejemplo, sustituir los ARN o marcadores de posición y, a continuación, utilizar la política JSON para crear o actualizar un rol de ejecución.

Ejemplo de política para una clave administrada por el cliente

En el siguiente ejemplo, se muestra la política de un rol de ejecución que puede usar para una clave administrada por el cliente.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": "s3:ListAllMyBuckets",
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
```

```
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
   ],
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
   ],
   "Resource": [
    "arn:aws:logs:us-east-1:111122223333: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:us-east-1:*:airflow-celery-*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "kms:Decrypt",
     "kms:DescribeKey",
     "kms:GenerateDataKey*",
     "kms:Encrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:111122223333:key/your-kms-cmk-id",
    "Condition": {
     "StringLike": {
      "kms:ViaService": [
       "sqs.us-east-1.amazonaws.com",
       "s3.us-east-1.amazonaws.com"
      ]
     }
    }
   }
  ]
}
```

A continuación, deberá permitir que Amazon MWAA asuma este rol para que pueda llevar a cabo acciones en su nombre. Para ello, puede agregar las entidades principales de servicio `"airflow.amazonaws.com"` y `"airflow-env.amazonaws.com"` a la lista de entidades de confianza para este rol de ejecución mediante la consola de IAM, o bien, incluir estas entidades principales de servicio en el documento de política de asunción de roles correspondiente a este

rol de ejecución por medio del comando create-role de IAM y mediante la AWS CLI. Consulte el siguiente ejemplo de documento de política de asunción de roles:

JSON

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

A continuación, asocie la siguiente política JSON a su clave administrada por el cliente. Esta política utiliza el prefijo "kms:EncryptionContext" en la clave de condición para permitir el acceso al grupo de registros de Apache Airflow en los Registros de CloudWatch.

```
{
  "Sid": "Allow logs access",
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.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-east-1:111122223333:*"
    }
  }
```

```
}
```

Ejemplo de política para una clave propiedad de AWS

En el siguiente ejemplo, se muestra la política de un rol de ejecución que puede usar para una clave propiedad de AWS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
{your-environment-name}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
```

```
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333: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:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Siguientes pasos

- Para más información acerca de los permisos que usted y sus usuarios de Apache Airflow deben tener para acceder a su entorno consulte Acceso a un entorno de Amazon MWAA.
- Información sobre Uso de claves maestras de cliente para el cifrado.
- Lea más ejemplos de políticas administradas por el cliente.

Prevención de la sustitución confusa entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema de la sustitución confusa. La suplantación entre servicios puedes producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). Se puede manipular el servicio que realiza las llamadas para usar los permisos para actuar en función de los recursos de otro cliente, de modo que no requiera permiso para acceder. Para evitarlo, AWS proporciona

herramientas que le ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Se recomienda usar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en el entorno de ejecución de su rol para limitar los permisos que Amazon MWAA concede a otros servicios para que accedan al recurso. Utiliza `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utiliza `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:airflow:*:123456789012:environment/*`.

El valor de `aws:SourceArn` debe ser el ARN de su entorno de Amazon MWAA, para el cual está creando una rol de ejecución.

En el siguiente ejemplo, se muestra cómo se pueden usar las claves de condición de contexto global `aws:SourceArn` y `aws:SourceAccount` en el rol de ejecución de la política de confianza de su entorno para evitar el problema del suplente confuso. Puede utilizar la siguiente política de confianza al crear un nuevo rol de ejecución.

JSON

```
{
    "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:us-
east-1:123456789012:environment/your-environment-name"
```

```
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Modos de acceso de Apache Airflow

La consola de Amazon Managed Workflows para Apache Airflow tiene opciones integradas para configurar el enrutamiento público o privado al servidor web de Apache Airflow de su entorno. En esta guía, se describen los modos de acceso disponibles para el servidor web de Apache Airflow en su entorno de Amazon Managed Workflows para Apache Airflow y los recursos adicionales que deberá configurar en su VPC de Amazon si elige la opción de red privada.

Contenido

Modos de acceso de Apache Airflow

Puede elegir un enrutamiento público o privado para su servidor web de Apache Airflow. Para habilitar el enrutamiento privado, elija red privada. De esta forma, los usuarios solo pueden acceder a un servidor web de Apache Airflow desde una VPC de Amazon. Para habilitar el enrutamiento

público, elija red pública. Esto permite a los usuarios acceder al servidor web de Apache Airflow a través de Internet.

Red pública

En el siguiente diagrama de arquitectura, se muestra un entorno de Amazon MWAA con un servidor web público.

El modo de acceso mediante red pública permite que los usuarios a los que se les haya otorgado acceso a la política de IAM de su entorno accedan a la UI de Apache Airflow a través de Internet.

En la siguiente imagen, se muestra dónde se encuentra la opción de red pública en la consola de Amazon MWAA.

Red privada

En el siguiente diagrama de arquitectura, se muestra un entorno de Amazon MWAA con un servidor web privado.

El modo de acceso de red privada limita el acceso a la interfaz de usuario de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno.

Al crear un entorno con acceso mediante red privada al servidor web, debe empaquetar todas sus dependencias en un archivo wheel de Python (`.whl`) y luego hacer referencia al `.whl` en su `requirements.txt`. Para obtener instrucciones sobre cómo empaquetar e instalar sus dependencias mediante el archivo wheel, consulte cómo administrar dependencias con archivos wheel de Python.

En la siguiente imagen, se muestra dónde se encuentra la opción de red privada en la consola de Amazon MWAA.

Información general sobre los modos de acceso

En esta sección se describen los puntos de conexión de VPC (AWS PrivateLink) que se crean en su VPC de Amazon al elegir el modo de acceso mediante red pública o mediante red privada.

Modo de acceso mediante red pública

Si elige el modo de acceso de red pública para su servidor web de Apache Airflow, el tráfico de la red se enruta públicamente a través de Internet.

- Amazon MWAA crea un punto de conexión de la interfaz de la VPC para su base de datos de metadatos de Amazon Aurora PostgreSQL. El punto de conexión se crea en las zonas de disponibilidad correspondientes a sus subredes privadas y es independiente de las demás Cuentas de AWS.
- A continuación, Amazon MWAA vincula una dirección IP de sus subredes privadas a los puntos de conexión de la interfaz. Se ha diseñado de esta manera siguiendo la práctica recomendada de vincular una sola IP de cada zona de disponibilidad de la VPC de Amazon.

Modo de acceso mediante red privada

Si elige el modo de acceso de red privada para su servidor web de Apache Airflow, el tráfico de red se enruta de forma privada dentro de su Amazon VPC.

- Amazon MWAA crea un punto de conexión de la interfaz de la VPC para el servidor web de Apache Airflow y un punto de conexión de la interfaz para su base de datos de metadatos de Amazon Aurora PostgreSQL. Los puntos de conexión se crean en las zonas de disponibilidad correspondientes a sus subredes privadas y son independientes del resto de Cuentas de AWS.
- A continuación, Amazon MWAA vincula una dirección IP de sus subredes privadas a los puntos de conexión de la interfaz. Se ha diseñado de esta manera siguiendo la práctica recomendada de vincular una sola IP de cada zona de disponibilidad de la VPC de Amazon.

Consulte the section called “Ejemplos de casos de uso para un modo de acceso a Amazon VPC y Apache Airflow” para obtener más información.

Configuración para los modos de acceso mediante red pública y mediante red privada

En la siguiente sección se describen los ajustes y configuraciones adicionales que deberá realizar en función del modo de acceso de Apache Airflow que haya elegido para su entorno.

Configuración para la red pública

Si elige la opción de red pública para el servidor web de Apache Airflow, podrá empezar a usar la UI de Apache Airflow en cuanto haya creado su entorno.

Deberá seguir los pasos que se describen a continuación para configurar el acceso de sus usuarios y los permisos para que su entorno pueda utilizar otros servicios de AWS.

1. Añada permisos. Amazon MWAA necesita permiso para utilizar otros servicios de AWS. Cuando se crea un entorno, Amazon MWAA crea un rol vinculado a servicios que le permite utilizar determinadas acciones de IAM para Amazon Elastic Container Registry (Amazon ECR), los registros de CloudWatch y Amazon EC2.

   Puede añadir permisos para que estos u otros servicios de AWS realicen acciones adicionales añadiendo permisos a su rol de ejecución. Consulte Rol de ejecución de Amazon MWAA para obtener más información.
2. Cree políticas de usuario. Es posible que deba crear varias políticas de IAM para que sus usuarios puedan configurar el acceso a su entorno y a la UI de Apache Airflow. Consulte Acceso a un entorno de Amazon MWAA para obtener más información.

Configuración para la red privada

Si elige la opción de red privada para el servidor web de Apache Airflow, tendrá que configurar el acceso de sus usuarios, permitir que su entorno use otros servicios de AWS y crear un mecanismo para acceder a los recursos de su VPC de Amazon desde su computadora.

1. Añada permisos. Amazon MWAA necesita permiso para utilizar otros servicios de AWS. Cuando se crea un entorno, Amazon MWAA crea un rol vinculado a servicios que le permite utilizar determinadas acciones de IAM para Amazon Elastic Container Registry (Amazon ECR), los registros de CloudWatch y Amazon EC2.

   Puede añadir permisos para que estos u otros servicios de AWS realicen acciones adicionales añadiendo permisos a su rol de ejecución. Consulte Rol de ejecución de Amazon MWAA para obtener más información.
2. Cree políticas de usuario. Es posible que deba crear varias políticas de IAM para que sus usuarios puedan configurar el acceso a su entorno y a la UI de Apache Airflow. Consulte Acceso a un entorno de Amazon MWAA para obtener más información.

3. Habilite el acceso a la red. Deberá crear un mecanismo en su VPC de Amazon para conectarse al punto de conexión de VPC (AWS PrivateLink) del servidor web de Apache Airflow. Por ejemplo, puede crear un túnel de VPN desde su ordenador mediante una AWS Client VPN.

Acceso al punto de conexión de VPC del servidor web de Apache Airflow (acceso mediante red privada)

Si elige la opción de red privada, tendrá que crear un mecanismo en su VPC de Amazon para poder acceder al punto de conexión de VPC (AWS PrivateLink) del servidor web de Apache Airflow. Recomendamos utilizar la misma VPC de Amazon, el mismo grupo de seguridad de VPC y las mismas subredes privadas que utiliza su entorno de Amazon MWAA para estos recursos.

Para obtener más información, consulte cómo administrar accesos a los puntos de conexión de la VPC.

Acceso a Apache Airflow

Amazon MWAA le permite acceder a su entorno de Apache Airflow mediante varios métodos: la consola de interfaz de usuario (UI) de Apache Airflow, la CLI de Apache Airflow y la API de REST de Apache Airflow. Puede usar la consola de Amazon MWAA para ver e invocar un DAG en la UI de Apache Airflow o usar las API de Amazon MWAA para obtener un token e invocar un DAG. En esta sección se describen los permisos necesarios para acceder a la interfaz de usuario de Apache Airflow, cómo generar un token para realizar llamadas a la API de Amazon MWAA directamente en el shell de comandos y los comandos compatibles en la CLI de Apache Airflow.

Temas

Requisitos previos

En la siguiente sección se describen los pasos preliminares necesarios para utilizar los comandos y scripts de esta sección.

Acceso

- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos de Amazon MWAA en Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess.
- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos Política de acceso completo a la consola y a las API: AmazonMWAAFullApiAccess de Amazon MWAA.

AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Abrir la interfaz de usuario de Apache Airflow

En la siguiente imagen, se muestra el enlace a la UI de Apache Airflow en la consola de Amazon MWAA.

Inicie sesión en Apache Airflow

Necesita permisos de Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess para su cuenta de Cuenta de AWS en AWS Identity and Access Management (IAM) para ver la UI de Apache Airflow.

Pasos para acceder a la interfaz de usuario de Apache Airflow

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Abrir interfaz de usuario de Airflow.

Cree un token de acceso al servidor web Apache Airflow

Puede usar los comandos de esta página para crear un token de acceso al servidor web. Un token de acceso le permite acceder a su entorno de Amazon MWAA. Por ejemplo, puede obtener un token y, a continuación, implementar los DAG mediante programación con las API de Amazon MWAA. La siguiente incluye sección los pasos para crear un token de inicio de sesión web de Apache Airflow mediante la AWS CLI, un script de bash, una solicitud de API POST o un script Python. El token devuelto en la respuesta es válido durante 60 segundos.

> **Important**
>
> Desde el 19 de agosto de 2025, Amazon MWAA agregó soporte para puntos de conexión IPv6 y ahora es compatible con puntos de conexión IPv4 e IPv6. A partir de esa fecha, todos los entornos recién creados usarán dominios `.on.aws` para la interfaz de usuario (UI) de Airflow. Los clientes deben migrar su UI de Airflow de dominios `.amazonaws.com` a `.on.aws` para estos entornos recién creados. Los servicios de punto de conexión de Virtual Private Cloud (VPC) para servidores web y bases de datos mantendrán sus dominios `.amazonaws.com` actuales sin necesidad de hacer cambios.

Contenido

Requisitos previos

En la siguiente sección se describen los pasos preliminares necesarios para utilizar los comandos y scripts de esta página.

Acceso

- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos de Amazon MWAA en Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess.
- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos Política de acceso completo a la consola y a las API: AmazonMWAAFullApiAccess de Amazon MWAA.

AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Uso de AWS CLI

El siguiente ejemplo usa el comando create-web-login-token en la AWS CLI para crear un token de inicio de sesión web de Apache Airflow.

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

Uso de un script de bash

El siguiente ejemplo usa el script bash para llamar al comando create-web-login-token en la AWS CLI para crear un token de inicio de sesión web de Apache Airflow.

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Sustituya los marcadores de posición en *rojo* por YOUR_HOST_NAME y YOUR_ENVIRONMENT_NAME. Por ejemplo, el nombre de host de una red pública tiene este aspecto (sin el https://):

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

3. (Opcional) Es posible que los usuarios de macOS y Linux tengan que ejecutar el siguiente comando para verificar que el script sea ejecutable.

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

4. Ejecute el siguiente script para obtener un token de inicio de sesión web.

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

La línea de comandos muestra lo siguiente:

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

Uso de un script de Python

El siguiente ejemplo utiliza el método boto3 create_web_login_token en un script de Python para crear un token de inicio de sesión web de Apache Airflow. Puede ejecutar este script fuera de Amazon MWAA. Para ello, solo tiene que instalar la biblioteca boto3. Es posible que deba crear un entorno virtual para instalar la biblioteca. Se supone que ha configurado las credenciales de autenticación de AWS para su cuenta.

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Sustituya el marcador de posición en *rojo* por YOUR_ENVIRONMENT_NAME.
3. Ejecute el siguiente script para obtener un token de inicio de sesión web.

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

Siguientes pasos

- Explore la operación de la API de Amazon MWAA utilizada para crear un token de inicio de sesión web en CreateWebLoginToken.

Configuración de un dominio personalizado para el servidor web Apache Airflow

Amazon Managed Workflows para Apache Airflow (Amazon MWAA) le permite configurar un dominio personalizado para el servidor web Apache Airflow administrado. Con un dominio personalizado, puede acceder al servidor web Apache Airflow administrado por Amazon MWAA de su entorno mediante la UI de Apache Airflow, la CLI de Apache Airflow o el servidor web Apache Airflow.

> **Note**
>
> Solo puede usar un dominio personalizado con un servidor web privado sin acceso a Internet.

Casos de uso de un dominio personalizado en Amazon MWAA

1. Comparta el dominio del servidor web con una aplicación en la nube en AWS: el uso de un dominio personalizado le permite definir una URL fácil de usar para acceder al servidor web, en lugar del nombre de dominio de servicio generado. Puede almacenar este dominio personalizado y compartirlo como una variable de entorno en sus aplicaciones.
2. Acceda a un servidor web privado: si quiere configurar el acceso a un servidor web en una VPC sin acceso a Internet, el uso de un dominio personalizado simplifica el flujo de trabajo de redireccionamiento de la URL.

Temas

- Configuración del dominio personalizado
- Configuración de la infraestructura de red

Configuración del dominio personalizado

Para configurar la característica de dominio personalizado, debe proporcionar el valor de dominio personalizado mediante la configuración `webserver.base_url` de Apache Airflow cuando crea o actualiza el entorno Amazon MWAA. Las siguientes limitaciones se aplican a su nombre de dominio personalizado:

- El valor debe ser un nombre de dominio completo (FQDN) sin protocolo o ruta. Por ejemplo, `your-custom-domain.com`.
- Amazon MWAA no permite ninguna ruta en la URL. Por ejemplo, `your-custom-domain.com/dags/` no es un nombre de dominio personalizado válido.
- La longitud de la URL está limitada a 255 caracteres ASCII.
- Si proporciona una cadena vacía, de forma predeterminada, el entorno se creará con una URL de servidor web generada por Amazon MWAA.

En el siguiente ejemplo, se muestra el uso de AWS CLI para crear un entorno con un nombre de dominio de servidor web personalizado.

```
aws mwaa create-environment \
--name my-mwaa-env \
--source-bucket-arn arn:aws:s3:::amzn-s3-demo-bucket \
--airflow-configuration-options '{"webserver.base_url":"my-custom-domain.com"}' \
--network-configuration '{"SubnetIds":["subnet-0123456789abcdef","subnet-
fedcba9876543210"]}' \
--execution-role-arn arn:aws:iam::123456789012:role/my-execution-role
```

Una vez que se crea o actualiza el entorno, debe configurar la infraestructura de red en su Cuenta de AWS para acceder al servidor web privado a través del dominio personalizado.

Para volver a la URL predeterminada generada por el servicio, actualice su entorno privado y elimine la opción de configuración `webserver.base_url`.

Configuración de la infraestructura de red

Siga los siguientes pasos para configurar la infraestructura de red necesaria para usarla con el dominio personalizado en su Cuenta de AWS.

1. Obtenga las direcciones IP de las interfaces de red de puntos de conexión (ENI) de Amazon VPC. Para ello, utilice primero `get-environment` para buscar el `WebserverVpcEndpointService` adecuado para su entorno.

```
aws mwaa get-environment --name your-environment-name
```

Si la operación se realiza correctamente, verá un resultado similar al siguiente.

```
{
  "Environment": {
    "AirflowConfigurationOptions": {},
    "AirflowVersion": "latest-version",
    "Arn": "environment-arn",
    "CreatedAt": "2024-06-01T01:00:00-00:00",
    "DagS3Path": "dags",
    .
    .
    .
    "WebserverVpcEndpointService": "web-server-vpc-endpoint-service",
    "WeeklyMaintenanceWindowStart": "TUE:21:30"
  }
}
```

Tome nota del valor `WebserverVpcEndpointService` y utilícelo para `web-server-vpc-endpoint-service` en el siguiente comando `describe-vpc-endpoints` de Amazon EC2. Use `--filters Name=service-name,Values=web-server-vpc-endpoint-service-id` en el siguiente comando.

2. Recupere los detalles del punto de conexión de Amazon VPC. Este comando obtiene detalles sobre los puntos de conexión de Amazon VPC que coinciden con un nombre de servicio específico, y devuelve el ID del punto de conexión y los ID de las interfaces de red asociadas en formato de texto.

```
aws ec2 describe-vpc-endpoints \
 --filters Name=service-name,Values=web-server-vpc-endpoint-service \
 --query 'VpcEndpoints[*].
{EndpointId:VpcEndpointId,NetworkInterfaceIds:NetworkInterfaceIds}' \
 --output text
```

3. Obtenga los detalles de la interfaz de red. Este comando recupera las direcciones IP privadas de cada interfaz de red asociada a los puntos de conexión de Amazon VPC identificados en el paso anterior.

```
for eni_id in $(
  aws ec2 describe-vpc-endpoints \
   --filters Name=service-name,Values=service-id \
   --query 'VpcEndpoints[*].NetworkInterfaceIds' \
   --output text
 ); do
 aws ec2 describe-network-interfaces \
  --network-interface-ids $eni_id \
  --query 'NetworkInterfaces[*].PrivateIpAddresses[*].PrivateIpAddress' \
  --output text
      done
```

4. Utilice `create-target-group` para crear un grupo de destino. Debe usar este grupo objetivo para registrar las direcciones IP para los puntos de conexión de Amazon VPC de su servidor web.

```
aws elbv2 create-target-group \
--name new-target-group-namne \
--protocol HTTPS \
--port 443 \
--vpc-id web-server-vpc-id \
--target-type ip \
--health-check-protocol HTTPS \
--health-check-port 443 \
--health-check-path / \
--health-check-enabled \
--matcher 'HttpCode="200,302"'
```

   Registre las direcciones IP mediante el comando `register-targets`.

```
aws elbv2 register-targets \
--target-group-arn target-group-arn \
--targets Id=ip-address-1 Id=ip-address-2
```

5. Solicite un certificado de ACM. Omita este paso si utiliza un certificado existente.

```
aws acm request-certificate \
--domain-name my-custom-domain.com \
```

```
--validation-method DNS
```

6. Configure un Application Load Balancer. En primer lugar, cree el equilibrador de carga y, a continuación, cree un oyente para el equilibrador de carga. Especifique el certificado de ACM que creó en el paso anterior.

```
aws elbv2 create-load-balancer \
--name my-mwaa-lb \
--type application \
--subnets subnet-id-1 subnet-id-2
```

```
aws elbv2 create-listener \
--load-balancer-arn load-balancer-arn \
--protocol HTTPS \
--port 443 \
--ssl-policy ELBSecurityPolicy-2016-08 \
--certificates CertificateArn=acm-certificate-arn \
--default-actions Type=forward,TargetGroupArn=target-group-arn
```

   Si usa un equilibrador de carga de red en una subred privada, configure un host bastión o un túnel de Site-to-Site VPN para acceder al servidor web.

7. Cree una zona alojada mediante Route 53 para el dominio.

```
aws route53 create-hosted-zone --name my-custom-domain.com \
--caller-reference 1
```

   Cree un registro A para el dominio. Para hacer esto mediante el uso de la AWS CLI, obtenga el ID de la zona alojada mediante `list-hosted-zones-by-name`, luego aplique el registro con `change-resource-record-sets`.

```
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
--dns-name my-custom-domain.com \
--query 'HostedZones[0].Id' --output text)
```

```
aws route53 change-resource-record-sets \
--hosted-zone-id $HOSTED_ZONE_ID \
--change-batch '{
  "Changes": [
    {
```

```
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "my-custom-domain.com",
        "Type": "A",
        "AliasTarget": {
          "HostedZoneId": "load-balancer-hosted-zone-id>",
          "DNSName": "load-balancer-dns-name",
          "EvaluateTargetHealth": true
        }
      }
    }
  ]
}'
```

8. Actualice las reglas del grupo de seguridad para el punto de conexión de Amazon VPC del servidor web para seguir el principio de privilegio mínimo. De ese modo, se permite tráfico HTTPS solo desde las subredes públicas donde se encuentra el equilibrador de carga de aplicación. Guarde el siguiente JSON de forma local. Por ejemplo, como `sg-ingress-ip-permissions.json`.

```
[
  {
    "IpProtocol": "tcp",
    "FromPort": 443,
    "ToPort": 443,
    "UserIdGroupPairs": [
      {
        "GroupId": "load-balancer-security-group-id"
      }
    ],
    "IpRanges": [
      {
        "CidrIp": "public-subnet-1-cidr"
      },
      {
        "CidrIp": "public-subnet-2-cidr"
      }
    ]
  }
]
```

Ejecute el siguiente comando de Amazon EC2 para actualizar las reglas del grupo de seguridad de entrada. Especifique el archivo JSON para `--ip-permissions`.

```
aws ec2 authorize-security-group-ingress \
--group-id <security-group-id> \
--ip-permissions file://sg-ingress-ip-permissions.json
```

Ejecute el siguiente comando de Amazon EC2 para actualizar las reglas de salida.

```
aws ec2 authorize-security-group-egress \
--group-id webserver-vpc-endpoint-security-group-id \
--protocol tcp \
--port 443 \
--source-group load-balancer-security-group-id
```

Abra la consola de Amazon MWAA y navegue hasta la interfaz de usuario de Apache Airflow. Si va a configurar un equilibrador de carga de red en una subred privada en lugar del equilibrador de carga de aplicación que se usa aquí, debe acceder al servidor web con una de las siguientes opciones.

- the section called “Tutorial: host bastión de Linux”
- the section called “Tutorial: AWS Client VPN”

Creación de un token de la CLI de Apache Airflow

> **Tip**
>
> La API REST es más moderna que la CLI y está diseñada para la integración programática con sistemas externos. REST es la forma preferida de interactuar con Apache Airflow.

Puede usar los comandos de esta página para generar un token de para la CLI y, a continuación, realizar llamadas a la API de Amazon Managed Workflows para Apache Airflow directamente en el shell de comandos. Por ejemplo, puede obtener un token y, a continuación, implementar los DAG mediante programación con las API de Amazon MWAA. La sección siguiente detalla los pasos para crear un token de la CLI de Apache Airflow mediante la AWS CLI, un script de cURL, de Python o de bash. El token devuelto en la respuesta es válido durante 60 segundos.

El token de la AWS CLI pretende sustituir a las acciones sincrónicas del shell, no a los comandos de API asíncronos. Por lo tanto, la simultaneidad disponible es limitada. Para garantizar que el servidor web responda a los usuarios, se recomienda no abrir solicitudes nuevas de la AWS CLI hasta que la anterior se complete correctamente.

Contenido

Requisitos previos

En la siguiente sección se describen los pasos preliminares necesarios para utilizar los comandos y scripts de esta página.

Acceso

- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos de Amazon MWAA en Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess.
- Acceso de la Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos Política de acceso completo a la consola y a las API: AmazonMWAAFullApiAccess de Amazon MWAA.

AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.

- AWS CLI: configuración rápida con `aws configure`.

Uso de AWS CLI

El ejemplo que sigue usa el comando create-cli-token en la AWS CLI para crear un token de la CLI de Apache Airflow.

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

Uso de un script de cURL

En el siguiente ejemplo, se usa un script de cURL para llamar al comando create-web-login-token en la AWS CLI para invocar la CLI de Apache Airflow a través de un punto de conexión del servidor web de Apache Airflow.

Apache Airflow v3

1. Copie la instrucción de cURL del archivo de texto y péguela en el shell de comandos.

> **Note**
>
> Tras copiarla en el portapapeles, puede que tenga que usar Edit > Paste (Editar > Pegar) en el menú del shell.

```
CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
&& CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
&& WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
&& CLI_RESULTS=$(curl -L --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 --logical-date $(date -u +"%Y-%m-%dT%H:
%M:%SZ")") \
&& echo "Output:" \
&& echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
&& echo "Errors:" \
&& echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

2. Sustituya los marcadores de posición *rojos* por la región de su entorno, *YOUR_DAG_NAME* y *YOUR_ENVIRONMENT_NAME*. Por ejemplo, el nombre de host de una red pública tiene este aspecto (sin el https://):

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

La línea de comandos muestra lo siguiente:

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

Apache Airflow v2

1. Copie la instrucción de cURL del archivo de texto y péguela en el shell de comandos.

> **Note**
>
> Tras copiarla en el portapapeles, puede que tenga que usar Edit > Paste (Editar > Pegar) en el menú del shell.

```
CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
&& CLI_TOKEN=$(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. Sustituya los marcadores de posición *rojos* por la región de su entorno, YOUR_DAG_NAME y YOUR_ENVIRONMENT_NAME. Por ejemplo, el nombre de host de una red pública tiene este aspecto (sin el https://):

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

La línea de comandos muestra lo siguiente:

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

Uso de un script de bash

El siguiente ejemplo usa un script de bash para llamar al comando create-cli-token en la AWS CLI para crear un token CLI de Apache Airflow.

Apache Airflow v3

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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 -L --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 --logical-date $(date -u +"%Y-%m-
%dT%H:%M:%SZ")"
```

2. Sustituya los marcadores de posición en *rojo* por YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME y YOUR_DAG_NAME. Por ejemplo, el nombre de host de una red pública tiene este aspecto (sin el https://):

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

3. (Opcional) Es posible que los usuarios de macOS y Linux tengan que ejecutar el siguiente comando para verificar que el script sea ejecutable.

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

4. Ejecute el script siguiente para crear un token de la CLI de Apache Airflow.

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

Apache Airflow v2

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Sustituya los marcadores de posición en *rojo* por YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME y YOUR_DAG_NAME. Por ejemplo, el nombre de host de una red pública tiene este aspecto (sin el https://):

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

3. (Opcional) Es posible que los usuarios de macOS y Linux tengan que ejecutar el siguiente comando para verificar que el script sea ejecutable.

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

4. Ejecute el script siguiente para crear un token de la CLI de Apache Airflow.

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

Uso de un script de Python

El siguiente ejemplo utiliza el método boto3 create_cli_token en un script de Python para crear un token de la CLI de Apache Airflow y activar un DAG. Puede ejecutar este script fuera de Amazon MWAA. Para ello, solo tiene que instalar la biblioteca boto3. Es posible que deba crear un entorno virtual para instalar la biblioteca. Se supone que ha configurado las credenciales de autenticación de AWS para su cuenta.

Apache Airflow v3

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Sustituya los marcadores de posición por `YOUR_ENVIRONMENT_NAME` y `YOUR_DAG_NAME`.
3. Ejecute el script siguiente para crear un token de la CLI de Apache Airflow.

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

Apache Airflow v2

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Sustituya los marcadores de posición por `YOUR_ENVIRONMENT_NAME` y `YOUR_DAG_NAME`.
3. Ejecute el script siguiente para crear un token de la CLI de Apache Airflow.

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

Siguientes pasos

- Analice la operación de la API de Amazon MWAA utilizada para crear un token de la CLI de en CreateCliToken.

Uso de la API de REST de Apache Airflow

Amazon Managed Workflows para Apache Airflow (Amazon MWAA) permite interactuar directamente con los entornos de Apache Airflow mediante la API de REST de Apache Airflow para entornos que ejecutan Apache Airflow v2.4.3 y versiones posteriores. Esto le permite acceder a sus entornos de Amazon MWAA y administrarlos mediante programación, lo que proporciona una forma estandarizada de invocar flujos de trabajo de orquestación de datos, gestionar sus DAG y supervisar el estado de varios componentes de Apache Airflow, como la base de datos de metadatos, el activador y el programador.

Para tener escalabilidad cuando se usa la API de REST de Apache Airflow, Amazon MWAA ofrece la opción de escalar horizontalmente la capacidad del servidor web para gestionar el aumento de la demanda, ya sea de solicitudes de API de REST, uso de la interfaz de la línea de comandos (CLI) o más usuarios simultáneos de la interfaz de usuario (UI) de Apache Airflow. Para obtener más información sobre cómo Amazon MWAA escala los servidores web, consulte the section called “Configuración del escalado automático del servidor web”.

Puede usar la API de REST de Apache Airflow para implementar los siguientes casos de uso en sus entornos:

- Acceso mediante programación: ahora puede iniciar las ejecuciones del DAG de Apache Airflow, administrar conjuntos de datos y recuperar el estado de varios componentes, como la base de

datos de metadatos, los activadores y los programadores, sin depender de la interfaz de usuario o la CLI de Apache Airflow.

- Integración con aplicaciones y microservicios externos: la compatibilidad con la API de REST permite crear soluciones personalizadas que integran sus entornos de Amazon MWAA con otros sistemas. Por ejemplo, puede iniciar flujos de trabajo en respuesta a eventos de sistemas externos, como trabajos de base de datos completados o registros de nuevos usuarios.
- Supervisión centralizada: puede crear paneles de supervisión que agreguen el estado de sus DAG en varios entornos de Amazon MWAA, lo que permite una supervisión y una administración centralizadas.

Para obtener más información sobre la API de REST de Apache Airflow, consulte la referencia de la API de REST de Apache Airflow.

Usando `InvokeRestApi`, puede acceder a la API de REST de Apache Airflow con credenciales de AWS. Como alternativa, también puede acceder a ella si obtiene un token de acceso al servidor web y lo usa para llamarlo.

Si obtiene un error con el mensaje `Update your environment to use InvokeRestApi` mientras usa la operación `InvokeRestApi`, significa que necesita actualizar su entorno de Amazon MWAA. Este error se produce cuando el entorno de Amazon MWAA no es compatible con los cambios más recientes relacionados con la característica `InvokeRestApi`. Para resolver este problema, actualice el entorno de Amazon MWAA para incorporar los cambios necesarios en la característica `InvokeRestApi`.

El tiempo de espera predeterminado para la operación `InvokeRestApi` de 10 segundos. Si la operación no se completa dentro de ese lapso, finaliza automáticamente y se produce un error. Procure que las llamadas a la API de REST estén diseñadas para completarse dentro de este período de espera para evitar errores.

Para tener escalabilidad cuando se usa la API de REST de Apache Airflow, Amazon MWAA ofrece la opción de escalar horizontalmente la capacidad del servidor web para gestionar el aumento de la demanda, ya sea de solicitudes de API de REST, uso de la interfaz de la línea de comandos (CLI) o más usuarios simultáneos de la interfaz de usuario (UI) de Apache Airflow. Para obtener más información sobre cómo Amazon MWAA escala los servidores web, consulte the section called “Configuración del escalado automático del servidor web”.

Puede usar la API de REST de Apache Airflow para implementar los siguientes casos de uso en sus entornos:

- Acceso mediante programación: ahora puede iniciar las ejecuciones del DAG de Apache Airflow, administrar conjuntos de datos y recuperar el estado de varios componentes, como la base de datos de metadatos, los activadores y los programadores, sin depender de la interfaz de usuario o la CLI de Apache Airflow.
- Integración con aplicaciones y microservicios externos: la compatibilidad con la API de REST permite crear soluciones personalizadas que integran sus entornos de Amazon MWAA con otros sistemas. Por ejemplo, puede iniciar flujos de trabajo en respuesta a eventos de sistemas externos, como trabajos de base de datos completados o registros de nuevos usuarios.
- Supervisión centralizada: puede crear paneles de supervisión que agreguen el estado de sus DAG en varios entornos de Amazon MWAA, lo que permite una supervisión y una administración centralizadas.

Para obtener más información sobre la API de REST de Apache Airflow, consulte la referencia de la API de REST de Apache Airflow.

Usando `InvokeRestApi`, puede acceder a la API de REST de Apache Airflow con credenciales de AWS. Como alternativa, también puede acceder a ella si obtiene un token de acceso al servidor web y, a continuación, lo utiliza para llamarlo.

- Si obtiene un error con el mensaje `Update your environment to use InvokeRestApi` mientras usa la operación `InvokeRestApi`, significa que necesita actualizar su entorno de Amazon MWAA. Este error se produce cuando el entorno de Amazon MWAA no es compatible con los cambios más recientes relacionados con la característica `InvokeRestApi`. Para resolver este problema, actualice el entorno de Amazon MWAA para incorporar los cambios necesarios en la característica `InvokeRestApi`.
- El tiempo de espera predeterminado para la operación `InvokeRestApi` de 10 segundos. Si la operación no se completa dentro de ese lapso, finaliza automáticamente y se produce un error. Procure que las llamadas a la API de REST estén diseñadas para completarse dentro de este período de espera para evitar errores.

> **Important**
>
> El tamaño de la carga útil de respuesta no puede ser superior a 6 MB. Si `RestApi` se ha superado este límite, se produce un error.

En los siguientes ejemplos, se muestra cómo realizar llamadas a la API de REST de Apache Airflow e iniciar una nueva ejecución del DAG:

Temas

Concesión de acceso a la API de REST de Apache Airflow: `airflow:InvokeRestApi`

Para acceder a la API de REST de Apache Airflow con credenciales de AWS, debe otorgar el permiso de `airflow:InvokeRestApi` en su política de IAM. En el siguiente ejemplo de política, especifique el rol de `Admin`, `Op`, `User`, `Viewer` o `Public` en `{airflow-role}` para personalizar el nivel de acceso del usuario. Para obtener más información, consulte la sección Roles predeterminados en la guía de referencia de Apache Airflow.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> Mientras configura un servidor web privado, la acción `InvokeRestApi` no se puede invocar desde fuera de una nube privada virtual (VPC). Puede utilizar la clave `aws:SourceVpc` para aplicar un control de acceso más detallado para esta operación. Para obtener más información, consulte aws:SourceVpc.

Llamado a la API de REST de Apache Airflow

En el siguiente script de ejemplo, se explica cómo utilizar la API de REST de Apache Airflow para enumerar los DAG disponibles en su entorno y cómo crear una variable de Apache Airflow:

```
import boto3

  env_name = "MyAirflowEnvironment"

  def list_dags(client):
    request_params = {
      "Name": env_name,
      "Path": "/dags",
      "Method": "GET",
      "QueryParameters": {
        "paused": False
      }
    }
  response = client.invoke_rest_api(
    **request_params
  )

  print("Airflow REST API response: ", response['RestApiResponse'])

  def create_variable(client):
    request_params = {
      "Name": env_name,
      "Path": "/variables",
      "Method": "POST",
      "Body": {
        "key": "test-restapi-key",
        "value": "test-restapi-value",
        "description": "Test variable created by MWAA InvokeRestApi API",
      }
```

```
    }
  response = client.invoke_rest_api(
    **request_params
  )

  print("Airflow REST API response: ", response['RestApiResponse'])

  if __name__ == "__main__":
    client = boto3.client("mwaa")
    list_dags(client)
    create_variable(client)
```

Creación de un token de sesión de servidor web y llamada a la API de REST de Apache Airflow

Para crear un token de acceso al servidor web, use la siguiente función de Python. Esta función primero llama a la API de Amazon MWAA para obtener un token de inicio de sesión web. El token de inicio de sesión web, que caduca después de 60 segundos, se cambia luego por un token de sesión web, que le permite acceder al servidor web y usar la API de REST de Apache Airflow. Si necesita más de 10 transacciones por segundo (TPS) de capacidad de limitación, puede usar este método para acceder a la API de REST de Apache Airflow.

El token de sesión caduca después de 12 horas.

> **Tip**
>
> A continuación, se enumeran los cambios clave en los siguientes ejemplos de código de Apache Airflow v2 a v3:
>
> - La ruta de la API de REST cambió de `/api/v1` a `/api/v2`
> - La ruta de inicio de sesión cambió de `/aws_maa/login` a `/pluginsv2/aws_mwaa/login`
> - La respuesta del inicio de sesión `response.cookies["_token"]` contiene información del token que debe usar para las siguientes llamadas a la API
> - Para una llamada a la API de REST, debe pasar información de `jwt_token` en los encabezados de la siguiente manera:
>
>   ```
>   headers = {
>     "Authorization": f"Bearer {jwt_token}",
>   ```

```
  "Content-Type": "application/json"
}
```

Apache Airflow v3

```
def get_token_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}/pluginsv2/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 successful
    if response.status_code == 200:

    # Return the hostname and the session cookie
    return (
      web_server_host_name,
      response.cookies['_token']
    )
    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
```

Apache Airflow v2

```
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
```

Una vez finalizada la autenticación, dispondrá de las credenciales para empezar a enviar solicitudes a los puntos de conexión de la API. En el ejemplo de la siguiente sección, use el punto de conexión `dags/{dag_name}/dagRuns`.

Apache Airflow v3

```
def trigger_dag(region, env_name, dag_id):
      """
      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_id (str): ID of the DAG to trigger.
      """

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

      # Retrieve the web server hostname and token for authentication
      try:
      web_server_host_name, jwt_token = get_token_info(region, env_name)
      if not jwt_token:
      logging.error("Authentication failed, no jwt token retrieved.")
      return
      except Exception as e:
      logging.error(f"Error retrieving token info: {str(e)}")
      return

      # Prepare headers and payload for the request
```

```
    request_headers = {
    "Authorization": f"Bearer {jwt_token}",
    "Content-Type": "application/json" # Good practice to include, even for GET
    }

    # sample request body input
    json_body = {"logical_date": "2025-09-17T14:15:00Z"}

    # Construct the URL for triggering the DAG
    url = f"https://{web_server_host_name}/api/v2/dags/{dag_id}/dagRuns"

    # Send the POST request to trigger the DAG
    try:
    response = requests.post(url, headers=request_headers, 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_id}")
    sys.exit(1)

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

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

Apache Airflow v2

```
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)
```

Referencia de los comandos de la CLI de Apache Airflow

En esta página, se describen los comandos de la CLI de Apache Airflow compatibles y no compatibles en Amazon Managed Workflows para Apache Airflow.

> **Tip**
>
> La REST de API es más moderna que la CLI y está diseñada para la integración programática con sistemas externos. REST es la forma preferida de interactuar con Apache Airflow.

Contenido

- Cómo agregar una configuración al activar un DAG
- Ejecución de comandos de la CLI en un túnel SSH hacia un host bastión

Requisitos previos

En la siguiente sección se describen los pasos preliminares necesarios para utilizar los comandos y scripts de esta página.

Acceso

- Acceso de la cuenta de Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos de Amazon MWAA en Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess.
- Acceso de la cuenta de Cuenta de AWS en AWS Identity and Access Management (IAM) a la política de permisos Política de acceso completo a la consola y a las API: AmazonMWAAFullApiAccess de Amazon MWAA.

AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

¿Qué ha cambiado?

- v3: arquitectura Airflow. Apache Airflow v3 introduce cambios arquitectónicos de última generación para mejorar la seguridad y la escalabilidad y facilitar el mantenimiento. Para obtener más información, consulte Actualización a Airflow 3.
- v2: estructura de comandos de la CLI de Airflow. La CLI de Apache Airflow v2 está organizada de manera que los comandos relacionados se agrupan como subcomandos, lo que significa que debe actualizar los scripts de Apache Airflow v1 si desea actualizar a Apache Airflow v2. Por ejemplo, `unpause` en Apache Airflow v1 es `dags unpause` en Apache Airflow v2. Para obtener más información, consulte los cambios de la CLI de Airflow 2.0.

Comandos de la CLI admitidos

En la siguiente sección, se enumeran los comandos de la CLI de Apache Airflow disponibles en Amazon MWAA.

Comandos admitidos

Apache Airflow v3

Versiones secundarias	Comando
v3.0.6	detalles de los activos
v3.0.6	lista de activos
v3.0.6	materializar activos
v3.0.6	crear relleno
v3.0.6	Hoja de referencia
v3.0.6	añadir conexiones
v3.0.6	eliminar conexion
v3.0.6	DAG borrados
v3.0.6	lista de DAG
v3.0.6	lista de tareas de DAG
v3.0.6	lista-importación-errores de DAG

Versiones secundarias	Comando
v3.0.6	lista-ejecuciones DAG
v3.0.6	próxima ejecución de DAG
v3.0.6	pausa de DAG
v3.0.6	informe de DAG
v3.0.6	reserializar DAG
v3.0.6	mostrar DAG
v3.0.6	estado de DAG
v3.0.6	prueba de DAG
v3.0.6	activador de DAG
v3.0.6	cancelar pausa en DAG
v3.0.6	limpiar db
v3.0.6	comportamientos de los proveedores
v3.0.6	lo que obtienen los proveedores
v3.0.6	enlaces de proveedores
v3.0.6	vínculos de proveedores

Versiones secundarias	Comando
v3.0.6	lista de proveedores
v3.0.6	notificaciones de proveedores
v3.0.6	secretos de los proveedores
v3.0.6	activador de proveedores
v3.0.6	widgets de proveedores
v3.0.6	añadir roles permanentes
v3.0.6	borrar roles permanentes
v3.0.6	creación de roles
v3.0.6	lista de roles
v3.0.6	borrar tareas
v3.0.6	tareas deps-fallidas
v3.0.6	lista de tareas
v3.0.6	representar tareas
v3.0.6	estado de las tareas

Versiones secundarias	Comando
v3.0.6	estados-ejecución-DAG de las tareas
v3.0.6	prueba de tareas
v3.0.6	eliminar variables
v3.0.6	obtener variables
v3.0.6	conjunto de variables
v3.0.6	lista de variables
v3.0.6	versión

Apache Airflow v2

Versiones secundarias	Comando
v2.0+	Hoja de referencia
v2.0+	añadir conexiones
v2.0+	eliminar conexion
v2.2+ (nota)	relleno de DAG
v2.0+	DAG borrados
v2.2+ (nota)	lista de DAG

Versiones secundarias	Comando
v2.0+	lista de tareas de DAG
v2.6+	lista-importación-errores de DAG
v2.2+ (nota)	lista-ejecuciones DAG
v2.2+ (nota)	próxima ejecución de DAG
v2.0+	pausa de DAG
v2.0+	informe de DAG
v2.4+	reserializar DAG
v2.0+	mostrar DAG
v2.0+	estado de DAG
v2.0+	prueba de DAG
v2.0+	activador de DAG
v2.0+	cancelar pausa en DAG
v2.4+	limpiar db
v2.0+	comportamientos de los proveedores

Versiones secundarias	Comando
v2.0+	lo que obtienen los proveedores
v2.0+	enlaces de proveedores
v2.0+	vínculos de proveedores
v2.0+	lista de proveedores
v2.8+	notificaciones de proveedores
v2.6+	secretos de los proveedores
v2.7+	activador de proveedores
v2.0+	widgets de proveedores
v2.6+	añadir roles permanentes
v2.6+	borrar roles permanentes
v2.6+	creación de roles
v2.0+	lista de roles
v2.0+	borrar tareas
v2.0+	tareas deps-fallidas

Versiones secundarias	Comando
v2.0+	lista de tareas
v2.0+	representar tareas
v2.0+	ejecutar tareas
v2.0+	estado de las tareas
v2.0+	estados-ejecución-DAG de las tareas
v2.0+	prueba de tareas
v2.0+	eliminar variables
v2.0+	obtener variables
v2.0+	conjunto de variables
v2.0+	lista de variables
v2.0+	versión

Uso de comandos que analizan los DAG

Si su entorno ejecuta Apache Airflow v2.0.2, los comandos de la CLI que analizan los DAG arrojarán error si el DAG usa complementos que dependen de paquetes instalados mediante un `requirements.txt`:

Apache Airflow v2.0.2

- `dags backfill`

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

Puede utilizar estos comandos de la CLI si sus DAG no utilizan complementos que dependan de paquetes instalados a través de un `requirements.txt`.

Código de muestra

La siguiente sección contiene ejemplos de diferentes formas de utilizar la CLI de Apache Airflow.

Cómo configurar, obtener o eliminar una variable de Apache Airflow v2

Puede utilizar el siguiente código de muestra para establecer, obtener o eliminar una variable con el 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
```

Cómo agregar una configuración al activar un DAG

Puede usar el siguiente código de ejemplo con Apache Airflow v2 para añadir una configuración al activar un DAG, por ejemplo `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)
```

Ejecución de comandos de la CLI en un túnel SSH hacia un host bastión

Use el siguiente ejemplo para ejecutar los comandos de la CLI de Airflow mediante un proxy de túnel SSH en un host bastión de Linux.

Uso de 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
   ```

Administración de las conexiones a Apache Airflow

En esta sección, se describen las diferentes formas de configurar una conexión de Apache Airflow para un entorno en Amazon Managed Workflows para Apache Airflow.

Temas

Descripción general de las variables y conexiones de Apache Airflow

En ciertos casos, es posible que desee especificar conexiones o variables adicionales para un entorno, como un perfil de AWS, o añadir su rol de ejecución en un objeto de conexión del almacén de metadatos de Apache Airflow y, a continuación, consultar la conexión desde dentro de un DAG.

- Apache Airflow autoadministrado. En una instalación autoadministrada de Apache Airflow, debe configurar las opciones de configuración de Apache Airflow en `airflow.cfg`.

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

- Apache Airflow en Amazon MWAA. En Amazon MWAA, debe añadir estos ajustes de configuración como opciones de configuración de Apache Airflow en la consola de Amazon MWAA. Las opciones de configuración de Apache Airflow se escriben como variables de entorno para su entorno y anulan el resto de las configuraciones existentes para el mismo ajuste.

Paquetes de proveedores de Apache Airflow instalados en entornos Amazon MWAA

En esta página, se enumeran los paquetes de proveedores de Apache Airflow que Amazon MWAA instala en todos los entornos compatibles de Apache Airflow. Para obtener más información sobre estos paquetes, consulte la referencia de Apache Airflow para paquetes extra.

> **Note**
>
> Amazon MWAA instala la versión 2.0.1 de Watchtower después de ejecutar `pip3 install -r requirements.txt` para garantizar que otras instalaciones de bibliotecas de Python no anulen la compatibilidad con los registros de CloudWatch.

Temas

Archivo de restricciones

A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que se está usando.

Los archivos de restricciones de Apache Airflow especifican las versiones de proveedores disponibles en el momento de la publicación de Apache Airflow. En muchos casos, de todos modos, los proveedores más recientes son compatibles con esa versión de Apache Airflow. Como debe usar restricciones, para especificar una versión más reciente de un paquete de proveedores, puede modificar el archivo de restricciones para una versión de proveedor específica:

1. Descargue el archivo de restricciones específicas de la versión desde GitHub, por ejemplo, https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt (reemplace “2.7.2” por la versión que desee usar).
2. Guarde el archivo de restricciones modificado en la carpeta “Amazon S3 DAGs” (DAG de Amazon S3) de su entorno de Amazon MWAA como `constraints-3.11-updated.txt`, por ejemplo.

3. Especifique sus requisitos como se muestra a continuación.

```
--constraint "/usr/local/airflow/dags/constraints-3.11-updated.txt"
apache-airflow-providers-amazon==version-number
```

> **Note**
>
> Si usa un servidor web privado, recomendamos que empaquete las bibliotecas necesarias como archivos WHL usando aws-mwaa-docker-images.

Paquetes de proveedores específicos para cada versión

La instalación de paquetes de proveedores le permite ver un tipo de conexión en la UI de Apache Airflow. También significa que no necesita especificar estos paquetes como una dependencia de Python en su archivo `requirements.txt`. En esta página, se enumeran los paquetes de proveedores de Apache Airflow que Amazon MWAA instala en todos los entornos compatibles de Apache Airflow.

> **Note**
>
> Para Apache Airflow v2 y versiones posteriores, Amazon MWAA instala la versión 2.0.1 de Watchtower después de ejecutar `pip3 install -r requirements.txt` para garantizar que otras instalaciones de bibliotecas de Python no anulen la compatibilidad con los registros de CloudWatch.

Puede especificar la última versión compatible de `apache-airflow-providers-amazon` para actualizar este proveedor.

Versiones de Apache Airflow compatibles:

v3.0.6

Tipo de conexión	Paquete
AWSConexión a	

Tipo de conexión	Paquete
	apache-airflow-providers-amazon[aiobotocore]==9.9.0
Conexión Postgres	apache-airflow-providers-postgres==6.2.1
Conexión FTP	apache-airflow-providers-ftp==3.13.1
Conexión Fab	apache-airflow-providers-fab==2.3.0
Conexión Celery	apache-airflow-providers-celery==3.12.1
Conexión HTTP	apache-airflow-providers-http==5.3.2
Conexión IMAP	apache-airflow-providers-imap==3.9.1
SQL común	apache-airflow-providers-common-sql==1.27.3
Conexión SQLite	apache-airflow-providers-sqlite==4.1.1

v2.10.3

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==9.0.0
Conexión Postgres	apache-airflow-providers-postgres==5.13.1
Conexión FTP	apache-airflow-providers-ftp==3.11.1

Tipo de conexión	Paquete
Conexión Fab	apache-airflow-providers-fab==1.5.0
Conexión Celery	apache-airflow-providers-celery==3.8.3
Conexión HTTP	apache-airflow-providers-http==4.13.2
Conexión IMAP	apache-airflow-providers-imap==3.7.0
SQL común	apache-airflow-providers-common-sql==1.19.0
Conexión SQLite	apache-airflow-providers-sqlite==3.9.0
Conexión SMTP	apache-airflow-providers-smtp==1.8.0

v2.10.1

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==8.28.0
Conexión Postgres	apache-airflow-providers-postgres==5.12.0
Conexión FTP	apache-airflow-providers-ftp==3.11.0
Conexión Fab	apache-airflow-providers-fab==1.3.0
Conexión Celery	apache-airflow-providers-celery==3.8.1

Tipo de conexión	Paquete
Conexión HTTP	apache-airflow-providers-http==4.13.0
Conexión IMAP	apache-airflow-providers-imap==3.7.0
SQL común	apache-airflow-providers-common-sql==1.16.0
Conexión SQLite	apache-airflow-providers-sqlite==3.9.0
Conexión SMTP	apache-airflow-providers-smtp==1.8.0

v2.9.2

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==8.24.0
Conexión Postgres	apache-airflow-providers-postgres==5.11.1
Conexión FTP	apache-airflow-providers-ftp==3.9.1
Conexión Fab	apache-airflow-providers-fab==1.1.1
Conexión Celery	apache-airflow-providers-celery==3.7.2
Conexión HTTP	apache-airflow-providers-http==4.11.1
Conexión IMAP	apache-airflow-providers-imap==3.6.1

Tipo de conexión	Paquete
SQL común	apache-airflow-providers-common-sql==1.14.0
Conexión SQLite	apache-airflow-providers-sqlite==3.8.1
Conexión SMTP	apache-airflow-providers-smtp==1.7.1

v2.8.1

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==8.16.0
Conexión Postgres	apache-airflow-providers-postgres==5.10.0
Conexión FTP	apache-airflow-providers-ftp==3.7.0
Conexión Celery	apache-airflow-providers-celery==3.5.1
Conexión HTTP	apache-airflow-providers-http==4.8.0
Conexión IMAP	apache-airflow-providers-imap==3.5.0
SQL común	apache-airflow-providers-common-sql==1.10.0
Conexión SQLite	apache-airflow-providers-sqlite==3.7.0

v2.7.2

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==8.7.1
Conexión Postgres	apache-airflow-providers-postgres==5.6.1
Conexión FTP	apache-airflow-providers-ftp==3.5.2
Conexión Celery	apache-airflow-providers-celery==3.3.4
Conexión HTTP	apache-airflow-providers-http==4.5.2
Conexión IMAP	apache-airflow-providers-imap==3.3.2
SQL común	apache-airflow-providers-common-sql==1.7.2
Conexión SQLite	apache-airflow-providers-sqlite==3.4.3

v2.6.3

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon[aiobotocore]==8.2.0
Conexión Postgres	apache-airflow-providers-postgres==5.5.1
Conexión FTP	apache-airflow-providers-ftp==3.4.2

Tipo de conexión	Paquete
Conexión Celery	apache-airflow-providers-celery==3.2.1
Conexión HTTP	apache-airflow-providers-http==4.4.2
Conexión IMAP	apache-airflow-providers-imap==3.2.2
SQL común	apache-airflow-providers-common-sql==1.5.2
Conexión SQLite	apache-airflow-providers-sqlite==3.4.2

v2.5.1

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon==7.1.0
Conexión Postgres	apache-airflow-providers-postgres==5.4.0
Conexión FTP	apache-airflow-providers-ftp==3.3.0
Conexión Celery	apache-airflow-providers-celery==3.1.0
Conexión HTTP	apache-airflow-providers-http==4.1.1
Conexión IMAP	apache-airflow-providers-imap==3.1.1
SQL común	apache-airflow-providers-common-sql==1.3.3
Conexión SQLite	apache-airflow-providers-sqlite==3.3.1

Tipo de conexión	Paquete

v2.4.3

Tipo de conexión	Paquete
AWSConexión a	apache-airflow-providers-amazon==6.0.0
Conexión Postgres	apache-airflow-providers-postgres==5.2.2
Conexión FTP	apache-airflow-providers-ftp==3.1.0
Conexión Celery	apache-airflow-providers-celery==3.0.0
Conexión HTTP	apache-airflow-providers-http==4.0.0
Conexión IMAP	apache-airflow-providers-imap==3.0.0
SQL común	apache-airflow-providers-common-sql==1.2.0
Conexión SQLite	apache-airflow-providers-sqlite==3.2.1

Información general sobre los tipos de conexión

Apache Airflow almacena las conexiones como un string de conexión de URI. Proporciona una plantilla de conexiones en la interfaz de usuario de Apache Airflow para generar el string de conexión de URI, independientemente del tipo de conexión. Si no hay ninguna plantilla de conexión disponible en la interfaz de usuario de Apache Airflow, se puede utilizar una plantilla de conexión alternativa para generar este string de conexión de URI, por ejemplo, mediante la plantilla de conexión HTTP. La principal diferencia es el prefijo URI, por ejemplo, `my-conn-type://`, que los proveedores de Apache Airflow suelen ignorar en una conexión. En esta página, se describe cómo utilizar las

plantillas de conexión de la interfaz de usuario de Apache Airflow de manera intercambiable para distintos tipos de conexión.

> **Warning**
>
> No sobrescriba la conexión `aws_default` en Amazon MWAA. Amazon MWAA utiliza esta conexión para llevar a cabo diversas tareas críticas, como la recopilación de registros de tareas. Sobrescribir esta conexión puede provocar que se pierdan datos y se interrumpa la disponibilidad del entorno.

Temas

Ejemplo de string de conexión de URI

En el siguiente ejemplo, se muestra un string de conexión de URI para el tipo de conexión 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'
```

Ejemplo de plantilla de conexión

En el siguiente ejemplo, se muestra la plantilla de conexión HTTP en la UI de Apache Airflow.

Apache Airflow v3

Apache Airflow v2

Ejemplo de uso de una plantilla de conexión HTTP para una conexión Jdbc

En el siguiente ejemplo, se muestra cómo usar la plantilla de conexión HTTP para una conexión Jdbc en la UI de Apache Airflow.

Apache Airflow v3

En el siguiente ejemplo, se muestra el string de conexión de URI generada por Apache Airflow para el ejemplo de esta sección.

```
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
```

En el siguiente ejemplo, se muestra cómo usar la plantilla de conexión HTTP para una conexión Jdbc para Apache Airflow v3 en la UI de Apache Airflow.

Apache Airflow v2

En el siguiente ejemplo, se muestra el string de conexión de URI generada por Apache Airflow para el ejemplo de esta sección.

```
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
```

En el siguiente ejemplo, se muestra cómo usar la plantilla de conexión HTTP para una conexión Jdbc para Apache Airflow v2 en la UI de Apache Airflow.

Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager

AWS Secrets Manager es un backend de Apache Airflow alternativo compatible con un entorno de Amazon Managed Workflows para Apache Airflow. En esta guía, se muestra cómo usar AWS Secrets Manager para almacenar de forma segura los secretos de las variables de Apache Airflow y una conexión de Apache Airflow en Amazon Managed Workflows para Apache Airflow.

> **Note**
>
> - Los secretos creados tendrán un costo. Para obtener información sobre el precio de Secrets Manager, consulte precios de AWS.

- Amazon MWAA también admite Almacén de parámetros de AWS Systems Manager como backend de secretos. Para obtener más información, consulte la documentación de Amazon Provider Package.

Contenido

Primer paso: otorgar permiso a Amazon MWAA para acceder a las claves secretas de Secrets Manager

El rol de ejecución de su entorno de Amazon MWAA necesita acceso de lectura a la clave secreta de AWS Secrets Manager. La siguiente política de IAM permite el acceso de lectura y escritura mediante la política SecretsManagerReadWrite administrada por AWS.

Pasos para asociar la política a su rol de ejecución

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija su rol de ejecución en el panel Permisos.
4. Seleccione Asociar políticas.
5. Escriba `SecretsManagerReadWrite` en el campo de texto Filtrar políticas.
6. Elija Asociar política.

Si no desea usar una política de permisos administrada por AWS, puede actualizar directamente el rol de ejecución de su entorno para permitir cualquier nivel de acceso a sus recursos de Secrets Manager. Por ejemplo, la siguiente declaración de política otorga acceso de lectura a todos los secretos que cree en una Región de AWS específica en Secrets Manager.

JSON

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

Segundo paso: crear el backend de Secrets Manager como opción de configuración de Apache Airflow

En la siguiente sección se describe cómo crear una opción de configuración de Apache Airflow en la consola de Amazon MWAA para el backend de AWS Secrets Manager. Si usa una configuración con el mismo nombre en `airflow.cfg`, la configuración que cree en los siguientes pasos tendrá prioridad y anulará los ajustes de configuración.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.

3. Elija Editar.
4. Elija Siguiente.
5. Seleccione Agregar configuración personalizada en el panel Opciones de configuración de Airflow. Agregue los siguientes pares clave-valor:

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

   b. **`secrets.backend_kwargs`**: Con **`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`**, se configura Apache Airflow para poder buscar cadenas de conexión y variables en rutas de `airflow/connections/*` y `airflow/variables/*`.

      Puede utilizar un patrón de búsqueda para reducir el número de llamadas a la API que Amazon MWAA realiza a Secrets Manager en su nombre. Si no especifica un patrón de búsqueda, Apache Airflow busca todas las conexiones y variables en el backend configurado. Al especificar un patrón, se reducen las posibles rutas que busca Apache Airflow. Así de reducen los costos relacionados con el uso de Secrets Manager con Amazon MWAA.

      Para especificar un patrón de búsqueda, especifique los parámetros `connections_lookup_pattern` y `variables_lookup_pattern`. Estos parámetros aceptan una cadena RegEx como entrada. Por ejemplo, para buscar secretos que comiencen por `test`, introduzca lo siguiente 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` y `variables_lookup_pattern`, debe instalar la versión 7.3.0 o superior de `apache-airflow-providers-amazon`. Para obtener más información sobre la actualización de los paquetes de proveedores a versiones más recientes, consulte Archivo de restricciones.

6. Seleccione Save.

Paso tres: generar una cadena URI de conexión de AWS a Apache Airflow

Para crear una cadena de conexión, utilice la tecla “tab” del teclado para aplicar los pares clave-valor en el objeto Conexión. También recomendamos crear una variable para el objeto `extra` en la sesión del shell. En la siguiente sección, se explican los pasos para generar una cadena URI de conexión a Apache Airflow para un entorno de Amazon MWAA mediante Apache Airflow o un script de Python.

Apache Airflow CLI

La siguiente sesión del shell utiliza la CLI de Airflow local para generar una cadena de conexión. Si no tiene la CLI instalada, le recomendamos que utilice el script de Python.

1. Abra una sesión del shell de Python:

```
python3
```

2. Escriba el siguiente comando:

```
>>> import json
```

3. Escriba el siguiente comando:

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

4. Cree una variable para el objeto `extra` en la sesión del shell. Sustituya los valores de ejemplo de *YOUR_EXECUTION_ROLE_ARN* por el ARN del rol de ejecución y la región en *us-east-1* (como `us-east-1`).

```
>>> extra=json.dumps({'role_arn': 'YOUR_EXECUTION_ROLE_ARN', 'region_name': 'us-east-1'})
```

5. Cree el objeto de conexión. Sustituya el valor de muestra en `myconn` por el nombre de la conexión de Apache Airflow.

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

6. Utilice la tecla “tab” del teclado para aplicar cada uno de los siguientes pares clave-valor del objeto de conexión. Sustituya los valores de muestra en *rojo*.

   a. Especifique el tipo de conexión de AWS:

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

   b. Especifique la opción de base de datos de Apache Airflow:

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

   c. Especifique la URL de la interfaz de usuario de Apache Airflow en Amazon MWAA:

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

   d. Especifique el ID de la clave de acceso de AWS (nombre de usuario) para iniciar sesión en Amazon MWAA:

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

   e. Especifique la clave de acceso secreta de AWS (contraseña) para iniciar sesión en Amazon MWAA:

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

   f. Especifique la variable de sesión de shell de `extra`:

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

   g. Cierre el objeto de conexión.

   ```
   ... )
   ```

7. Imprima la cadena URI de conexión:

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

   Consulte la cadena URI de conexión en la respuesta:

```
'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

El siguiente script de Python no requiere la CLI de Apache Airflow.

1. Copie el contenido del código de ejemplo siguiente y guárdelo 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 = 'us-east-1'

        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. Sustituya los marcadores de posición en *rojo*.
3. Ejecute el siguiente script para generar una cadena de conexión.

```
python3 mwaa_connection.py
```

Paso cuatro: añadir las variables en Secrets Manager

En la siguiente sección se describe cómo crear el secreto de una variable en Secrets Manager.

Pasos para crear el secreto

1. Abra la consola de AWS Secrets Manager.

2. Elija Almacenar un secreto nuevo.
3. Elija Otro tipo de secreto.
4. En el panel Especificar los pares clave-valor para almacenarlos en este secreto, elija Texto simple.
5. Añada el valor de la variable como Texto simple en el siguiente formato.

   ```
   "YOUR_VARIABLE_VALUE"
   ```

   Por ejemplo, para especificar un número entero:

   ```
   14
   ```

   Por ejemplo, para especificar una cadena:

   ```
   "mystring"
   ```
6. En Clave de cifrado, elija una opción de clave de AWS KMS de la lista desplegable.
7. Introduzca un nombre en el campo de texto para el nombre del secreto con el formato que se indica a continuación.

   ```
   airflow/variables/YOUR_VARIABLE_NAME
   ```

   Por ejemplo:

   ```
   airflow/variables/test-variable
   ```
8. Elija Siguiente.
9. En la página Configurar secreto, en el panel Nombre y descripción del secreto, haga lo siguiente.

   a. En Nombre del secreto, proporcione un nombre para el secreto.

   b. (Opcional) En Descripción, escriba una descripción del nombre de su secreto.

   Elija Siguiente.
10. En Configurar la rotación. Opcional, deje las opciones predeterminadas y seleccione Siguiente.
11. Repita estos pasos en Secrets Manager para cualquier variable adicional que desee añadir.
12. En la página Revisar, revise los detalles de su secreto y, a continuación, elija Almacenar.

Paso cinco: añadir la conexión en Secrets Manager

En la siguiente sección se describe cómo crear el secreto para el URI de la cadena de conexión en Secrets Manager.

Pasos para crear el secreto

1. Abra la consola de AWS Secrets Manager.
2. Elija Almacenar un secreto nuevo.
3. Elija Otro tipo de secreto.
4. En el panel Especificar los pares clave-valor para almacenarlos en este secreto, elija Texto simple.
5. Añada la cadena URI de conexión como Texto simple con el formato que se indica a continuación.

   ```
   YOUR_CONNECTION_URI_STRING
   ```

   Por ejemplo:

   ```
   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**
   >
   > Apache Airflow analiza cada uno de los valores de la cadena de conexión. No use comillas simples ni dobles, pues al hacerlo se analizaría la conexión como una sola cadena.

6. En Clave de cifrado, elija una opción de clave de AWS KMS de la lista desplegable.
7. Introduzca un nombre en el campo de texto para el nombre del secreto con el formato que se indica a continuación.

   ```
   airflow/connections/YOUR_CONNECTION_NAME
   ```

   Por ejemplo:

```
airflow/connections/myconn
```

8. Elija Siguiente.
9. En la página Configurar secreto, en el panel Nombre y descripción del secreto, haga lo siguiente.

   a. En Nombre del secreto, proporcione un nombre para el secreto.

   b. (Opcional) En Descripción, escriba una descripción del nombre de su secreto.

   Elija Siguiente.
10. En Configurar la rotación. Opcional, deje las opciones predeterminadas y seleccione Siguiente.
11. Repita estos pasos en Secrets Manager para cualquier variable adicional que desee añadir.
12. En la página Revisar, revise los detalles de su secreto y, a continuación, elija Almacenar.

Código de muestra

- Aprenda a usar la clave secreta para la conexión de Apache Airflow (`myconn`) en esta página usando el código de ejemplo que se encuentra en Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow.
- Aprenda a usar la clave secreta para la variable de Apache Airflow (`test-variable`) en esta página usando el código de ejemplo que se encuentra en Uso de una clave secreta en AWS Secrets Manager para una variable de Apache Airflow.

Recursos

- Para obtener más información sobre cómo configurar los secretos en Secrets Manager con la consola y la AWS CLI, consulte cómo crear un secreto en la guía del usuario de AWS Secrets Manager.
- Utilice un script de Python para migrar un gran volumen de variables y conexiones de Apache Airflow a Secrets Manager en Mueva sus conexiones y variables de Apache Airflow a AWS Secrets Manager.

Siguientes pasos

- Aprenda a generar un token para acceder a la interfaz de usuario de Apache Airflow en Acceso a Apache Airflow.

Administración de entornos Amazon MWAA

La consola Amazon Managed Workflows para Apache Airflow contiene opciones integradas para configurar el acceso público o privado a la interfaz de usuario de Apache Airflow. También incluye opciones integradas para configurar el tamaño del entorno, cuándo escalar los procesos de trabajo y opciones de configuración de Apache Airflow que permiten anular las configuraciones de Apache Airflow a las que normalmente solo se puede acceder en `airflow.cfg`. En este capítulo, se describe cómo usar estas configuraciones en la consola Amazon MWAA.

Temas

Configuración de la clase de entorno Amazon MWAA

La clase de entorno que elija para su entorno Amazon MWAA determinará el tamaño de los contenedores AWS Fargate gestionados por AWS en los que se ejecuta Celery Executor y de la base de datos de metadatos PostgreSQL Amazon Aurora gestionada por AWS en la que los programadores de Apache Airflow crean instancias de tareas. En esta página, se describe cada clase de entorno de Amazon MWAA y los pasos para actualizar la clase de entorno en la consola Amazon MWAA.

Secciones

Capacidades del entorno

La siguiente sección contiene las tareas simultáneas predeterminadas de Apache Airflow, la memoria de acceso aleatorio (RAM) y las unidades de procesamiento centralizado virtuales (vCPU) de cada clase de entorno. Las tareas simultáneas enumeradas suponen que la simultaneidad de las tareas no supera la capacidad de procesamiento de trabajo de Apache Airflow en el entorno.

En la siguiente tabla, la capacidad de los DAG se refiere a las definiciones de los DAG, no a las ejecuciones, y supone que los DAG son dinámicos en un único archivo de Python y están escritos siguiendo las prácticas recomendadas de Apache Airflow.

Las ejecuciones de tareas dependen del número de tareas programadas simultáneamente y se parte del supuesto de que el número de ejecuciones de DAG programadas para que comiencen al mismo tiempo no supere el valor predeterminado de `max_dagruns_per_loop_to_schedule`, así como el tamaño y la cantidad de procesos de trabajo, tal como se detalla en esta página.

mw1.micro

- Capacidad de hasta 25 DAG
- 3 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 1 vCPU, 3 GB de RAM
  - Procesos de trabajo y programador: 1 vCPU, 3 GB de RAM
  - Base de datos: 2 vCPU, 4 GB de RAM

> **Note**
>
> mw1.micro no admite escalado automático.

mw1.small

- Capacidad de hasta 50 DAG
- 5 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 1 vCPU, 2 GB de RAM cada uno
  - Procesos de trabajo: 1 vCPU, 2 GB de RAM cada uno

- Programadores: 1 vCPU, 2 GB de RAM cada uno
- Base de datos: 2 vCPU, 4 GB de RAM

mw1.medium

- Capacidad de hasta 250 DAG
- 10 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 1 vCPU de 2 GB de RAM cada uno
  - Procesos de trabajo: 2 vCPU de 4 GB de RAM cada uno
  - Programadores: 2 vCPU de 4 GB de RAM cada uno
  - Base de datos: 2 vCPU de 8 GB de RAM

mw1.large

- Capacidad de hasta 1000 DAG
- 20 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 2 vCPU de 4 GB de RAM cada uno
  - Procesos de trabajo: 4 vCPU de 8 GB de RAM cada uno
  - Programadores: 4 vCPU de 8 GB de RAM cada uno
  - Base de datos: 2 vCPU de 8 GB de RAM

mw1.xlarge

- Capacidad de hasta 2000 DAG
- 40 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 4 vCPU de 12 GB de RAM cada uno
  - Procesos de trabajo: 8 vCPU de 24 GB de RAM cada uno
  - Programadores: 8 vCPU de 24 GB de RAM cada uno
  - Base de datos: 4 vCPU de 32 GB de RAM

mw1.2xlarge

- Capacidad de hasta 4000 DAG
- 80 tareas simultáneas (de forma predeterminada)
- Componentes:
  - Servidores web: 8 vCPU de 24 GB de RAM cada uno
  - Procesos de trabajo: 16 vCPU de 48 GB de RAM cada uno
  - Programadores: 16 vCPU de 48 GB de RAM cada uno
  - Base de datos: 8 vCPU de 64 GB de RAM

Se puede utilizar `celery.worker_autoscale` para aumentar las tareas por proceso de trabajo. Para obtener más información, consulte the section called “Ejemplo de caso de uso de alto rendimiento”.

Programadores de Apache Airflow

La siguiente sección contiene las opciones de programadores de Apache Airflow disponibles en Amazon MWAA y de qué modo el número de programadores afecta al número de desencadenadores.

En Apache Airflow, un desencadenador administra las tareas y las aplaza hasta que se cumplen determinadas condiciones especificadas mediante un desencadenador. En Amazon MWAA, el desencadenador se ejecuta junto con el programador en la misma tarea de Fargate. Al aumentar el número de programadores, se aumenta, en consecuencia, el número de desencadenadores disponibles, lo que optimiza la forma en que el entorno administra las tareas aplazadas. Esto garantiza una administración eficiente de las tareas, programándolas rápidamente para que se ejecuten cuando se cumplan las condiciones.

Apache Airflow v3

- v3: para entornos más grandes que mw1.micro, acepta valores comprendidos entre 2 y 5. El valor predeterminado es 2 para todos los tamaños de entorno, excepto mw1.micro, cuyo valor de forma predeterminada es 1.

Apache Airflow v2

- v2: para entornos con un tamaño superior a mw1.micro, acepta valores comprendidos entre 2 y 5. El valor predeterminado es 2 para todos los tamaños de entorno, excepto mw1.micro, cuyo valor de forma predeterminada es 1.

Configuración del escalado automático de los procesos de trabajo de Amazon MWAA

El mecanismo de escalado automático aumenta automáticamente la cantidad de procesos de trabajo de Apache Airflow en respuesta a las tareas en ejecución y en cola en su entorno de Amazon Managed Workflows para Apache Airflow, y elimina procesos de trabajo adicionales cuando no hay más tareas en cola o en ejecución. En esta página, se describe cómo configurar el escalado automático especificando el número máximo de procesos de trabajo de Apache Airflow que se ejecutan en su entorno con la consola de Amazon MWAA.

> **Note**
>
> Amazon MWAA utiliza las métricas de Apache Airflow para determinar cuándo se necesitan más procesos de trabajo de Celery Executor y aumenta el número de procesos de trabajo de Fargate hasta el valor especificado por `max-workers` según sea necesario. A medida que los procesos de trabajo adicionales completan el trabajo y la carga de trabajo disminuye, Amazon MWAA los elimina y, por lo tanto, vuelve al valor establecido por `min-workers`. Si los procesos de trabajo eligen nuevas tareas mientras se reducen, Amazon MWAA se queda con el recurso de Fargate y no elimina el proceso de trabajo. Para obtener más información, consulte Cómo funciona el escalado automático de Amazon MWAA.

Secciones

Cómo funciona el escalado de los procesos de trabajo

Amazon MWAA utiliza métricas de `RunningTasks` y `QueuedTasks`, donde (tareas en ejecución + tareas en cola)/(tareas por proceso de trabajo) = (procesos necesarios). Si el número de procesos requerido es superior al número actual de procesos de trabajo, Amazon MWAA añadirá los contenedores de procesos de trabajo de Fargate a ese valor, hasta alcanzar el valor máximo especificado por `max-workers`.

A medida que la carga de trabajo disminuye y la suma de las métricas `RunningTasks` y `QueuedTasks` se reduce, Amazon MWAA solicita a Fargate reducir verticalmente los procesos de trabajo para el entorno. Los procesos de trabajo que están terminando de trabajar permanecerán protegidos durante la reducción hasta que terminen su trabajo. Según la carga de trabajo, es posible que las tareas queden en cola mientras se reducen los procesos de trabajo.

Uso de la consola de Amazon MWAA

Puede elegir el número máximo de procesos de trabajo que se pueden ejecutar en su entorno simultáneamente en la consola de Amazon MWAA. Por defecto, puede especificar un valor máximo de hasta 25.

Pasos para configurar el número de procesos de trabajo

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione Editar.
4. Elija Siguiente.
5. Introduzca un valor en Recuento máximo de procesos de trabajo en el panel de clases de entorno.
6. Seleccione Save.

> **Note**
>
> Este proceso puede tardar unos minutos hasta que los cambios se apliquen en el entorno.

Ejemplo de caso de uso de alto rendimiento

La sección siguiente describe el tipo de configuraciones que puede usar para habilitar el alto rendimiento y el paralelismo en un entorno.

Apache Airflow en las instalaciones

Por lo general, en una plataforma de Apache Airflow en las instalaciones, se configuran los ajustes de paralelismo, escalado automático y simultaneidad de tareas en su archivo `airflow.cfg`:

- `core.parallelism`: el número máximo de instancias de tareas que el programador puede ejecutar simultáneamente.
- `core.dag_concurrency`: la simultaneidad máxima de los DAG (no de los procesos de trabajo).
- `celery.worker_autoscale`: el número máximo y mínimo de tareas que se puede ejecutar simultáneamente en cualquier proceso de trabajo.

Por ejemplo, si `core.parallelism` estuviera configurado en `100` y `core.dag_concurrency` estuviera configurado en 7, con 2 DAG solo podría ejecutar un total de `14` tareas simultáneamente. Por ejemplo, cada DAG está configurado para ejecutar solo 7 tareas simultáneamente (en `core.dag_concurrency`), aunque el paralelismo general esté configurado en `100` (en `core.parallelism`).

> **Note**
>
> `core.dag_concurrency` no está disponible en Apache Airflow v3.

En un entorno de Amazon MWAA

En un entorno de Amazon MWAA, puede configurar estos ajustes directamente en la consola de Amazon MWAA con Uso de las opciones de configuración de Apache Airflow en Amazon MWAA, Configuración de la clase de entorno Amazon MWAA y el mecanismo de escalado automático del número máximo de procesos de trabajo. Si bien `core.dag_concurrency` no está disponible en la lista desplegable como opción de configuración de Apache Airflow en la consola de Amazon MWAA, puede agregarla como opción de configuración personalizada de Apache Airflow.

Supongamos que, cuando creó su entorno, eligió los ajustes siguientes:

1. La clase de entorno mw1.small, que controla el número máximo de tareas simultáneas que cada proceso de trabajo puede ejecutar por defecto y la vCPU de los contenedores.
2. La configuración predeterminada de `10` procesos de trabajo en el Número máximo de procesos de trabajo.
3. Una opción de configuración de Apache Airflow para `celery.worker_autoscale` de `5,5` tareas por proceso de trabajo.

Esto significa que puede ejecutar 50 tareas simultáneas en su entorno. Las tareas superiores a 50 se pondrán en cola y esperarán a que se completen las tareas en ejecución.

Ejecución de más tareas simultáneas. Puede modificar su entorno para ejecutar más tareas simultáneamente mediante las siguientes configuraciones:

1. Aumente el número máximo de tareas simultáneas que cada proceso de trabajo puede ejecutar por defecto y la vCPU de los contenedores eligiendo la clase de entorno `mw1.medium` (10 tareas simultáneas de forma predeterminada).
2. Añádala `celery.worker_autoscale` como un opción de configuración de Apache Airflow.
3. Aumente el número máximo de procesos de trabajo. En este ejemplo, aumentar el número máximo procesos de trabajo de `10` a `20` duplicaría el número de tareas que el entorno puede ejecutar simultáneamente.

Especifique el número mínimo de procesos de trabajo. También puede especificar el número mínimo y máximo de procesos de trabajo de Apache Airflow que se ejecutan en su entorno mediante la AWS Command Line Interface (AWS CLI). Por ejemplo:

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

Para obtener más información, consulte el comando update-environment en la AWS CLI.

Solución de problemas de tareas bloqueadas en estado de ejecución

En ocasiones excepcionales, para Apache Airflow puede parecer que todavía hay tareas en ejecución. Para resolver este problema, debe borrar la tarea pendiente en la interfaz de usuario de Apache Airflow. Para obtener información, consulte el tema sobre solución de problemas Solución de problemas de Amazon Managed Workflows para Apache Airflow.

Siguientes pasos

- Obtenga más información sobre las mejores prácticas que recomendamos para ajustar el rendimiento de su entorno Ajuste del desempeño de Apache Airflow en Amazon MWAA.

Configuración del escalado automático del servidor web de Amazon MWAA

Para los entornos que ejecutan Apache Airflow v2.2.2 y versiones posteriores, Amazon MWAA escala dinámicamente sus servidores web para gestionar las cargas de trabajo fluctuantes, lo que a su vez evita problemas de rendimiento durante los picos de carga. Al escalar automáticamente la cantidad de servidores web en función del uso de la CPU y el número de conexiones activas, Amazon MWAA garantiza que su entorno de Apache Airflow pueda adaptarse bien a una mayor demanda, ya sea por solicitudes de API de REST, el uso de CLI o más usuarios simultáneos de la interfaz de usuario de Apache Airflow.

Secciones

- Cómo funciona el escalado de servidores web
- Uso de la consola de Amazon MWAA

Cómo funciona el escalado de servidores web

Amazon MWAA usa la métrica del contenedor `CPUUtilization` y la métrica del equilibrador de carga `ActiveConnectionCount` para determinar si es necesario escalar los servidores web en función de la cantidad de tráfico. Si `CPUUtilization` es superior a 70 o `ActiveConnectionCount` es superior a 15, Amazon MWAA agregará más contenedores de servidores web Fargate hasta el valor máximo especificado por `MaxWebservers`.

A medida que el tráfico disminuye y los valores de `CPUUtilization` y `ActiveConnectionCount` se reducen, Amazon MWAA solicita a Fargate reducir verticalmente los contenedores del servidor web para el entorno hasta el valor mínimo establecido por `MinimumWebservers`.

Uso de la consola de Amazon MWAA

Puede elegir el número de servidores web que se pueden ejecutar en su entorno simultáneamente en la consola de Amazon MWAA. De forma predeterminada, el número mínimo de servidores web es dos y el número máximo de servidores web es cinco.

Pasos para configurar el número de servidores web

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione Editar.
4. Elija Siguiente.
5. Introduzca un valor en Maximum webserver count (Número máximo de servidores web) en el panel Environment class (Clase de entorno).
6. A continuación, introduzca un valor en Minimum webserver count (Número mínimo de servidores web).
7. Seleccione Save.

> **Note**
>
> Este proceso puede tardar unos minutos hasta que los cambios se apliquen en el entorno.

Uso de las opciones de configuración de Apache Airflow en Amazon MWAA

Las opciones de configuración de Apache Airflow se pueden adjuntar a su entorno de Amazon Managed Workflows para Apache Airflow como variables de entorno. Puede elegir una opción de la lista desplegable sugerida o especificar opciones de configuración personalizadas para su versión de Apache Airflow en la consola Amazon MWAA. En esta página, se describen las opciones de configuración de Apache Airflow disponibles y cómo usarlas para anular los ajustes de configuración de Apache Airflow en su entorno.

Contenido

Requisitos previos

Para poder llevar a cabo los pasos de esta página, necesitará lo siguiente.

- Permisos: el administrador debe haber concedido a su Cuenta de AWS acceso a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno. Además, su rol de ejecución debe permitir que su entorno de Amazon MWAA acceda a los recursos de AWS que utiliza su entorno.
- Acceso: si tiene que acceder a los repositorios públicos para instalar dependencias directamente en el servidor web, su entorno debe estar configurado con acceso a un servidor web de red pública. Para obtener más información, consulta the section called “Modos de acceso de Apache Airflow”.
- Configuración de Amazon S3: el bucket de Amazon S3 que se utiliza para almacenar los DAG, los complementos personalizados en `plugins.zip` y las dependencias de Python en

`requirements.txt` deben estar configurados con el acceso público bloqueado y el control de versiones activado.

Funcionamiento

Al crear un entorno, Amazon MWAA adjunta los ajustes de configuración que usted especifique en la consola Amazon MWAA en las opciones de configuración de Airflow como variables de entorno al contenedor AWS Fargate de su entorno. Si utiliza una configuración con el mismo nombre en `airflow.cfg`, las opciones que especifique en la consola Amazon MWAA anularán los valores incluidos en `airflow.cfg`.

Si bien no exponemos el `airflow.cfg` en la interfaz de usuario de Apache Airflow de un entorno de Amazon MWAA de manera predeterminada, puede cambiar las opciones de configuración de Apache Airflow directamente en la consola de Amazon MWAA, incluida la configuración de `webserver.expose_config` para exponer las configuraciones.

Uso de las opciones de configuración para cargar complementos

De forma predeterminada, en Apache Airflow v2 y versiones posteriores, los complementos se configuran para que se carguen de forma “lenta” mediante la configuración `core.lazy_load_plugins : True`. Si usa complementos personalizados, debe agregar `core.lazy_load_plugins : False` como opción de configuración de Apache Airflow para cargar los complementos al inicio de cada proceso de Airflow y anular la configuración predeterminada.

Información general de las opciones de configuración

Cuando agrega una configuración en la consola Amazon MWAA, Amazon MWAA escribe la configuración como una variable de entorno.

- Opciones enumeradas. Puede elegir uno de los ajustes de configuración disponibles para su versión de Apache Airflow en la lista desplegable. Por ejemplo, `dag_concurrency: 16`. El ajuste de configuración se traduce al contenedor de Fargate de su entorno como `AIRFLOW__CORE__DAG_CONCURRENCY : 16`
- Opciones personalizadas. También puede especificar las opciones de configuración de Airflow que no aparecen en la lista desplegable para su versión de Apache Airflow. Por ejemplo, `foo.user: YOUR_USER_NAME`. El ajuste de configuración se traduce al contenedor de Fargate de su entorno como `AIRFLOW__FOO__USER : YOUR_USER_NAME`

Opciones de configuración de Apache Airflow

En la siguiente imagen, se muestra dónde puede personalizar las opciones de configuración de Apache Airflow en la consola Amazon MWAA.

Referencia de Apache Airflow

Para obtener una lista de las opciones de configuración compatibles con Apache Airflow, consulte la referencia de configuración en la guía de referencia de Apache Airflow. Para ver las opciones de la versión de Apache Airflow que ejecuta en Amazon MWAA, seleccione la versión en la lista desplegable.

Uso de la consola de Amazon MWAA

A continuación, se explican los pasos que debe seguir para añadir una opción de configuración de Apache Airflow a su entorno.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Editar.
4. Elija Siguiente.
5. Seleccione Agregar configuración personalizada en el panel Opciones de configuración de Airflow.
6. En la lista desplegable, elija una opción de configuración e introduzca un valor. También puede escribir una configuración personalizada e introducir un valor.
7. Seleccione Agregar configuración personalizada para cada configuración que desee agregar.
8. Seleccione Save.

Referencia de la configuración

La siguiente sección contiene la lista de configuraciones de Apache Airflow disponibles en la lista desplegable de la consola de Amazon MWAA.

Configuración de correo electrónico

En la siguiente lista, se muestran las opciones de configuración de las notificaciones por correo electrónico de Airflow que están disponibles en Amazon MWAA para Apache Airflow v2 y v3.

Recomendamos utilizar el puerto 587 para el tráfico SMTP. De forma predeterminada, AWS bloquea el tráfico SMTP de salida en el puerto 25 de todas las instancias de Amazon EC2. Si desea enviar tráfico saliente por el puerto 25, solicite que se elimine esta restricción.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
email.email_backend	La utilidad Apache Airflow utilizada para las notificaciones por correo electrónico en email_backend.	airflow.utils.email.send_email_smtp
smtp.smtp_host	El nombre del servidor saliente utilizado como dirección de correo electrónico en smtp_host.	localhost
smtp.smtp_starttls	La seguridad de la capa de transporte (TLS) se usa para cifrar el correo electrónico a través de Internet en smtp_starttls.	False
smtp.smtp_ssl	Se usa la capa de sockets seguros (SSL) para conectar el servidor y el cliente de correo electrónico en smtp_ssl.	True
smtp.smtp_port	El puerto de protocolo de control de transmisión (TCP) designado al servidor en smtp_port.	587

Opción de configuración de Airflow	Descripción	Ejemplo de valor
smtp.smtp_mail_from	La dirección de correo electrónico saliente en smtp_mail_from.	myemail@domain.com

Configuración de tareas

En la siguiente lista, se muestran las configuraciones disponibles en la lista desplegable para las tareas de Airflow en Amazon MWAA para Apache Airflow v2 y v3.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
core.default_task_retries	El número de veces que se debe volver a intentar una tarea de Apache Airflow en default_task_retries.	3
core.parallelism	El número máximo de instancias de tareas que se pueden ejecutar simultáne amente en todo el entorno en paralelo (paralelismo).	40

Configuraciones del programador

En la siguiente lista, se muestran las configuraciones del programador de Apache Airflow que están disponibles en el menú desplegable de Amazon MWAA para Apache Airflow v2 y v3.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
scheduler.catchup_by_default		False

Opción de configuración de Airflow	Descripción	Ejemplo de valor
	Indica al programador que cree una ejecución de DAG para “alcanzar” el intervalo de tiempo específico de catchup_by_default.	
scheduler.scheduler_zombie_task_threshold **Note** No está disponible en Apache Airflow v3.	Indica al programador si debe marcar la instancia de la tarea como fallida y volver a programarla en scheduler_zombie_task_threshold.	300

Configuraciones del proceso de trabajo

En la siguiente lista, se muestran las configuraciones del proceso de trabajo de Airflow que están disponibles en el menú desplegable de Amazon MWAA para Apache Airflow v2 y v3.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
celery.worker_autoscale	El número máximo y mínimo de tareas que se pueden ejecutar simultáneamente en cualquier proceso de trabajo que utilice el Celery Executor de worker_autoscale. El valor debe estar separado por comas en el siguiente orden: `max_concurrency,min_concurrency` .	16,12

Configuraciones del servidor web

En la siguiente lista, se muestran las configuraciones del servidor web de Apache Airflow que están disponibles en el menú desplegable de Amazon MWAA para Apache Airflow v2 y v3.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
webserver.default_ui_timezone **Note** No está disponible en Apache Airflow v3.	La configuración de fecha y hora de la interfaz de usuario de Apache Airflow predeterminada en default_ui_timezone. **Note** Si se configura la opción `default_ui_timezone` , no se modifica la zona horaria en la que está programada la ejecución de los DAG. Para cambiar la zona horaria de los DAG, puede utilizar un complemento personalizado. Para obtener más información, consulta the section called “Cómo cambiar la zona horaria de un DAG”.	America/New_York

Configuraciones del desencadenador

En la siguiente lista, se muestran las configuraciones del desencadenador de Apache Airflow que están disponibles en Amazon MWAA para Apache Airflow v2 y v3.

Opción de configuración de Airflow	Descripción	Ejemplo de valor
mwaa.triggerer_enabled	Se utiliza para activar y desactivar el desencadenador en Amazon MWAA. De forma predeterminada, este valor se establece en `True`. Si se establece como `False`, Amazon MWAA no iniciará ningún proceso desencade nador en los programadores.	True
triggerer.default_capacity (en la versión 2) triggerer.capacity (en la versión 3)	Define el número de desencadenamientos que cada desencadenador puede ejecutar en paralelo. En Amazon MWAA, esta capacidad se establece para cada desencadenador y para cada programador, ya que ambos componentes funcionan juntos. El valor predeterminado para cada programador es `60`, `125`, `250`, `500` y `1000` para instancia s pequeñas, medianas y grandes, xlarge y 2xlarge respectivamente.	125

Ejemplos y código de ejemplo

Ejemplo de DAG

Puede usar el siguiente DAG para imprimir las opciones de configuración `email_backend` de Apache Airflow. Para ejecutarlo en respuesta a eventos de Amazon MWAA, copie el código en la carpeta DAG de su entorno en su bucket de almacenamiento de 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()
```

Ejemplo de configuración de las notificaciones por correo electrónico

Las siguientes opciones de configuración de Apache Airflow se pueden utilizar para una cuenta de correo electrónico de Gmail.com con una contraseña de aplicación. Para obtener más información, consulte Cómo iniciar sesión con contraseñas de aplicaciones en la guía de referencia de Gmail.

Siguientes pasos

- Obtenga información sobre cómo cargar la carpeta de DAG a su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Actualización de un entorno de Amazon MWAA

> **Note**
>
> Las actualizaciones rápidas de Amazon MWAA aún no son compatibles con las regiones Canada West (Calgary) y Asia Pacific (Malasia).

Las actualizaciones del entorno de Amazon MWAA aplican los cambios y las revisiones de seguridad más recientes. También puede editar las configuraciones existentes y actualizar la versión de Apache Airflow. En esta guía, se describen los pasos para actualizar entornos de Amazon MWAA.

Contenido

Antes de empezar

- La red de VPC que especifica para su entorno no se puede cambiar después crearlo.
- Necesita un bucket de Amazon S3 configurado para bloquear todo el acceso público, con el control de versiones del bucket activado.
- Necesita una cuenta de Cuenta de AWS con permisos para usar Amazon MWAA y un permiso en AWS Identity and Access Management (IAM) para crear roles de IAM. Si elige el modo de acceso

red privada para el servidor web de Apache Airflow, que limita el acceso de Apache Airflow dentro de su Amazon VPC, necesitará permiso en IAM para crear puntos de conexión de Amazon VPC.

- Para activar las actualizaciones rápidas del entorno, debe actualizar a la versión 2.4.3 de Apache Airflow o versiones superiores. Para actualizar la versión Airflow, consulte Cambio de versión de Apache Airflow.

Estrategia de reemplazo de procesos de trabajo

Puede elegir una estrategia de reemplazo de procesos de trabajo para controlar la forma en que Amazon MWAA gestiona los procesos de trabajado activos durante una actualización del entorno. Puede seleccionar una de las siguientes estrategias:

Actualizaciones forzosas

La actualización forzada es la estrategia de reemplazo de trabajadores predeterminada. Las actualizaciones forzadas detienen inmediatamente a todos los trabajadores activos, lo que provoca un error en las tareas en ejecución durante la actualización.

Actualizaciones rápidas

Las actualizaciones correctas permiten a los trabajadores seguir ejecutando las tareas durante un máximo de 12 horas antes de cerrar. Evita que las tareas produzcan un error debido a las interrupciones de las actualizaciones, siempre que estas finalicen en menos de 12 horas. Las nuevas tareas se redirigen a los trabajadores actualizados.

Para habilitar las actualizaciones rápidas en un entorno existente, debe completar una actualización forzosa y asegurarse de que el entorno tenga la versión 2.4.3 de Apache Airflow o una versión superior.

> **Note**
>
> Si realiza una actualización mientras el entorno está en estado `MAINTENANCE`, la estrategia de sustitución de procesos de trabajo para cualquier actualización del entorno en curso pasa de `GRACEFUL` a `FORCED`. La actualización se realizará una vez completado el mantenimiento.

Actualización de recursos del entorno

Las actualizaciones del entorno de Amazon MWAA usan la configuración del entorno existente de forma predeterminada. Para actualizar el entorno sin cambiar la configuración actual:

1. Abra la página Entornos en la consola de Amazon MWAA.
2. En la lista Environments (Entornos), elija el entorno que desee actualizar.
3. En la página del entorno, elija Editar para editar el entorno.
4. Elija Siguiente hasta llegar a la página Revisar y guardar.
5. En la página Revisar y guardar, revise los cambios y, a continuación, seleccione Guardar.

Actualización de un entorno

En la siguiente sección, se describen los pasos para actualizar entornos de Amazon MWAA.

Paso 1: especificar los detalles

Pasos para especificar los detalles del entorno

1. Abra la página Entornos en la consola de Amazon MWAA.
2. En la lista Environments (Entornos), elija el entorno que desee actualizar.
3. En la página del entorno, elija Editar para editar el entorno.
4. En la sección Detalles del entorno, para la versión Airflow, elija en la lista desplegable el nuevo número de versión de Apache Airflow al que desee actualizar el entorno.

   > **Note**
   >
   > Antes de actualizar, asegúrese de que sus DAG y otros recursos de flujo de trabajo sean compatibles con la nueva versión de Apache Airflow. Para obtener más información, consulta Cambio de versión de Apache Airflow.

5. En Código DAG de Amazon S3, especifique lo siguiente:

   a. Un bucket de S. Elija Explorar S3 y seleccione su bucket de Amazon S3 o introduzca el URI de Amazon S3.

   b. Una carpeta DAG. Elija Explorar S3 y seleccione la carpeta `dags` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.

c. Un archivo de complementos (opcional). Elija Explorar S3 y seleccione el archivo `plugins.zip` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.

d. Un archivo de requisitos (opcional). Elija Explorar S3 y seleccione el archivo `requirements.txt` en su bucket de Amazon S3 o introduzca el URI de Amazon S3.

e. Un archivo de script de inicio (opcional). Elija Browse S3 (Explorar) y seleccione el archivo de script en su bucket de Amazon S3 o introduzca el URI de Amazon S3.

6. Elija Siguiente.

Paso 2: configurar los ajustes avanzados

Configuración de opciones avanzadas

1. En Webserver access (Acceso al servidor web), seleccione el modo de acceso de Apache Airflow preferido:

   a. Una red privada. Esto limita el acceso a la interfaz de usuario de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno. Para este paso, necesita permiso para crear puntos de conexión de VPC de Amazon.

   > **Note**
   >
   > Elija la opción red privada si solo se puede acceder a la UI de Apache Airflow desde una red corporativa y no necesita acceder a repositorios públicos para cumplir con los requisitos de instalación del servidor web. Si elige este modo de acceso, deberá crear un mecanismo para acceder al servidor web de Apache Airflow en su VPC de Amazon. Para obtener más información, consulta Acceso al punto de conexión de VPC del servidor web Apache Airflow (acceso mediante red privada).

   b. Red pública. Esto permite que los usuarios con acceso a la política de IAM de su entorno accedan a la UI de Apache Airflow a través de Internet.

2. En Security groups (Grupos de seguridad), elija el grupo de seguridad que se haya usado para proteger su VPC de Amazon:

   a. Por defecto, Amazon MWAA crea un grupo de seguridad en su VPC de Amazon con reglas de entrada y salida específicas en Crear un nuevo grupo de seguridad.

b. Opcional. Desactive la casilla de verificación de Crear nuevo grupo de seguridad para seleccionar hasta 5 grupos de seguridad.

> **Note**
>
> Debe configurarse un grupo de seguridad de Amazon VPC existente con reglas de entrada y salida específicas para permitir el tráfico de red. Consulte Seguridad en la VPC en Amazon MWAA para obtener más información.

3. En Clase de entorno, elija una clase de entorno.

   Le recomendamos que elija el tamaño más pequeño necesario para soportar su carga de trabajo. Puede cambiar la clase de entorno en cualquier momento.

4. En Número máximo de procesos de trabajo, especifique el número máximo de procesos de trabajo de Apache Airflow que se ejecutarán en el entorno.

   Para obtener más información, consulta Ejemplo de caso de uso de alto rendimiento.

5. Especifique el número máximo de servidores web y el número mínimo de servidores web para configurar la forma en la que Amazon MWAA escala los servidores web de Apache Airflow en su entorno.

   Para obtener más información sobre el escalado automático del servidor web, consulte the section called “Configuración del escalado automático del servidor web”.

6. En Cifrado, elija una opción de cifrado de datos:

   a. De forma predeterminada, Amazon MWAA usa una clave de AWS propia para cifrar los datos.

   b. Opcional. Seleccione Personalizar la configuración de cifrado (avanzada) para elegir una clave de AWS KMS diferente. Si decide especificar una clave administrada por el cliente en este paso, debe especificar un identificador de clave de AWS KMS o un ARN. Amazon MWAA no admite alias ni claves multirregionales de AWS KMS. Si especificó una clave de Amazon S3 para el cifrado del servidor en su bucket de Amazon S3, debe especificar la misma clave para su entorno de Amazon MWAA.

> **Note**
>
> Debe tener permisos sobre la clave para seleccionarla en la consola de Amazon MWAA. También debe conceder permisos para que Amazon MWAA utilice la clave adjuntando la política descrita en Asociación de políticas de claves.

7. Recomendado. En Monitorización, elija una o más categorías de registro para Configuración del registro de Airflow para enviar los registros de Apache Airflow a Registros de CloudWatch:

   a. Registros de tareas de Airflow. En Nivel de registro, elija qué tipo de registro de tareas de Apache Airflow se enviará a CloudWatch Logs.

   b. Registros del servidor web de Airflow. En Log level (Nivel de registro), elija qué tipo de registro del servidor web de Apache Airflow se enviará a los registros de CloudWatch.

   c. Registros del programador de Airflow. En Nivel de registro, elija qué tipo de registro del programador de Apache Airflow se enviará a CloudWatch Logs.

   d. Registros de procesos de trabajo de Airflow. En Nivel de registro, elija qué tipo de registro de procesos de trabajo de Apache Airflow se enviará a CloudWatch Logs.

   e. Registros de procesamiento del DAG de Airflow. En Nivel de registro, elija qué tipo de registro del DAG Apache Airflow se enviará a CloudWatch Logs.

8. Opcional. Para ver las opciones de configuración de Airflow, elija Agregar una opción de configuración personalizada.

   Puede elegir de la lista desplegable sugerida de opciones de configuración de Apache Airflow para su versión de Apache Airflow o especificar opciones de configuración personalizadas. Por ejemplo, `core.default_task_retries`: `3`.

9. En Permisos, elija un rol de ejecución:

   a. Por defecto, Amazon MWAA crea un rol de ejecución en Crear un rol nuevo. Para usar esta opción, debe tener permiso para crear roles de IAM.

   b. Opcional. Elija Introduzca el ARN del rol para escribir el nombre de recurso de Amazon (ARN) de un rol de ejecución existente.

10. En Update specifications (Especificaciones de actualización), elija una Estrategia de reemplazo de procesos de trabajo para controlar cómo se gestionan los procesos de trabajo activos durante una actualización.

11. Elija Siguiente.

Paso 3: consulta y actualización

Pasos para consultar un resumen del entorno

- Consulte el resumen del entorno y elija Save (Guardar).

> **Note**
>
> Se tarda entre 20 y 30 minutos en actualizar un entorno usando actualizaciones forzosas. Las actualizaciones rápidas del entorno pueden tardar hasta 12 horas en completarse, ya que esperan a que finalicen las tareas en curso.

Cambio de versión de Apache Airflow

Amazon MWAA admite la actualización a versiones secundarias. Esto significa que puede actualizar su entorno de una versión `x.4.z` a otra `x.5.z` o de una versión `x.5.z` a otra `x.4.z`. Para realizar una actualización de una versión principal, por ejemplo, de una versión `1.y.z` a otra `2.y.z`, debe crear un entorno nuevo y migrar sus recursos. Para obtener más información sobre la actualización a una nueva versión principal de Apache Airflow, consulte Migración a un nuevo entorno de Amazon MWAA en la Guía de migración de Amazon MWAA.

Durante el proceso de actualización, Amazon MWAA captura una instantánea de los metadatos del entorno, actualiza los procesos de trabajo, los programadores y el servidor web a la nueva versión de Apache Airflow y, finalmente, restaura la base de datos de metadatos con la instantánea.

Antes de actualizar, verifique que sus DAG y otros recursos de flujo de trabajo sean compatibles con la nueva versión de Apache Airflow a la que se está actualizando. Si usa `requirements.txt` para gestionar las dependencias, también debe verificar que las dependencias que especifique en los requisitos sean compatibles con la nueva versión.

Temas

- Actualice los recursos del flujo de trabajo
- Especificación de la nueva versión

Actualice los recursos del flujo de trabajo

Siempre que cambie las versiones de Apache Airflow, asegúrese de hacer referencia a la URL `--constraint` correcta en su `requirements.txt`.

> **Warning**
>
> Si durante la actualización se especifican requisitos que no son compatibles con la versión de Apache Airflow de destino, el proceso de reversión a la versión anterior de Apache Airflow con la versión de requisitos anterior puede ser muy lento.

Migre los recursos del flujo de trabajo

1. Cree una bifurcación del repositorio aws-mwaa-docker-images y clone una copia del ejecutor local de Amazon MWAA.
2. Diríjase a la rama del repositorio aws-mwaa-docker-images que coincida con la versión a la que se está actualizando.
3. Para actualizar los `requirements.txt`, siga las prácticas recomendadas que se indican en Administrar las dependencias de Python, en la Guía del usuario de Amazon MWAA.
4. (Opcional) Para acelerar el proceso de actualización, limpie la base de datos de metadatos del entorno. Los entornos con una gran cantidad de metadatos pueden tardar mucho más en actualizarse.
5. Una vez que haya probado correctamente los recursos de flujo de trabajo, copie los DAG, `requirements.txt` y plugins en el bucket de Amazon S3 de su entorno.

Ahora puede editar el entorno, especificar una nueva versión de Apache Airflow e iniciar el procedimiento de actualización.

Especificación de la nueva versión

Cuando haya completado la actualización de los recursos de flujo de trabajo para garantizar la compatibilidad con la nueva versión de Apache Airflow, haga lo siguiente para editar los detalles del entorno y especificar la versión de Apache Airflow a la que desee actualizar.

> **Note**
>
> Al realizar una actualización, todas las tareas que se estén ejecutando en ese momento en el entorno finalizan durante el procedimiento. El procedimiento de actualización puede tardar hasta dos horas, durante las cuales el entorno no está disponible.

Especifique una versión nueva mediante la consola

1. Abra la página Entornos en la consola de Amazon MWAA.
2. En la lista Environments (Entornos), elija el entorno que desee actualizar.
3. En la página del entorno, elija Editar para editar el entorno.
4. En la sección Detalles del entorno, para la versión Airflow, elija en la lista desplegable el nuevo número de versión de Apache Airflow al que desee actualizar el entorno.
5. Elija Siguiente hasta llegar a la página Revisar y guardar.
6. En la página Revisar y guardar, revise los cambios y, a continuación, seleccione Guardar.

Al aplicar los cambios, el entorno comienza el procedimiento de actualización. Durante este proceso, el estado de su entorno indica qué acciones está llevando a cabo Amazon MWAA y si el procedimiento se ha realizado correctamente.

Si la actualización se realiza correctamente, se mostrará el estado `UPDATING` y, a continuación `CREATING_SNAPSHOT`, cuando Amazon MWAA realice la copia de seguridad de los metadatos. Por último, el estado volverá primero a `UPDATING`, y después a `AVAILABLE`, cuando finalice el procedimiento.

Si el entorno no se actualiza, el entorno mostrará el estado `ROLLING_BACK`. Si la reversión se realiza correctamente, primero se mostrará el estado `UPDATE_FAILED`, lo que indica que la actualización no se ha realizado, pero que el entorno está disponible. Si se produce un error en la reversión, aparecerá el estado `UNAVAILABLE`, lo que indica que no se puede acceder al entorno.

Uso de un script de inicio con Amazon MWAA

Un script de inicio es un script del shell (`.sh`) que uno aloja en el bucket de Amazon S3 de su entorno de forma similar a sus DAG, requisitos y complementos. Amazon MWAA ejecuta este script durante el inicio en cada componente individual de Apache Airflow (proceso de trabajo, programador

y servidor web) antes de instalar los requisitos e inicializar el proceso de Apache Airflow. Utilice un script de inicio para hacer lo siguiente:

- Instalar los tiempos de ejecución: instale los tiempos de ejecución de Linux necesarios para sus flujos de trabajo y conexiones.
- Configurar las variables de entorno: configure las variables de entorno para cada componente de Apache Airflow. Sobrescriba variables comunes como `PATH`, `PYTHONPATH` y `LD_LIBRARY_PATH`.
- Administrar las claves y los tokens: transfiera los tokens de acceso a los repositorios personalizados a `requirements.txt` y configure las claves de seguridad.

En los temas siguientes, se describe cómo configurar un script de inicio para instalar tiempos de ejecución de Linux, establecer variables de entorno y solucionar problemas relacionados con los registros de CloudWatch.

Temas

Configuración de un script de inicio

Para usar un script de inicio con su entorno Amazon MWAA existente, cargue un archivo `.sh` en el bucket Amazon S3 de su entorno. A continuación, para asociar el script al entorno, especifique lo siguiente en los detalles del entorno:

- La ruta URL de Amazon S3 al script: la ruta relativa al script alojado en su bucket, por ejemplo, `s3://mwaa-environment/startup.sh`.
- El ID de versión de Amazon S3 del script: la versión del script del shell de inicio de su bucket de Amazon S3. Debe especificar el ID de versión que Amazon S3 asigna al archivo cada vez que actualice el script. Los ID de versión son cadenas opacas unicode, codificadas en UTF-8, listas para URL que no tienen más de 1024 bytes de longitud, por ejemplo: `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`.

Para completar los pasos de esta sección, utilice el siguiente script de ejemplo. El script genera el valor asignado a `MWAA_AIRFLOW_COMPONENT`. Esta variable de entorno identifica cada componente de Apache Airflow en el que se ejecuta el script.

Copie el código y guárdelo localmente como `startup.sh`.

```
#!/bin/sh

echo "Printing Apache Airflow component"
echo $MWAA_AIRFLOW_COMPONENT
```

A continuación, cargue el script en su bucket de Amazon S3.

Consola de administración de AWS

Carga de un script del shell (consola)

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. En la lista Buckets, elija el nombre del bucket asociado a su entorno.
3. En la pestaña Objetos, elija Cargar.
4. En la página Cargar, arrastre y suelte el script del shell que ha creado.
5. Seleccione Cargar.

El script aparece en la lista objetos. Amazon S3 crea un nuevo ID de versión para el archivo. Si actualiza el script y lo vuelve a cargar con el mismo nombre de archivo, se asigna un nuevo ID de versión al archivo.

AWS CLI

Creación y carga de un script del shell (CLI)

1. Abra un nuevo símbolo del sistema y ejecute el comando `ls` de Amazon S3 para enumerar e identificar el bucket asociado a su entorno.

```
aws s3 ls
```

2. Navegue hasta la carpeta en la que guardó el script del shell. Utilice `cp` en una nueva ventana de símbolo del sistema para cargar el script en su bucket. Sustituya *amazn-s3-demo-bucket* por su información.

```
aws s3 cp startup.sh s3://amzn-s3-demo-bucket/startup.sh
```

Si se ejecuta correctamente, Amazon S3 muestra la ruta URL del objeto:

```
upload: ./startup.sh to s3://amzn-s3-demo-bucket/startup.sh
```

3. Utilice el siguiente comando para recuperar el último ID de versión del script.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix startup --
query 'Versions[?IsLatest].[VersionId]' --output text
```

```
BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Este identificador de versión se especifica al asociar el script a un entorno.

Ahora, asocie el script a su entorno.

Consola de administración de AWS

Asociación del script a un entorno (consola)

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione la fila del entorno que desee actualizar y, a continuación, elija Editar.
3. En la página Especificar detalles, en Archivo de script de inicio (opcional), introduzca la URL de Amazon S3 del script, por ejemplo: `s3://amzn-s3-demo-bucket/startup-sh.`.
4. Elija la versión más reciente en la lista desplegable o elija Examinar S3 para buscar el script.
5. Elija Siguiente y a continuación, vaya a la página Revisar y guardar.
6. Revise los cambios y, a continuación, seleccione Guardar.

Las actualizaciones del entorno pueden tardar entre 10 y 30 minutos. Amazon MWAA ejecuta el script de inicio a medida que se reinicia cada componente del entorno.

AWS CLI

Asociación del script a un entorno (CLI)

- Abra un símbolo del sistema y utilice `update-environment` para especificar la URL de Amazon S3 y el ID de versión del script.

```
aws mwaa update-environment \
--name your-mwaa-environment \
--startup-script-s3-path startup.sh \
--startup-script-s3-object-version BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Si el proceso se ha realizado correctamente, Amazon MWAA devuelve el nombre de recurso de Amazon (ARN) del entorno:

```
arn:aws:airflow:us-west-2:123456789012:environment/your-mwaa-environment
```

La actualización del entorno puede tardar entre 10 y 30 minutos. Amazon MWAA ejecuta el script de inicio a medida que se reinicia cada componente del entorno.

Por último, recupere los eventos de registro para comprobar que el script funciona según lo previsto. Al activar el registro para cada componente de Apache Airflow, Amazon MWAA crea un grupo de registros y un flujo de registros nuevos. Para obtener más información, consulte lostipos de registro de Apache Airflow.

Consola de administración de AWS

Pasos para obtener el flujo de registro de Apache Airflow (consola)

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione su entorno.
3. En el panel Monitoring (Monitorización), elija el grupo cuyos registros desea ver, por ejemplo, el grupo de registros del programador de Airflow.
4. En la consola de CloudWatch, en la lista de flujos de registro, elija un flujo con el siguiente prefijo: `startup_script_execution_ip`.

5. En el panel Log events (Eventos de registro), verá el resultado del comando que imprime el valor de `MWAA_AIRFLOW_COMPONENT`. Por ejemplo, en el caso de los registros del programador, verá lo siguiente:

```
Printing Apache Airflow component
        scheduler
        Finished running startup script. Execution time: 0.004s.
        Running verification
        Verification completed
```

Puede repetir los pasos anteriores para ver los registros de los procesos de trabajo y del servidor web.

Instalación de los tiempos de ejecución de Linux mediante un script de inicio

Utilice un script de inicio para actualizar el sistema operativo de un componente de Apache Airflow e instale bibliotecas de tiempo de ejecución adicionales para utilizarlas con sus flujos de trabajo. Por ejemplo, se ejecuta el siguiente script `yum update` para actualizar el sistema operativo.

Cuando se ejecuta `yum update` en un script de inicio, debe excluir Python usando `--exclude=python*` como se muestra en el ejemplo. Para que su entorno se ejecute, Amazon MWAA instala una versión específica de Python compatible con su entorno. Por lo tanto, no puede actualizar la versión de Python del entorno mediante un script de inicio.

```
#!/bin/sh

echo "Updating operating system"
sudo yum update -y --exclude=python*
```

Para instalar tiempos de ejecución en un componente específico de Apache Airflow, utilice `MWAA_AIRFLOW_COMPONENT` e `if` e instrucciones condicionales `fi`. En este ejemplo, se ejecuta un único comando para instalar la biblioteca `libaio` en el programador y el proceso de trabajo, pero no en el servidor web.

> **Important**
>
> - Si ha configurado un servidor web privado, debe usar la siguiente condición o proporcionar todos los archivos de instalación de forma local para evitar que agota el tiempo de espera durante la instalación.
> - Utilice `sudo` para ejecutar operaciones que requieren privilegios administrativos.

```
#!/bin/sh

if [[ "${MWAA_AIRFLOW_COMPONENT}" != "webserver" ]]
  then
    sudo yum -y install libaio

fi
```

Puede usar un script de inicio para comprobar la versión de 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"
```

Amazon MWAA no admite la anulación de la versión predeterminada de Python, ya que esto podría provocar incompatibilidades con las bibliotecas de Apache Airflow instaladas.

Configuración de las variables de entorno mediante un script de inicio

Utilice scripts de inicio para establecer variables de entorno y modificar las configuraciones de Apache Airflow. En el siguiente ejemplo, se define una nueva variable `ENVIRONMENT_STAGE`. Puede hacer referencia a esta variable en un DAG o en sus módulos personalizados.

```
#!/bin/sh

export ENVIRONMENT_STAGE="development"
echo "$ENVIRONMENT_STAGE"
```

Utilice scripts de inicio para sobrescribir variables comunes de Apache Airflow o del sistema. Por ejemplo, configure `LD_LIBRARY_PATH` para indicar a Python que busque binarios en la ruta que especifique. Esto le permite proporcionar binarios personalizados para sus flujos de trabajo mediante complementos:

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/local/airflow/plugins/your-custom-binary
```

Variables de entorno reservadas

Amazon MWAA reserva un conjunto de variables de entorno críticas. Si sobrescribe una variable reservada, Amazon MWAA la restaura a su valor predeterminado. A continuación se enumeran las variables reservadas:

- `MWAA__AIRFLOW__COMPONENT`: se utiliza para identificar el componente Apache Airflow con uno de los siguientes valores: `scheduler`, `worker` o `webserver`.
- `AIRFLOW__WEBSERVER__SECRET_KEY`: la clave secreta utilizada para firmar de forma segura las cookies de sesión en el servidor web Apache Airflow.
- `AIRFLOW__CORE__FERNET_KEY`: la clave utilizada para cifrar y descifrar los datos confidenciales almacenados en la base de datos de metadatos, por ejemplo, las contraseñas de conexión.
- `AIRFLOW_HOME`: la ruta al directorio principal de Apache Airflow, donde los archivos de configuración y los archivos DAG se almacenan localmente.
- `AIRFLOW__CELERY__BROKER_URL`: la URL del agente de mensajes utilizado para la comunicación entre el programador de Apache Airflow y los nodos de trabajo de Celery.
- `AIRFLOW__CELERY__RESULT_BACKEND`: la URL de la base de datos utilizada para almacenar los resultados de las tareas de Celery.
- `AIRFLOW__CORE__EXECUTOR`: la clase de ejecutor que debe usar Apache Airflow. En Amazon MWAA, se trata de `CeleryExecutor`
- `AIRFLOW__CORE__LOAD_EXAMPLES`: se utiliza para activar o desactivar la carga de ejemplos de DAG.
- `AIRFLOW__METRICS__METRICS_BLOCK_LIST`: se utiliza para gestionar qué métricas de Apache Airflow emite y captura Amazon MWAA en CloudWatch.
- `SQL_ALCHEMY_CONN`: la cadena de conexión de la base de datos de RDS para PostgreSQL utilizada para almacenar los metadatos de Apache Airflow en Amazon MWAA.

- AIRFLOW__CORE__SQL_ALCHEMY_CONN: se utiliza con el mismo propósito que SQL_ALCHEMY_CONN, pero siguiendo la nueva convención de nomenclatura de Apache Airflow.
- AIRFLOW__CELERY__DEFAULT_QUEUE: la cola predeterminada para las tareas de Celery en Apache Airflow.
- AIRFLOW__OPERATORS__DEFAULT_QUEUE: la cola predeterminada para las tareas que utilizan operadores específicos de Apache Airflow.
- AIRFLOW_VERSION: la versión de Apache Airflow instalada en el entorno Amazon MWAA.
- AIRFLOW_CONN_AWS_DEFAULT: las credenciales de AWS predeterminadas que se utilizan para la integración con otros servicios de AWS.
- AWS_DEFAULT_REGION: establece la Región de AWS predeterminada que se usa con las credenciales predeterminadas para la integración con otros servicios de AWS.
- AWS_REGION: si se define, esta variable de entorno invalida los valores de la variable AWS_DEFAULT_REGION de entorno y la región de configuración del perfil.
- PYTHONUNBUFFERED: se utiliza para enviar flujos stdout y stderr a registros de contenedor.
- AIRFLOW__METRICS__STATSD_ALLOW_LIST: se utiliza para configurar una lista de permitidos de prefijos separados por comas para enviar las métricas que comienzan con los elementos de la lista.
- AIRFLOW__METRICS__STATSD_ON: activa el envío de métricas a StatsD.
- AIRFLOW__METRICS__STATSD_HOST: se utiliza para conectarse al daemon StatSD.
- AIRFLOW__METRICS__STATSD_PORT: se utiliza para conectarse al daemon StatSD.
- AIRFLOW__METRICS__STATSD_PREFIX: se utiliza para conectarse al daemon StatSD.
- AIRFLOW__CELERY__WORKER_AUTOSCALE: establece la simultaneidad máxima y mínima.
- AIRFLOW__CORE__DAG_CONCURRENCY: establece el número de instancias de tareas que el programador puede ejecutar simultáneamente en un DAG.
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG: establece el número máximo de tareas activas por DAG.
- AIRFLOW__CORE__PARALLELISM: define el número máximo de instancias de tareas que se pueden ejecutar simultáneamente.
- AIRFLOW__SCHEDULER__PARSING_PROCESSES: establece el número máximo de procesos analizados por el programador para programar los DAG.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT: define, en número de segundos, el tiempo de espera para que un proceso de trabajo confirme la tarea antes de que el mensaje se entregue a otro proceso de trabajo.

- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION`: establece la Región de AWS para el transporte de Celery subyacente.
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES`: establece la cola para el transporte de Celery subyacente.
- `AIRFLOW_SCHEDULER_ALLOWED_RUN_ID_PATTERN`: se utiliza para verificar la validez de la entrada del `run_id` parámetro al activar un DAG.
- `AIRFLOW__WEBSERVER__BASE_URL`: la URL del servidor web utilizado para alojar la interfaz de usuario de Apache Airflow.
- `PYTHONPATH` (solo para Apache Airflow v2.9 y versiones posteriores): reservado por Amazon MWAA para garantizar que todas las funcionalidades básicas del entorno funcionen correctamente.

> **Note**
>
> Para las versiones de Apache Airflow anteriores a la v2.9, `PYTHONPATH` es una variable de entorno sin reservas.

Variables de entorno sin reserva

Puede utilizar un script de inicio para sobrescribir las variables de entorno no reservadas. En la siguiente lista, se ofrecen algunas de las variables comunes:

- `PATH`: especifica una lista de directorios en los que el sistema operativo busca archivos ejecutables y scripts. Cuando se ejecuta un comando en la línea de comandos, el sistema comprueba los directorios en `PATH` para buscar y ejecutar el comando. Al crear tareas u operadores personalizados en Apache Airflow, es posible que necesite recurrir a scripts o ejecutables externos. Si los directorios que contienen estos archivos no se encuentran en los valores especificados en la variable `PATH`, las tareas no se ejecutarán si el sistema no puede localizarlos. Al añadir los directorios adecuados a `PATH`, las tareas de Apache Airflow pueden buscar y ejecutar los ejecutables necesarios.
- `PYTHONPATH`: utilizada por el intérprete de Python para determinar en qué directorios buscar los módulos y paquetes importados. Es una lista de directorios que puede añadir a la ruta de búsqueda predeterminada. Esto permite al intérprete buscar y cargar bibliotecas de Python no incluidas en la biblioteca estándar o instaladas en los directorios del sistema. Use esta variable para agregar sus módulos y paquetes de Python personalizados y úselos con sus DAG.

> **Note**
>
> Para Apache Airflow v2.9 y versiones posteriores, `PYTHONPATH` es una variable de entorno reservada.

- `LD_LIBRARY_PATH`: variable de entorno utilizada por el enlazador y el cargador dinámicos de Linux para buscar y cargar bibliotecas compartidas. Especifica una lista de directorios que contienen bibliotecas compartidas, en las que se busca antes que en los directorios predeterminados de las bibliotecas del sistema. Utilice esta variable para especificar los binarios personalizados.
- `CLASSPATH`: utilizada por el Java Runtime Environment (JRE) y el kit de desarrollo de Java Development Kit (JDK) para localizar y cargar clases, bibliotecas y recursos de Java en tiempo de ejecución. Es una lista de directorios, archivos JAR y archivos ZIP que contienen código Java compilado.

Trabajo con DAG en Amazon MWAA

Para ejecutar grafos acíclicos dirigidos (DAG) en un entorno de Amazon Managed Workflows para Apache Airflow, debe copiar los archivos al bucket de almacenamiento de Amazon S3 adjunto a su entorno y, a continuación, informar a Amazon MWAA de dónde se encuentran los DAG y los archivos auxiliares en la consola de Amazon MWAA. Amazon MWAA se encarga de sincronizar los DAG entre los procesos de trabajo, los programadores y el servidor web. En esta guía, se describe cómo añadir o actualizar sus DAG e instalar complementos personalizados y dependencias de Python en un entorno Amazon MWAA.

Temas

Descripción general del bucket de Amazon S3

Los buckets de Amazon S3 para un entorno Amazon MWAA deben tener el acceso público bloqueado. De forma predeterminada, todos los recursos de Amazon S3 (buckets, objetos y subrecursos relacionados como, por ejemplo, la configuración del ciclo de vida) son privados.

- Solo el propietario del recurso, la Cuenta de AWS que creó el bucket, puede acceder a dicho recurso. El propietario del recurso (por ejemplo, su administrador) puede conceder permisos de acceso a terceros escribiendo una política de control de acceso.
- La política de acceso que configure debe tener permiso para añadir DAG, complementos personalizados en `plugins.zip` y dependencias de Python en `requirements.txt` a su bucket de Amazon S3. Para ver un ejemplo de política que contenga los permisos necesarios, consulte AmazonMWAAFullConsoleAccess.

Un bucket de Amazon S3 para un entorno Amazon MWAA debe tener el control de versiones habilitado. Cuando el control de versiones de buckets de Amazon S3 está habilitado, cada vez que se crea una nueva versión, se crea una nueva copia.

- El control de versiones está habilitado para los complementos personalizados de un `plugins.zip` y para dependencias de Python de un `requirements.txt` de su bucket de Amazon S3.
- Debe especificar la versión de un `plugins.zip` y un `requirements.txt` en la consola de Amazon MWAA cada vez que se actualicen estos archivos en su bucket de Amazon S3.

Cómo añadir o actualizar DAG

Los gráficos acíclicos dirigidos (DAG) se definen dentro de un archivo de Python que define la estructura del DAG como código. Para cargar DAG en su entorno, puede usar la AWS CLI o la consola de Amazon S3. En esta página, se describen los pasos para añadir o actualizar los DAG de Apache Airflow en su entorno de Amazon Managed Workflows para Apache Airflow mediante la carpeta `dags` de su bucket de Amazon S3.

Secciones

Requisitos previos

Para poder llevar a cabo los pasos de esta página, necesitará lo siguiente.

- Permisos: el administrador debe haber concedido a su Cuenta de AWS acceso a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno. Además, su rol de ejecución debe permitir que su entorno de Amazon MWAA acceda a los recursos de AWS que utiliza su entorno.
- Acceso: si tiene que acceder a los repositorios públicos para instalar dependencias directamente en el servidor web, su entorno debe estar configurado con acceso a un servidor web de red

pública. Para obtener más información, consulta the section called “Modos de acceso de Apache Airflow”.

- Configuración de Amazon S3: el bucket de Amazon S3 que se utiliza para almacenar los DAG, los complementos personalizados en `plugins.zip` y las dependencias de Python en `requirements.txt` deben estar configurados con el acceso público bloqueado y el control de versiones activado.

Funcionamiento

Un gráfico acíclico dirigido (DAG) se define dentro de un archivo de Python que define la estructura del DAG como código. Consta de lo siguiente:

- Una definición de DAG.
- Operadores que describen cómo ejecutar el DAG y las tareas que se van a ejecutar.
- Relaciones entre los operadores que describen el orden en el que se ejecutan las tareas.

Para ejecutar una plataforma Apache Airflow en un entorno Amazon MWAA, debe copiar la definición del DAG en la carpeta `dags` del bucket de almacenamiento. Por ejemplo, la carpeta DAG de su bucket de almacenamiento debe tener el siguiente aspecto:

Example Carpeta de DAG

```
dags/
# dag_def.py
```

Amazon MWAA sincroniza automáticamente los objetos nuevos y modificados de su bucket de Amazon S3 con la carpeta `/usr/local/airflow/dags` del programador y los contenedores de procesos de trabajo de Amazon MWAA cada 30 segundos, lo que preserva la jerarquía de archivos de la fuente de Amazon S3, independientemente del tipo de archivo. El tiempo que tardan los nuevos DAG en aparecer en la UI de Apache Airflow depende de `scheduler.dag_dir_list_interval`. Los cambios en los DAG existentes se recogerán en el siguiente ciclo de procesamiento de los DAG.

> **Note**
>
> No es necesario incluir el archivo de configuración `airflow.cfg` en la carpeta del DAG. Puede anular las configuraciones predeterminadas de Apache Airflow desde la consola

de Amazon MWAA. Para obtener más información, consulta Uso de las opciones de configuración de Apache Airflow en Amazon MWAA.

¿Qué ha cambiado?

Para revisar los cambios de una versión específica de Apache Airflow, consulte la página de notas de la versión.

- Configuraciones de Apache Airflow v3: referencia de configuración
- Información sobre la interfaz pública de Apache Airflow v2: Interfaz pública de Airflow

Pruebas de los DAG mediante la utilidad de la CLI de Amazon MWAA

- La utilidad de la interfaz de la línea de comandos (CLI) replica entornos en Amazon Managed Workflows para Apache Airflow de forma local.
- La CLI crea localmente una imagen de contenedor de Docker similar a una imagen de producción de Amazon MWAA. Esto le permite ejecutar un entorno local de Apache Airflow para desarrollar y probar los DAG, los complementos personalizados y las dependencias antes de implementarlos en Amazon MWAA.
- Para ejecutar la CLI, consulte aws-mwaa-docker-images en GitHub.

Cómo cargar el código DAG en Amazon S3

Puede usar la consola de Amazon S3 o la AWS Command Line Interface (AWS CLI) para cargar un código DAG a su bucket de Amazon S3. En los siguientes pasos se supone que está cargando el código (`.py`) a una carpeta con el nombre `dags` en su bucket de Amazon S3.

Uso de AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Carga mediante la AWS CLI

1. Use el siguiente comando para obtener una lista de todos los buckets de Amazon S3.

```
aws s3 ls
```

2. Utilice el comando siguiente para enumerar los archivos y las carpetas del bucket de Amazon S3 para su entorno.

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

3. El siguiente comando carga el archivo `dag_def.py` en una carpeta `dags`.

```
aws s3 cp dag_def.py s3://amzn-s3-demo-bucket/dags/
```

   Si aún no existe una carpeta con el nombre `dags` en su bucket de Amazon S3, este comando crea la carpeta `dags` y carga el archivo con el nombre `dag_def.py` en la nueva carpeta.

Uso de la consola de Amazon S3

La consola de Amazon S3 es una interfaz de usuario basada en la web que le permite crear y administrar los recursos de su bucket de Amazon S3. En los siguientes pasos se supone que tiene una carpeta DAG denominada `dags`.

Carga del contenido usando la consola de Amazon S3

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione el enlace del bucket de S3 en el panel de códigos de DAG en S3 para abrir el bucket de almacenamiento en la consola.
4. Elija la carpeta `dags`.
5. Seleccione Cargar.
6. Elija Añadir archivo.
7. Seleccione la copia local de su `dag_def.py`, elija Cargar.

Especificación de la ruta a su carpeta DAG en la consola Amazon MWAA (la primera vez)

En los pasos siguientes, se supone que está especificando la ruta de una carpeta del bucket de Amazon S3 denominada `dags`.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Elija el entorno en el que desee ejecutar los DAG.
3. Seleccione Editar.
4. En el panel DAG code in Amazon S3 (Código DAG en Amazon S3), elija Browse S3 (Navegar en S3) junto al campo DAG folder (Carpeta de DAG).
5. Seleccione su carpeta `dags`.
6. Seleccione Elegir.
7. Seleccione Siguiente, Actualizar entorno.

Visualización de cambios en la UI de Apache Airflow

Necesita permisos de Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess para su cuenta de Cuenta de AWS en AWS Identity and Access Management (IAM) para ver la UI de Apache Airflow.

Pasos para acceder a la interfaz de usuario de Apache Airflow

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Abrir interfaz de usuario de Airflow.

Siguientes pasos

Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

Instalación de complementos personalizados

Amazon Managed Workflows para Apache Airflow es compatible con el administrador de complementos integrado en Apache Airflow, lo que le permite utilizar operadores, enlaces, sensores o interfaces personalizados de Apache Airflow. En esta página se describen los pasos para instalar los complementos personalizados de Apache Airflow en su entorno de Amazon Managed Workflows usando un archivo `plugins.zip`.

Contenido

Requisitos previos

Para poder llevar a cabo los pasos de esta página, necesitará lo siguiente.

- Permisos: el administrador debe haber concedido a su Cuenta de AWS acceso a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno. Además, su rol de ejecución debe permitir que su entorno de Amazon MWAA acceda a los recursos de AWS que utiliza su entorno.
- Acceso: si tiene que acceder a los repositorios públicos para instalar dependencias directamente en el servidor web, su entorno debe estar configurado con acceso a un servidor web de red pública. Para obtener más información, consulta the section called “Modos de acceso de Apache Airflow”.
- Configuración de Amazon S3: el bucket de Amazon S3 que se utiliza para almacenar los DAG, los complementos personalizados en `plugins.zip` y las dependencias de Python en `requirements.txt` deben estar configurados con el acceso público bloqueado y el control de versiones activado.

Funcionamiento

Para ejecutar complementos personalizados en su entorno, debe hacer tres cosas:

1. Cree un archivo `plugins.zip` local.
2. Cargue el archivo `plugins.zip` en su bucket de Amazon S3.
3. Especifique la versión de este archivo en el campo Archivo de complementos de la consola de Amazon MWAA.

> **Note**
>
> Si es la primera vez que sube un archivo `plugins.zip` a su bucket de Amazon S3, también tendrá que especificar la ruta al archivo en la consola de Amazon MWAA. Solo necesita realizar este paso una vez.

Cuándo usar los complementos

Los complementos solo son necesarios para ampliar la interfaz de usuario de Apache Airflow, como se describe en la documentación de Apache Airflow. Los operadores personalizados se pueden colocar directamente en la carpeta `/dags` junto con el código DAG.

Si necesita crear sus propias integraciones con sistemas externos, colóquelas en la carpeta `/dags` o en una de sus subcarpetas, pero no en la carpeta `plugins.zip`. En Apache Airflow 2.x, los complementos se utilizan principalmente para ampliar la interfaz de usuario.

Del mismo modo, no se deben colocar otras dependencias en `plugins.zip`. En su lugar, se pueden almacenar en una ubicación dentro de la carpeta `/dags` de Amazon S3, donde se sincronizarán con cada contenedor de Amazon MWAA antes de que se inicie Apache Airflow.

> **Note**
>
> Cualquier archivo de la carpeta `/dags` o en `plugins.zip` que no defina explícitamente un objeto DAG de Apache Airflow debe figurar en un archivo `.airflowignore`.

Información general de los complementos personalizados

El administrador de complementos integrado de Apache Airflow puede integrar características externas en su núcleo simplemente colocando archivos en una carpeta `$AIRFLOW_HOME/plugins`. Le permite usar operadores, enlaces, sensores o interfaces personalizados de Apache Airflow. La siguiente sección proporciona un ejemplo de estructuras de directorios planas y anidadas en un entorno de desarrollo local y las instrucciones de importación resultantes, que determinan la estructura de directorios dentro de un plugins.zip.

Directorio de complementos personalizados y límites de tamaño

El programador y los procesos de trabajo de Apache Airflow buscan complementos personalizados durante el inicio en el contenedor de Fargate administrado por AWS para su entorno en `/usr/local/airflow/plugins/*`.

- Estructura de directorios. La estructura de directorios (en `/*`) se basa en el contenido del archivo `plugins.zip`. Por ejemplo, si su `plugins.zip` contiene el directorio `operators` como directorio de nivel superior, el directorio se extraerá en `/usr/local/airflow/plugins/operators` dentro de su entorno.
- Límite de tamaño. Se recomienda un archivo `plugins.zip` de menos de 1 GB. Cuanto mayor sea el tamaño del archivo `plugins.zip`, mayor será el tiempo de inicio en un entorno. Aunque Amazon MWAA no limita el tamaño de un archivo `plugins.zip` de forma explícita, si las dependencias no se pueden instalar en diez minutos, el servicio Fargate agotará el tiempo de espera e intentará revertir el entorno a un estado estable.

> **Note**
>
> Para los entornos que usan Apache Airflow v2.0.2, Amazon MWAA limita el tráfico saliente en el servidor web Apache Airflow y no le permite instalar complementos ni dependencias de Python directamente en el servidor web. A partir de la versión 2.2.2 de Apache Airflow, Amazon MWAA puede instalar complementos y dependencias directamente en el servidor web.

Ejemplos de complementos personalizados

En la siguiente sección, se usa un código de muestra de la guía de referencia de Apache Airflow para mostrar cómo estructurar su entorno de desarrollo local.

Ejemplo de uso de una estructura de directorios plana en plugins.zip

Apache Airflow v3

En el siguiente ejemplo, se muestra un archivo `plugins.zip` con una estructura de directorios plana para Apache Airflow v3.

Example directorio plano con el complement plugins.zip **`PythonVirtualenvOperator`**

En el siguiente ejemplo, se muestra el árbol principal de un archivo plugins.zip para el complemento personalizado `PythonVirtualenvOperator` en Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

En el siguiente ejemplo, se muestra el complemento personalizado `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 v2

El siguiente ejemplo, se muestra un archivo `plugins.zip` con una estructura de directorios plana para Apache Airflow v2.

Example directorio plano con el complement plugins.zip **`PythonVirtualenvOperator`**

En el siguiente ejemplo, se muestra el árbol principal de un archivo plugins.zip para el complemento personalizado `PythonVirtualenvOperator` en Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

En el siguiente ejemplo, se muestra el complemento personalizado `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'
```

Ejemplo de uso de una estructura de directorios anidada en plugins.zip

Apache Airflow v3

En el siguiente ejemplo, se muestra un archivo `plugins.zip` con directorios independientes para `hooks`, `operators` y un directorio `sensors`.

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
```

En el siguiente ejemplo, se muestran las instrucciones de importación en el DAG (carpeta de DAG) que usa los complementos 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]
```

En los siguientes ejemplos, se muestran cada una de las instrucciones de importación necesarias en los archivos de complementos 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 los pasos que se indican en Probar complementos personalizados con la utilidad CLI de Amazon MWAA y, a continuación, consulte cómo crear un archivo plugins.zip para comprimir el contenido en su directorio `plugins`. Por ejemplo, `cd plugins`.

Apache Airflow v2

En el siguiente ejemplo, se muestra un archivo `plugins.zip` con directorios independientes para `hooks`, `operators` y un directorio `sensors`.

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
```

En el siguiente ejemplo, se muestran las instrucciones de importación en el DAG (carpeta de DAG) que usa los complementos 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]
```

En los siguientes ejemplos, se muestran cada una de las instrucciones de importación necesarias en los archivos de complementos 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 los pasos que se indican en Probar complementos personalizados con la utilidad CLI de Amazon MWAA y, a continuación, consulte cómo crear un archivo plugins.zip para comprimir el contenido en su directorio `plugins`. Por ejemplo, `cd plugins`.

Crear un archivo plugins.zip

En los pasos siguientes se describen los pasos que recomendamos para crear un archivo plugins.zip de forma local.

Paso uno: pruebe los complementos personalizados con la utilidad CLI de Amazon MWAA

- La utilidad de la interfaz de la línea de comandos (CLI) replica entornos en Amazon Managed Workflows para Apache Airflow de forma local.
- La CLI crea localmente una imagen de contenedor de Docker similar a una imagen de producción de Amazon MWAA. Esto le permite ejecutar un entorno local de Apache Airflow para desarrollar y probar los DAG, los complementos personalizados y las dependencias antes de implementarlos en Amazon MWAA.
- Para ejecutar la CLI, consulte aws-mwaa-docker-images en GitHub.

Paso dos: crear el archivo plugins.zip

Puede utilizar una utilidad de archivado ZIP integrada o cualquier otra utilidad ZIP (como 7zip) para crear un archivo .zip.

> **Note**
>
> La utilidad zip integrada para el sistema operativo Windows puede agregar subcarpetas al crear un archivo .zip. Le recomendamos comprobar el contenido del archivo plugins.zip antes de subirlo a su bucket de Amazon S3 para asegurarse de que no se hayan añadido directorios adicionales.

1. Cambie los directorios a su directorio local de complementos de Airflow. Por ejemplo:

```
myproject$ cd plugins
```

2. Ejecute el siguiente comando para asegurarse de que el contenido tiene permisos de ejecución (solo para macOS y Linux).

```
plugins$ chmod -R 755 .
```

3. Comprima el contenido de la carpeta `plugins`.

```
plugins$ zip -r plugins.zip .
```

Cómo cargar `plugins.zip` a Amazon S3

Puede usar la consola de Amazon S3 o la AWS Command Line Interface (AWS CLI) para cargar un archivo `plugins.zip` a su bucket de Amazon S3.

Uso de AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Carga mediante la AWS CLI

1. En el símbolo del sistema, vaya hasta el directorio en el que está almacenado el archivo `plugins.zip`. Por ejemplo:

```
cd plugins
```

2. Use el siguiente comando para obtener una lista de todos los buckets de Amazon S3.

```
aws s3 ls
```

3. Utilice el comando siguiente para enumerar los archivos y las carpetas del bucket de Amazon S3 para su entorno.

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

4. Utilice el siguiente comando para cargar el archivo `plugins.zip` en el bucket de Amazon S3 para su entorno.

```
aws s3 cp plugins.zip s3://amzn-s3-demo-bucket/plugins.zip
```

Uso de la consola de Amazon S3

La consola de Amazon S3 es una interfaz de usuario basada en la web que le permite crear y administrar los recursos de su bucket de Amazon S3.

Carga del contenido usando la consola de Amazon S3

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione el enlace del bucket S3 en el panel de códigos de DAG en S3 para abrir el bucket de almacenamiento en la consola.
4. Seleccione Cargar.
5. Elija Añadir archivo.
6. Seleccione la copia local de su `plugins.zip`, elija Cargar.

Instalación de complementos personalizados en su entorno

En esta sección se describe cómo instalar los complementos personalizados que ha cargado en su bucket de Amazon S3 especificando la ruta al archivo plugins.zip y especificando la versión del archivo plugins.zip cada vez que se actualiza el archivo zip.

Especificación de la ruta a **`plugins.zip`** en la consola MWAA de Amazon (la primera vez)

Si es la primera vez que sube un archivo `plugins.zip` a su bucket de Amazon S3, también tendrá que especificar la ruta al archivo en la consola de Amazon MWAA. Solo necesita realizar este paso una vez.

1. Abra la página Entornos en la consola de Amazon MWAA.

2. Seleccione un entorno.
3. Elija Editar.
4. En el panel de códigos DAG en Amazon S3, elija Browse S3 (Explorar S3) junto al campo Plugins file - optional (Archivo de complementos: opcional).
5. Seleccione el archivo `plugins.zip` en su bucket de Amazon S3.
6. Seleccione Elegir.
7. Seleccione Siguiente, Actualizar entorno.

Especificación de la versión de **`plugins.zip`** en la consola de Amazon MWAA

Debe especificar la versión de su archivo `plugins.zip` en la consola de Amazon MWAA cada vez que cargue una nueva versión de su `plugins.zip` en su bucket de Amazon S3.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Editar.
4. En el panel Código de DAG en Amazon S3, elija una versión de `plugins.zip` de la lista desplegable.
5. Elija Siguiente.

Ejemplos de casos de uso de plugins.zip

- Obtenga información sobre cómo crear un complemento personalizado en Creación de un complemento con Apache Hive y Hadoop.
- Obtenga información sobre cómo crear un complemento personalizado en Complemento personalizado para parchear PythonVirtualEnvOperator .
- Obtenga información sobre cómo crear un complemento personalizado en Complemento personalizado con Oracle.
- Obtenga información sobre cómo crear un complemento personalizado en the section called “Cómo cambiar la zona horaria de un DAG”.

Siguientes pasos

Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

Instalación de dependencias de Python

Una dependencia de Python es cualquier paquete o distribución que no esté incluido en la instalación base de Apache Airflow para su versión de Apache Airflow en su entorno de Amazon Managed Workflows para Apache Airflow. En esta página, se describen los pasos para instalar las dependencias de Python de Apache Airflow en su entorno de Amazon MWAA mediante un archivo `requirements.txt` en su bucket de Amazon S3.

Contenido

Requisitos previos

Para poder llevar a cabo los pasos de esta página, necesitará lo siguiente.

- Permisos: el administrador debe haber concedido a su Cuenta de AWS acceso a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno. Además, su rol de ejecución debe permitir que su entorno de Amazon MWAA acceda a los recursos de AWS que utiliza su entorno.
- Acceso: si tiene que acceder a los repositorios públicos para instalar dependencias directamente en el servidor web, su entorno debe estar configurado con acceso a un servidor web de red pública. Para obtener más información, consulta the section called “Modos de acceso de Apache Airflow”.
- Configuración de Amazon S3: el bucket de Amazon S3 que se utiliza para almacenar los DAG, los complementos personalizados en `plugins.zip` y las dependencias de Python en `requirements.txt` deben estar configurados con el acceso público bloqueado y el control de versiones activado.

Funcionamiento

En Amazon MWAA, para instalar todas las dependencias de Python, debe cargar un archivo `requirements.txt` en su bucket de Amazon S3 y, a continuación, especificar la versión del archivo en la consola de Amazon MWAA cada vez que actualice el archivo. Amazon MWAA ejecuta `pip3 install -r requirements.txt` para instalar las dependencias de Python en el programador de Apache Airflow y en cada uno de los procesos de trabajo.

Para ejecutar las dependencias de Python en su entorno, debe hacer tres cosas:

1. Cree un archivo `requirements.txt` local.
2. Cargue el `requirements.txt` local en su bucket de Amazon S3.
3. Especifique la versión de este archivo en el campo Archivo de requisitos de la consola de Amazon MWAA.

> **Note**
>
> Si es la primera vez que crea y sube un archivo `requirements.txt` a su bucket de Amazon S3, también tendrá que especificar la ruta al archivo en la consola de Amazon MWAA. Solo necesita realizar este paso una vez.

Descripción general de las dependencias de Python

Puede instalar los extras de Apache Airflow y otras dependencias de Python desde el Python Package Index (PyPI.org), ruedas de Python (`.whl`) o las dependencias de Python alojadas en un repositorio privado compatible con PyPI/PEP-503 en su entorno.

Límites de ubicación y tamaño de las dependencias de Python

El programador de Apache Airflow y los procesos de trabajo buscan los paquetes en el archivo `requirements.txt` y los paquetes se instalan en el entorno en `/usr/local/airflow/.local/bin`.

- Límite de tamaño. Recomendamos un archivo `requirements.txt` que haga referencia a bibliotecas cuyo tamaño combinado sea inferior a 1 GB. Cuantas más bibliotecas necesite instalar Amazon MWAA, mayor será el tiempo de inicio de un entorno. Aunque Amazon MWAA no limita el tamaño de las bibliotecas instaladas de forma explícita, si las dependencias no se pueden instalar en diez minutos, el servicio Fargate agotará el tiempo de espera e intentará revertir el entorno a un estado estable.

Creación de un archivo requirements.txt

En los pasos siguientes se describen los pasos que recomendamos para crear un archivo requirements.txt de forma local.

Paso uno: probar las dependencias de Python con la utilidad CLI de Amazon MWAA

- La utilidad de la interfaz de la línea de comandos (CLI) replica entornos en Amazon Managed Workflows para Apache Airflow de forma local.
- La CLI crea localmente una imagen de contenedor de Docker similar a una imagen de producción de Amazon MWAA. Esto le permite ejecutar un entorno local de Apache Airflow para desarrollar y probar los DAG, los complementos personalizados y las dependencias antes de implementarlos en Amazon MWAA.
- Para ejecutar la CLI, consulte aws-mwaa-docker-images en GitHub.

Paso dos: crear el **`requirements.txt`**

La siguiente sección describe cómo especificar las dependencias de Python desde el Python Package Index en un archivo `requirements.txt`.

Apache Airflow v3

1. Hacer una prueba local. Añada bibliotecas adicionales de forma iterativa para encontrar la combinación adecuada de paquetes y sus versiones antes de crear un archivo `requirements.txt`. Para ejecutar la utilidad de la CLI de Amazon MWAA, consulte aws-mwaa-docker-images en GitHub.
2. Revise los extras del paquete Apache Airflow. Para acceder a una lista de los paquetes instalados para Apache Airflow v3 en Amazon MWAA, consulte aws-mwaa-docker-images `requirements.txt` en el sitio web de GitHub.
3. Añada instrucciones respecto a las restricciones. Añada el archivo de restricciones para su entorno Apache Airflow v3 en la parte superior del archivo `requirements.txt`. Los archivos de restricciones de Apache Airflow especifican las versiones de proveedores disponibles en el momento de la publicación de Apache Airflow.

   En el siguiente ejemplo, sustituya *`{environment-version}`* con el número de versión de su entorno y *`{Python-version}`* con la versión de Python compatible con su entorno.

   Para obtener información sobre la versión de Python compatible con su entorno de Apache Airflow, consulte las versiones de Apache Airflow.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   Si el archivo de restricciones determina que el paquete `xyz==1.0` no es compatible con otros paquetes de su entorno, `pip3 install` no podrá impedir que se instalen bibliotecas incompatibles en su entorno. Si se genera un error al instalar algún paquete, podrá ver los registros de errores de cada componente de Apache Airflow (el programador, el proceso de trabajo y el servidor web) en el flujo de registro correspondiente en los registros de CloudWatch. Para obtener más información sobre los tipos de registros, consulte the section called “Visualización de registros de Airflow”.
4. Paquetes de Apache Airflow. Añada los extras del paquete y la versión (==). Esto ayuda a evitar que se instalen en su entorno paquetes del mismo nombre, pero de una versión diferente.

   ```
   apache-airflow[package-extra]==2.5.1
   ```

5. Bibliotecas Python. Añada el nombre del paquete y la versión (==) al archivo `requirements.txt`. Esto permite evitar que se lleve a cabo una actualización futura de última hora de PyPi.org automáticamente.

```
library == version
```

Example Boto3 y psycopg2-binary

Este caso se proporciona como ejemplo. Las bibliotecas boto y psycopg2-binary vienen incluidas en la instalación base de Apache Airflow v3, por lo que no es necesario especificarlas en un archivo `requirements.txt`.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Si un paquete se especifica sin versión, Amazon MWAA instalará la última versión del paquete que encuentre en PyPi.org. Esta versión puede entrar en conflicto con otros paquetes de su `requirements.txt`.

Apache Airflow v2

1. Hacer una prueba local. Añada bibliotecas adicionales de forma iterativa para encontrar la combinación adecuada de paquetes y sus versiones antes de crear un archivo `requirements.txt`. Para ejecutar la utilidad de la CLI de Amazon MWAA, consulte aws-mwaa-docker-images en GitHub.
2. Revise los extras del paquete Apache Airflow. Para acceder a una lista de los paquetes instalados para Apache Airflow v2 en Amazon MWAA, consulte aws-mwaa-docker-images `requirements.txt` en el sitio web de GitHub.
3. Añada instrucciones respecto a las restricciones. Añada el archivo de restricciones para su entorno Apache Airflow v2 en la parte superior del archivo `requirements.txt`. Los archivos de restricciones de Apache Airflow especifican las versiones de proveedores disponibles en el momento de la publicación de Apache Airflow.

   A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA

especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que utilice.

En el siguiente ejemplo, sustituya *{environment-version}* con el número de versión de su entorno y *{Python-version}* con la versión de Python compatible con su entorno.

Para obtener información sobre la versión de Python compatible con su entorno de Apache Airflow, consulte las versiones de Apache Airflow.

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

Si el archivo de restricciones determina que el paquete `xyz==1.0` no es compatible con otros paquetes de su entorno, `pip3 install` no podrá impedir que se instalen bibliotecas incompatibles en su entorno. Si se genera un error al instalar algún paquete, podrá ver los registros de errores de cada componente de Apache Airflow (el programador, el proceso de trabajo y el servidor web) en el flujo de registro correspondiente en los registros de CloudWatch. Para obtener más información sobre los tipos de registros, consulte the section called “Visualización de registros de Airflow”.

4. Paquetes de Apache Airflow. Añada los extras del paquete y la versión (==). Esto ayuda a evitar que se instalen en su entorno paquetes del mismo nombre, pero de una versión diferente.

```
apache-airflow[package-extra]==2.5.1
```

5. Bibliotecas Python. Añada el nombre del paquete y la versión (==) al archivo `requirements.txt`. Esto permite evitar que se lleve a cabo una actualización futura de última hora de PyPi.org automáticamente.

```
library == version
```

Example Boto3 y psycopg2-binary

Este caso se proporciona como ejemplo. Las bibliotecas boto y psycopg2-binary vienen incluidas en la instalación base de Apache Airflow v2, por lo que no es necesario especificarlas en un archivo `requirements.txt`.

```
boto3==1.17.54
```

```
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Si un paquete se especifica sin versión, Amazon MWAA instalará la última versión del paquete que encuentre en PyPi.org. Esta versión puede entrar en conflicto con otros paquetes de su `requirements.txt`.

Cómo cargar **`requirements.txt`** a Amazon S3

Puede usar la consola de Amazon S3 o la AWS Command Line Interface (AWS CLI) para cargar un archivo `requirements.txt` a su bucket de Amazon S3.

Uso de AWS CLI

La AWS Command Line Interface (AWS CLI) es una herramienta de código abierto que le permite interactuar con los servicios de AWS mediante el uso de comandos en el shell de la línea de comandos. Para completar los pasos de esta página, necesita lo siguiente:

- AWS CLI: instalar la versión 2.
- AWS CLI: configuración rápida con `aws configure`.

Carga mediante la AWS CLI

1. Use el siguiente comando para obtener una lista de todos los buckets de Amazon S3.

   ```
   aws s3 ls
   ```

2. Utilice el comando siguiente para enumerar los archivos y las carpetas del bucket de Amazon S3 para su entorno.

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

3. El siguiente comando carga un archivo `requirements.txt` en el bucket de Amazon S3.

   ```
   aws s3 cp requirements.txt s3://amzn-s3-demo-bucket/requirements.txt
   ```

Uso de la consola de Amazon S3

La consola de Amazon S3 es una interfaz de usuario basada en la web que le permite crear y administrar los recursos de su bucket de Amazon S3.

Carga del contenido usando la consola de Amazon S3

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione el enlace del bucket S3 en el panel de códigos de DAG en S3 para abrir el bucket de almacenamiento en la consola.
4. Seleccione Cargar.
5. Elija Añadir archivo.
6. Seleccione la copia local de su `requirements.txt`, elija Cargar.

Instalación de dependencias de Python en su entorno

En esta sección, se describe cómo instalar las dependencias que ha cargado en su bucket de Amazon S3 especificando la ruta al archivo requirements.txt y especificando la versión del archivo requirements.txt cada vez que se actualiza.

Especificación de la ruta a **`requirements.txt`** en la consola MWAA de Amazon (la primera vez)

Si es la primera vez que crea y sube un archivo `requirements.txt` a su bucket de Amazon S3, también tendrá que especificar la ruta al archivo en la consola de Amazon MWAA. Solo necesita realizar este paso una vez.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Editar.
4. En el panel DAG code in Amazon S3 (Código DAG en Amazon S3), elija Browse S3 (Explorar S3) junto al campo Requirements file - optional (Archivo de requisitos: opcional).
5. Seleccione el archivo `requirements.txt` en su bucket de Amazon S3.
6. Seleccione Elegir.

7. Seleccione Siguiente, Actualizar entorno.

Puede empezar a usar los nuevos paquetes inmediatamente después de que su entorno termine de actualizarse.

Especificación de la versión de `requirements.txt` en la consola de Amazon MWAA

Debe especificar la versión de su archivo `requirements.txt` en la consola de Amazon MWAA cada vez que cargue una nueva versión de su `requirements.txt` en su bucket de Amazon S3.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Editar.
4. En el panel Código de DAG en Amazon S3, elija una versión de `requirements.txt` de la lista desplegable.
5. Seleccione Siguiente, Actualizar entorno.

Puede empezar a usar los nuevos paquetes inmediatamente después de que su entorno termine de actualizarse.

Visualización de registros para su `requirements.txt`

Consulte los registros de Apache Airflow correspondientes al programador encargado de programar sus flujos de trabajo y de analizar su carpeta de `dags`. En los siguientes pasos, se describe cómo abrir el grupo de registros del programador en la consola de Amazon MWAA y cómo ver los registros de Apache Airflow en la consola de registros de CloudWatch.

Para acceder a los registros de un **`requirements.txt`**

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija el Grupo de registro del programador de Airflow en el panel de Monitorización.
4. Seleccione el registro `requirements_install_ip` en los flujos de registro.
5. Consulte la lista de paquetes que se hayan instalado en el entorno en `/usr/local/airflow/.local/bin`. Por ejemplo:

```
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. Consulte la lista de paquetes y compruebe si se produjo algún error en alguno de ellos durante la instalación. Si algo salió mal, es posible que aparezca un error similar al siguiente:

```
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))
```

Siguientes pasos

Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

Eliminación de archivos en Amazon S3

En esta página, se describe cómo funciona el control de versiones en un bucket de Amazon S3 para un entorno de Amazon Managed Workflows para Apache Airflow y los pasos que se deben seguir para eliminar un archivo DAG, `plugins.zip` o `requirements.txt`.

Contenido

- Siguientes pasos

Requisitos previos

Para poder llevar a cabo los pasos de esta página, necesitará lo siguiente.

- Permisos: el administrador debe haber concedido a su Cuenta de AWS acceso a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno. Además, su rol de ejecución debe permitir que su entorno de Amazon MWAA acceda a los recursos de AWS que utiliza su entorno.
- Acceso: si tiene que acceder a los repositorios públicos para instalar dependencias directamente en el servidor web, su entorno debe estar configurado con acceso a un servidor web de red pública. Para obtener más información, consulta the section called “Modos de acceso de Apache Airflow”.
- Configuración de Amazon S3: el bucket de Amazon S3 que se utiliza para almacenar los DAG, los complementos personalizados en `plugins.zip` y las dependencias de Python en `requirements.txt` deben estar configurados con el acceso público bloqueado y el control de versiones activado.

Descripción general del control de versiones

Los archivos `requirements.txt` y `plugins.zip` de su bucket de Amazon S3 están versionados. Cuando se habilita el control de versiones de un bucket de Amazon S3 para un objeto y se elimina un artefacto (por ejemplo, plugins.zip) de un bucket de Amazon S3, el archivo no se elimina por completo. Cada vez que se elimina un artefacto en Amazon S3, se crea una nueva copia del archivo, que es un error 404 (Object not found)/archivo 0 k que dice `I'm not here`. Amazon S3 lo denomina marcador de eliminación. Un marcador de eliminación es una versión “nula” del archivo con un nombre de clave (o clave) y un ID de versión al igual que cualquier otro objeto.

Recomendamos borrar las versiones de los archivos y los marcadores de eliminación periódicamente para reducir los costos de almacenamiento de su bucket de Amazon S3. Para eliminar por completo las versiones de los archivos “no actuales” (anteriores), debe eliminar las versiones de los archivos y, a continuación, el marcador de eliminación de la versión.

Funcionamiento

Amazon MWAA ejecuta una operación de sincronización en su bucket de Amazon S3 cada treinta segundos. Esto hace que cualquier eliminación de DAG en un bucket de Amazon S3 se sincronice con la imagen de Airflow del contenedor de Fargate.

En el caso de los archivos `plugins.zip` y `requirements.txt`, los cambios solo tienen lugar después de una actualización del entorno, cuando Amazon MWAA crea una nueva imagen de Airflow del contenedor de Fargate con los complementos personalizados y dependencias de Python. Si elimina la versión actual de un archivo `requirements.txt` o `plugins.zip` y, a continuación, actualiza el entorno sin proporcionar una nueva versión para el archivo eliminado, se producirá un error en la actualización y aparecerá un mensaje de error, como `Unable to read version {version number} of file {file name}`.

Eliminación de un DAG en Amazon S3

Los archivos DAG (`.py`) no están versionados y se pueden eliminar directamente en la consola de Amazon S3. En los siguientes pasos, se describe cómo eliminar un DAG del bucket de Amazon S3.

Eliminación de un DAG

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Seleccione el enlace del bucket de S3 en el panel de códigos de DAG en S3 para abrir el bucket de almacenamiento en la consola.
4. Elija la carpeta `dags`.
5. Seleccione el DAG y, a continuación, Eliminar.
6. En ¿Eliminar objetos?, escriba `delete`.
7. Elija Eliminar objetos.

> **Note**
>
> Apache Airflow conserva el historial de ejecuciones de DAG. Después de ejecutar un DAG en Apache Airflow, este permanece en la lista de DAG de Airflow independientemente del estado del archivo, hasta que usted lo elimine de Apache Airflow. Para eliminar un DAG

en Apache Airflow, pulse el botón rojo “Delete” (Eliminar) situado en la columna de Links (Enlaces).

Eliminación de un archivo requirements.txt o plugins.zip “actual” de un entorno

Actualmente, no se puede eliminar un archivo plugins.zip o requirements.txt de un entorno después de haberlo añadido, pero estamos trabajando para solucionar el problema. Mientras tanto, una solución alternativa es apuntar a un archivo de texto o zip vacío, respectivamente.

Eliminación de una versión “no actual” (anterior) de archivos requirements.txt o plugins.zip

Los archivos `requirements.txt` y `plugins.zip` de su bucket de Amazon S3 están versionados en Amazon MWAA. Si desea eliminar por completo estos archivos de su bucket de Amazon S3, debe recuperar la versión actual (121212) del objeto (por ejemplo, plugins.zip), eliminarla y, a continuación, borrar el marcador de eliminación de las versiones del archivo.

También puede eliminar versiones de archivos “no actuales” (anteriores) en la consola de Amazon S3; sin embargo, tendrá que borrar el marcador de eliminación mediante una de las siguientes opciones.

- Para recuperar la versión del objeto, consulte Recuperar versiones de objetos de un bucket habilitado para el control de versiones en la guía de Amazon S3.
- Para eliminar la versión del objeto, consulte Eliminar versiones de objetos de un bucket con control de versiones habilitado en la guía de Amazon S3.
- Para borrar un marcador de eliminación, consulte Gestión de marcadores de eliminación en la guía de Amazon S3.

Uso de ciclos de vida para eliminar versiones “no actuales” (anteriores) y eliminar marcadores automáticamente

Puede configurar una política de ciclo de vida para su bucket de Amazon S3 para eliminar las versiones “no actuales” (anteriores) de los archivos plugins.zip y requirements.txt de su bucket de

Amazon S3 tras un número determinado de días, o para eliminar el marcador de eliminación de un objeto vencido.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. En el código DAG de Amazon S3, elija su bucket de Amazon S3.
4. Elija Crear regla de ciclo de vida.

Ejemplo de política de ciclo de vida para eliminar versiones “no actuales” de requirements.txt y eliminar marcadores automáticamente

En el siguiente ejemplo, se indica cómo crear una regla de ciclo de vida que elimine permanentemente las versiones “no actuales” de un archivo requirements.txt y sus marcadores de eliminación después de treinta días.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. En el código DAG de Amazon S3, elija su bucket de Amazon S3.
4. Elija Crear regla de ciclo de vida.
5. En Nombre de la regla de ciclo de vida, escriba `Delete previous requirements.txt versions and delete markers after thirty days`.
6. En Prefijo, requisitos.
7. En Acciones de la regla del ciclo de vida, seleccione Eliminar definitivamente las versiones anteriores de los objetos y Eliminar los marcadores de eliminación vencidos o las cargas multiparte incompletas.
8. En Número de días después de los cuales los objetos pasan a ser versiones anteriores, escriba `30`.
9. En Marcadores de eliminación de objetos vencidos, elija Eliminar marcadores de eliminación de objetos vencidos; los objetos se eliminarán permanentemente después de 30 días.

Siguientes pasos

- Obtenga más información sobre los marcadores de eliminación de Amazon S3 en Gestión de marcadores de eliminación.

- Obtenga más información sobre los ciclos de vida de Amazon S3 en Vencimiento de objetos.

Red

En esta guía se describe la configuración de red de Amazon VPC que necesitará para un entorno de Amazon MWAA.

Secciones

Acerca de las redes en Amazon MWAA

Una Amazon VPC es una red virtual que está vinculada a su Cuenta de AWS. Le ofrece seguridad en la nube y la capacidad de escalar de forma dinámica al proporcionar un control detallado de la infraestructura virtual y la segmentación del tráfico de la red. En esta página, se se describe la infraestructura de Amazon VPC con enrutamiento público o enrutamiento privado necesaria para que sea compatible con un entorno de Amazon Managed Workflows para Apache Airflow.

Contenido

Términos

Enrutamiento público

Red de Amazon VPC con acceso a Internet.

Enrutamiento privado

Red de Amazon VPC sin acceso a Internet.

Elementos compatibles

En la siguiente tabla se describen los tipos de Amazon VPC compatibles con Amazon MWAA.

Tipos de Amazon VPC	Compatible	
Amazon VPC propiedad de la cuenta que intenta crear el entorno.	Sí	
Amazon VPC compartida en la que varias Cuentas de AWS crean sus recursos de AWS.	Sí	

Información general sobre la infraestructura de la VPC

Al crear un entorno de Amazon MWAA, Amazon MWAA crea entre uno y dos puntos de conexión de VPC para su entorno en función del modo de acceso a Apache Airflow que haya elegido para su entorno. Estos puntos de conexión aparecen como interfaces de red elástica (ENI) con IP privadas en su Amazon VPC. Una vez creados estos puntos de conexión, todo el tráfico destinado a estas IP se enruta de forma privada o pública a los servicios de AWS correspondientes que utilice su entorno.

En la siguiente sección, se describe la infraestructura de Amazon VPC necesaria para enrutar el tráfico de forma pública a través de Internet o de forma privada en su Amazon VPC.

Enrutamiento público a través de Internet

En esta sección se describe la infraestructura de Amazon VPC de un entorno con enrutamiento público. Para ello, necesitará la siguiente infraestructura de la VPC:

- Grupo de seguridad de la VPC. Un grupo de seguridad de VPC funciona como un firewall virtual de para controlar el tráfico entrante y saliente en una instancia.
  - Se pueden especificar hasta 5 grupos de seguridad.
  - El grupo de seguridad debe especificar una regla de entrada con autorreferencia para sí mismo.
  - El grupo de seguridad debe especificar una regla de salida para todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).
  - El grupo de seguridad debe permitir todo el tráfico de la regla de autorreferencia. Por ejemplo, (Recomendado) Ejemplo de grupo de seguridad con autorreferencia para todos los accesos .
  - Opcionalmente, el grupo de seguridad puede restringir todavía más el tráfico especificando el rango de puertos para el rango de puertos HTTPS `443` y un rango de puertos TCP `5432`. Por ejemplo, (Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 5432 y (Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 443.
- Dos subredes públicas. Una subred pública es una subred asociada a la tabla de ruteo con ruta a la gateway de internet.
  - Se necesitan dos subredes públicas. Esto permite a Amazon MWAA crear una nueva imagen de contenedor para su entorno en la otra zona de disponibilidad, en caso de que un contenedor falle.
  - Las subredes deben estar en diferentes zonas de disponibilidad. Por ejemplo, `us-east-1a`, `us-east-1b`.
  - Las subredes deben enrutarse a una puerta de enlace NAT (o instancia NAT) con una dirección IP elástica (EIP).
  - Las subredes deben tener una tabla de enrutamiento que dirija el tráfico vinculado a Internet a la puerta de enlace de Internet.
- Dos subredes privadas. Una subred pública es una subred no asociada a la tabla de enrutamiento que tiene una ruta a la puerta de enlace de Internet.
  - Se necesitan dos subredes privadas. Esto permite a Amazon MWAA crear una nueva imagen de contenedor para su entorno en la otra zona de disponibilidad, en caso de que un contenedor falle.
  - Las subredes deben estar en diferentes zonas de disponibilidad. Por ejemplo, `us-east-1a`, `us-east-1b`.
  - Las subredes deben tener una tabla de enrutamiento a un dispositivo NAT (puerta de enlace o instancia).

  - Las subredes no deben tener una ruta hacia una puerta de enlace de Internet.
  - Establezca `assignIpV6AddressOnCreation` como `true` para subredes IPv6.
  - Para las subredes privadas IPv6, debe tener una conexión a la puerta de enlace de Internet de solo salida (EIGW).
- Una Lista de control de acceso (ACL) de red). Una NACL gestiona (mediante reglas de permiso o denegación) el tráfico entrante y saliente a nivel de subred.
  - La NACL debe tener una regla de entrada que permita todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).
  - La NACL debe tener una regla de salida que permita todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).
  - Por ejemplo, (Recomendado) Ejemplos de ACL.
- Dos puertas de enlace NAT (o instancias NAT). El dispositivo NAT reenvía el tráfico desde las instancias de la subred privada a Internet o a otros servicios de AWS y, a continuación, enruta la respuesta de nuevo a las instancias.
  - El dispositivo NAT debe estar conectado a una subred pública. (Un dispositivo NAT para cada subred pública.)
  - El dispositivo NAT debe tener una dirección IPv4 elástica (EIP) conectada a cada subred pública.
- Un gateway de Internet. Una puerta de enlace de Internet conecta una Amazon VPC a Internet y otros servicios de AWS.
  - Una puerta de enlace de Internet debe asociarse a la Amazon VPC.

Enrutamiento privado sin acceso a Internet

En esta sección se describe la infraestructura de Amazon VPC de un entorno con enrutamiento privado. Para ello, necesitará la siguiente infraestructura de la VPC:

- Grupo de seguridad de la VPC. Un grupo de seguridad de VPC funciona como un firewall virtual de para controlar el tráfico entrante y saliente en una instancia.
  - Se pueden especificar hasta 5 grupos de seguridad.
  - El grupo de seguridad debe especificar una regla de entrada con autorreferencia para sí mismo.
  - El grupo de seguridad debe especificar una regla de salida para todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).

  - El grupo de seguridad debe permitir todo el tráfico de la regla de autorreferencia. Por ejemplo, (Recomendado) Ejemplo de grupo de seguridad con autorreferencia para todos los accesos .
  - Opcionalmente, el grupo de seguridad puede restringir todavía más el tráfico especificando el rango de puertos para el rango de puertos HTTPS `443` y un rango de puertos TCP `5432`. Por ejemplo, (Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 5432 y (Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 443.
- Dos subredes privadas. Una subred pública es una subred no asociada a la tabla de enrutamiento que tiene una ruta a la puerta de enlace de Internet.
  - Se necesitan dos subredes privadas. Esto permite a Amazon MWAA crear una nueva imagen de contenedor para su entorno en la otra zona de disponibilidad, en caso de que un contenedor falle.
  - Las subredes deben estar en diferentes zonas de disponibilidad. Por ejemplo, `us-east-1a`, `us-east-1b`.
  - Las subredes deben tener una tabla de enrutamiento a los puntos de conexión de la VPC.
  - Las subredes deben tener una tabla de enrutamiento a la EIGW para poder descargarla de Internet como parte de un DAG.
  - Las subredes no deben tener una tabla de enrutamiento a un dispositivo NAT (puerta de enlace o instancia) ni una puerta de enlace de Internet.
- Una Lista de control de acceso (ACL) de red). Una NACL gestiona (mediante reglas de permiso o denegación) el tráfico entrante y saliente a nivel de subred.
  - La NACL debe tener una regla de entrada que permita todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).
  - La NACL debe tener una regla de salida que deniegue todo el tráfico (`0.0.0.0/0`; use `::/0` para IPv6).
  - Por ejemplo, (Recomendado) Ejemplos de ACL.
- Tabla de enrutamiento local. Una tabla de enrutamiento local es una ruta predeterminada para la comunicación dentro de la VPC.
  - La tabla de enrutamiento local debe estar asociada a sus subredes privadas.
  - La tabla de enrutamiento local debe permitir que las instancias de la VPC se comuniquen con su propia red. Por ejemplo, si usa un AWS Client VPN para acceder al punto de conexión de la interfaz de VPC de su servidor web de Apache Airflow, la tabla de enrutamiento debe enrutarse al punto de conexión de VPC.

- Puntos de conexión de VPC para cada servicio de AWS usado por su entorno y puntos de conexión de VPC de Apache Airflow en la misma Región de AWS y Amazon VPC que su entorno de Amazon MWAA.
  - Un punto final de VPC para cada servicio de AWS utilizado por el entorno y puntos de conexión de VPC para Apache Airflow. Por ejemplo, (Obligatorio) Puntos de conexión de VPC.
  - Los puntos conexión de VPC deben tener habilitado el DNS privado.
  - Los puntos de conexión de VPC deben estar asociados a las dos subredes privadas de su entorno.
  - Los puntos de conexión de VPC deben estar asociados al grupo de seguridad de su entorno.
  - Debe configurarse la política de puntos de conexión de VPC para cada punto de conexión para que permita el acceso a los servicios de AWS usados por el entorno. Por ejemplo, (Recomendado) Ejemplo de política de punto de conexión de VPC que permite todos los accesos.
  - Para permitir el acceso de los bucket, debe configurarse una política de puntos de conexión de VPC para Amazon S3. Por ejemplo, (Recomendado) Ejemplo de política de puntos de conexión de la puerta de enlace de Amazon S3 que permite el acceso al bucket.

Ejemplos de casos de uso para un modo de acceso a Amazon VPC y Apache Airflow

En esta sección, se describen los diferentes casos de uso para el acceso a la red en su Amazon VPC y el modo de acceso al servidor web de Apache Airflow que debe elegir en la consola de Amazon MWAA.

Se permite el acceso a Internet: nueva red de Amazon VPC

Si su organización permite el acceso a Internet en su VPC y desea que los usuarios accedan a su servidor web de Apache Airflow a través de Internet:

1. Cree una red de Amazon VPC con acceso a Internet.
2. Cree un entorno con el modo de acceso de red pública para su servidor web de Apache Airflow.
3. Nuestra recomendación: le recomendamos que utilice la plantilla de inicio rápido de CloudFormation que crea la infraestructura de Amazon VPC, un bucket de Amazon S3 y un entorno de Amazon MWAA al mismo tiempo. Consulte Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow para obtener más información.

Si su organización permite el acceso a Internet en su VPC y desea limitar el acceso de los usuarios a su servidor web de Apache Airflow dentro de su VPC:

1. Cree una red de Amazon VPC con acceso a Internet.
2. Cree un mecanismo para acceder al punto de conexión de la interfaz de la VPC de su servidor web de Apache Airflow desde su computadora.
3. Cree un entorno con el modo de acceso de red privada para su servidor web de Apache Airflow.
4. Recomendaciones:

    a. Le recomendamos utilizar la consola de Amazon MWAA en Opción 1: crear la red de VPC en la consola de Amazon MWAA o la plantilla de CloudFormation en Opción 2: Creación de una red Amazon VPC con acceso a Internet.

    b. Recomendamos configurar el acceso mediante un AWS Client VPN a su servidor web de Apache Airflow en Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN.

No se permite el acceso a Internet: nueva red de Amazon VPC

Si su organización no permite el acceso a Internet en su VPC:

1. Cree una red de Amazon VPC sin acceso a Internet.
2. Cree un mecanismo para acceder al punto de conexión de la interfaz de la VPC de su servidor web de Apache Airflow desde su computadora.
3. Cree puntos de conexión de VPC para cada servicio de AWS que utilice su entorno.
4. Cree un entorno con el modo de acceso de red privada para su servidor web de Apache Airflow.
5. Recomendaciones:

    a. Recomendamos usar la plantilla de CloudFormation para crear una Amazon VPC sin acceso a Internet y los puntos de conexión de la VPC para cada servicio de AWS usado por Amazon MWAA en Opción 3: Creación de una red de Amazon VPC sin acceso a Internet.

    b. Recomendamos configurar el acceso mediante un AWS Client VPN a su servidor web de Apache Airflow en Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN.

No se permite el acceso a Internet: red de Amazon VPC existente

Si su organización no permite el acceso a Internet en su VPC y usted ya dispone de la red de Amazon VPC necesaria sin acceso a Internet:

1. Cree puntos de conexión de VPC para cada servicio de AWS que utilice su entorno.
2. Cree puntos de conexión de VPC para Apache Airflow.
3. Cree un mecanismo para acceder al punto de conexión de la interfaz de la VPC de su servidor web de Apache Airflow desde su computadora.
4. Cree un entorno con el modo de acceso de red privada para su servidor web de Apache Airflow.
5. Recomendaciones:
    a. Recomendamos crear y adjuntar los puntos de conexión de VPC necesarios para cada servicio de AWS utilizado por Amazon MWAA y los puntos de conexión de VPC necesarios para Apache Airflow en Creación de los puntos de conexión del servicio de VPC necesarios en una Amazon VPC con enrutamiento privado.
    b. Recomendamos configurar el acceso mediante un AWS Client VPN a su servidor web de Apache Airflow en Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN.

Seguridad en la VPC en Amazon MWAA

En esta página se describen los componentes de Amazon VPC que se utilizan para proteger su entorno de Amazon Managed Workflows para Apache Airflow y las configuraciones necesarias para estos componentes.

Contenido

  - (Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 443
- Políticas de puntos de conexión de VPC (solo enrutamiento privado)
  - (Recomendado) Ejemplo de política de punto de conexión de VPC que permite todos los accesos
  - (Recomendado) Ejemplo de política de puntos de conexión de la puerta de enlace de Amazon S3 que permite el acceso al bucket

Términos

Enrutamiento público

Red de Amazon VPC con acceso a Internet.

Enrutamiento privado

Una red de Amazon VPC sin acceso a Internet.

Información general acerca de la seguridad

Los grupos de seguridad y las listas de control de acceso (ACL) ofrecen maneras de controlar el tráfico de red en las subredes e instancias de su Amazon VPC mediante las reglas que usted especifique.

- El tráfico de red entrante y saliente de una subred se puede controlar mediante listas de control de acceso (ACL). Solo necesita una ACL y la misma ACL se puede usar en varios entornos.
- Un grupo de seguridad de Amazon VPC puede controlar el tráfico de red entrante y saliente de una instancia. Puede usar entre uno y cinco grupos de seguridad por entorno.
- El tráfico de red entrante y saliente de una instancia también se puede controlar mediante políticas de punto de conexión de VPC. Si su organización no permite que se acceda a Internet dentro de Amazon VPC y usa una red de Amazon VPC con enrutamiento privado, se necesita una política de puntos de conexión de VPC para los puntos de conexión de VPC de AWS y los puntos de conexión de VPC de Apache Airflow.

Listas de control de acceso (ACL) de red

Una lista de control de acceso (ACL) de red puede administrar (mediante reglas de permiso o de rechazo) el tráfico entrante y saliente a nivel de subred. Una ACL no tiene estado, lo que significa

que las reglas de entrada y salida se deben especificar de manera independiente y explícita. Se usa para especificar los tipos de tráfico de red que está permitido que entren o salgan de las instancias de una red de VPC.

Todas las Amazon VPC incluyen una ACL predeterminada que permite todo el tráfico de entrada y salida. Puede editar las reglas de ACL predeterminadas o crear una ACL personalizada y adjuntarla a sus subredes. Una subred solo puede tener una ACL conectada a ella en cualquier momento, pero una ACL se puede conectar a varias subredes.

(Recomendado) Ejemplos de ACL

En el siguiente ejemplo, se muestran las reglas de ACL de entrada y de salida que se pueden usar para una Amazon VPC con enrutamiento público o privado.

Número de regla	Tipo	Protocolo	Rango de puerto	Origen	Permitir/ Denegar
100	Todo el tráfico IPv4	Todos	Todos	0.0.0.0/0	Permitir
*	Todo el tráfico IPv4	Todos	Todos	0.0.0.0/0	Denegar

Grupos de seguridad de la VPC

Un grupo de seguridad VPC funciona como un firewall virtual que controla el tráfico a nivel de instancia. Los grupos de seguridad tienen la característica “con estado”, lo que significa que cuando se permite una conexión entrante, puede responder. Se usan para especificar los tipos de tráfico de red permitidos desde las instancias de una red de VPC.

Todas las Amazon VPC incluyen un grupo de seguridad predeterminado. De forma predeterminada, este carece de reglas de entrada. Tiene una regla de salida que permite todo el tráfico saliente. Puede editar las reglas predeterminadas del grupo de seguridad o crear un grupo de seguridad personalizado y adjuntarlo a su Amazon VPC. En Amazon MWAA, debe configurar las reglas de entrada y salida para dirigir el tráfico en sus puertas de enlace NAT.

(Recomendado) Ejemplo de grupo de seguridad con autorreferencia para todos los accesos

En el siguiente ejemplo, se muestran las reglas de entrada del grupo de seguridad que permiten todo el tráfico a una Amazon VPC con enrutamiento público o privado. El grupo de seguridad de este ejemplo es una regla con autorreferencia a sí mismo.

Tipo	Protocolo	Tipo de origen	Origen		
Todo el tráfico	Todos	Todos	sg-0909e8e81919 / my-mwaa-vpc-security-group		

En el siguiente ejemplo, se muestran las reglas de salida de los grupos de seguridad.

Tipo	Protocolo	Tipo de origen	Origen		
Todo el tráfico	Todos	Todos	0.0.0.0/0		

(Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 5432

En el siguiente ejemplo, se muestran las reglas de entrada de los grupos de seguridad que permiten todo el tráfico HTTPS en el puerto 5432 para la base de datos de metadatos PostgreSQL de Amazon Aurora (propiedad de Amazon MWAA) de su entorno.

> **Note**
>
> Si decide restringir el tráfico mediante esta regla, tendrá que añadir otra que permita el tráfico TCP en el puerto 443.

Tipo	Protocolo	Rango de puerto	Tipo de origen	Origen	
TCP personalizada	TCP	5432	Personalizado	sg-0909e8e81919 / my-mwaa-vpc-security-group	

(Opcional) Ejemplo de grupo de seguridad que restringe el acceso entrante al puerto 443

En el siguiente ejemplo, se muestran las reglas de entrada del grupo de seguridad que permiten todo el tráfico TCP del puerto 443 del servidor web de Apache Airflow.

Tipo	Protocolo	Rango de puerto	Tipo de origen	Origen	
HTTPS	TCP	443	Personalizado	sg-0909e8e81919 / my-mwaa-vpc-security-group	

Políticas de puntos de conexión de VPC (solo enrutamiento privado)

Una política de punto de conexión de VPC (AWS PrivateLink) controla el acceso a los servicios de AWS desde su subred privada. Una política de punto de conexión de VPC es una política de

recursos de IAM que se adjunta a un punto de conexión de VPC. En esta sección, se describen los permisos necesarios para las políticas de puntos de conexión de VPC para cada punto de conexión de VPC.

Le recomendamos que utilice una política de VPC de tipo interfaz para cada uno de los puntos de conexión de VPC que haya creado que permita el acceso total a todos los servicios de AWS, y que utilice su rol de ejecución exclusivamente para los permisos de AWS.

(Recomendado) Ejemplo de política de punto de conexión de VPC que permite todos los accesos

En el siguiente ejemplo, se muestra una política de punto de conexión de interfaz de VPC para una Amazon VPC con enrutamiento privado.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

(Recomendado) Ejemplo de política de puntos de conexión de la puerta de enlace de Amazon S3 que permite el acceso al bucket

En el siguiente ejemplo, se muestra una política de puntos de conexión de la puerta de enlace de VPC que proporciona acceso a los bucket de Amazon S3 necesarios para las operaciones de Amazon ECR de una Amazon VPC con enrutamiento privado. Esto es necesario para poder recuperar la imagen de Amazon ECR, además del bucket en el que se almacenan los DAG y los archivos auxiliares.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
```

```
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-us-east-1-starport-layer-bucket/*"]
    }
  ]
}
```

Administración del acceso a los puntos de conexión de Amazon VPC específicos del servicio en Amazon MWAA

Un punto de conexión de VPC (AWS PrivateLink) permite conectar de forma privada una VPC a los servicios alojados en AWS sin necesidad de una puerta de enlace de Internet, un dispositivo NAT, una VPN ni proxies de firewall. Estos puntos de conexión son dispositivos con escalabilidad horizontal y alta disponibilidad que permiten la comunicación entre las instancias de su VPC y los servicios de AWS. En esta página, se describen los puntos de conexión de VPC creados por Amazon MWAA y cómo acceder al punto de conexión de VPC del servidor web Apache Airflow si ha elegido el modo de acceso de Red privada en Amazon Managed Workflows para Apache Airflow.

Contenido

Precios

- AWS PrivateLink Precios de

Descripción de puntos de conexión de VPC

Cuando crea un entorno Amazon MWAA, Amazon MWAA crea entre uno y dos puntos de conexión de VPC para dicho entorno. Estos puntos de conexión aparecen como interfaces de red elástica (ENI) con IP privadas en Amazon VPC. Una vez creados estos puntos de conexión, todo el tráfico destinado a estas IP se enruta de forma privada o pública a los servicios de AWS correspondientes que utilice su entorno.

Modo de acceso mediante red pública

Si elige el modo de acceso de Red pública para su servidor web Apache Airflow, el tráfico de la red se enruta públicamente a través de Internet.

- Amazon MWAA crea un punto de conexión de la interfaz de la VPC para su base de datos de metadatos de Amazon Aurora PostgreSQL. El punto de conexión se crea en las zonas de disponibilidad correspondientes a sus subredes privadas y es independiente de las demás Cuentas de AWS.
- A continuación, Amazon MWAA vincula una dirección IP de sus subredes privadas a los puntos de conexión de la interfaz. Se ha diseñado de esta manera siguiendo la práctica recomendada de vincular una sola IP de cada zona de disponibilidad de la VPC de Amazon.

Modo de acceso mediante red privada

Si elige el modo de acceso de Red privada para su servidor web Apache Airflow, el tráfico de red se enruta de forma privada dentro de Amazon VPC.

- Amazon MWAA crea un punto de conexión de la interfaz de la VPC para el servidor web Apache Airflow y un punto de conexión de la interfaz para la base de datos de metadatos de Amazon Aurora PostgreSQL. Los puntos de conexión se crean en las zonas de disponibilidad asignadas a sus subredes privadas y son independientes de otras Cuentas de AWS.
- A continuación, Amazon MWAA vincula una dirección IP de sus subredes privadas a los puntos de conexión de la interfaz. Se ha diseñado de esta manera siguiendo la práctica recomendada de vincular una sola IP de cada zona de disponibilidad de la VPC de Amazon.

Permiso para usar otros servicios de AWS

Los puntos de conexión de la interfaz utilizan el rol de ejecución de su entorno en AWS Identity and Access Management (IAM) para administrar los permisos de los recursos de AWS utilizados por su entorno. Según se vayan habilitando más servicios de AWS para un entorno, será necesario configurar permisos para cada uno de ellos usando el rol de ejecución del entorno. Para agregar permisos, consulte Rol de ejecución de Amazon MWAA.

Si elige el modo de acceso de Red privada para su servidor web Apache Airflow, también debe habilitar el permiso en la política de punto de conexión de VPC para cada punto de conexión. Consulte the section called “Políticas de puntos de conexión de VPC (solo enrutamiento privado)” para obtener más información.

Acceso a los puntos de conexión de VPC

En esta sección, se describe cómo acceder a los puntos de conexión de VPC creados por Amazon MWAA y cómo identificar las direcciones IP privadas de su punto de conexión de VPC Apache Airflow.

Acceso a puntos de conexión de VPC en la consola de Amazon VPC

En la siguiente sección, se muestran los pasos para acceder a los puntos de conexión de VPC creados por Amazon MWAA y cualquier punto de conexión de VPC que haya creado si usa un enrutamiento privado para Amazon VPC.

Para acceder a los puntos de conexión de VPC

1. Abra la página Puntos de conexión de la consola de Amazon VPC.
2. Seleccione su Región de AWS.
3. Consulte los puntos de conexión de la interfaz de VPC creados por Amazon MWAA y cualquier punto de conexión de VPC que haya creado si usa un enrutamiento privado en Amazon VPC.

Para obtener más información sobre los puntos de conexión del servicio de VPC que se requieren para una Amazon VPC con enrutamiento privado, consulte Creación de los puntos de conexión del servicio de VPC necesarios en una Amazon VPC con enrutamiento privado.

Identificación de direcciones IP privadas del servidor web Apache Airflow y el punto de conexión de VPC

En los siguientes pasos, se describe cómo recuperar el nombre de host del servidor web Apache Airflow y el punto de conexión de la interfaz de VPC, así como las direcciones IP privadas.

1. Use el siguiente comando de la AWS Command Line Interface (AWS CLI) para recuperar el nombre de host del servidor web Apache Airflow.

   ```
   aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query
    'Environment.WebserverUrl'
   ```

   Debería ver algo similar a la siguiente respuesta:

   ```
   "99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
   ```

2. Ejecute un comando dig en el nombre de host devuelto en la respuesta al comando anterior. Por ejemplo:

   ```
   dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-
   west-2.airflow.amazonaws.com
   ```

   Debería ver algo similar a la siguiente respuesta:

   ```
   vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
   west-2.vpce.amazonaws.com.
   ```

3. Use el siguiente comando de la AWS Command Line Interface (AWS CLI) para recuperar el nombre DNS del punto de conexión de VPC devuelto en la respuesta al comando anterior. Por ejemplo:

   ```
   aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-
   svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
   ```

   Debería ver algo similar a la siguiente respuesta:

   ```
   "DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
   west-2.vpce.amazonaws.com",
   ```

4. Ejecute un comando nslookup o dig en el nombre de host de Apache Airflow y en el nombre DNS del punto de conexión de VPC para recuperar las direcciones IP. Por ejemplo:

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

Debería ver algo similar a la siguiente respuesta:

```
192.0.5.1
192.0.6.1
```

Acceso al punto de conexión de VPC del servidor web Apache Airflow (acceso mediante red privada)

Si elige el modo de acceso de red privada para su servidor web Apache Airflow, tendrá que crear un mecanismo para acceder al punto de conexión de la interfaz de VPC de su servidor web Apache Airflow. Debe usar la misma Amazon VPC, el mismo grupo de seguridad de VPC y las mismas subredes privadas que su entorno de Amazon MWAA para estos recursos.

Uso de AWS Client VPN

AWS Client VPN es un servicio de VPN basado en cliente administrado que permite obtener acceso de forma segura a sus recursos de AWS y otros recursos en la red local. Proporciona una conexión TLS segura desde cualquier ubicación mediante el cliente OpenVPN.

Recomendamos seguir el tutorial de Amazon MWAA para configurar una Client VPN: Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN.

Uso de un host bastión de Linux

Un host bastión es un servidor cuya finalidad es proporcionar acceso a una red privada desde una red externa, por ejemplo, a través de Internet desde su equipo. Las instancias de Linux se encuentran en una subred pública y están configuradas con un grupo de seguridad que permite el acceso SSH desde el grupo de seguridad adjunto a la instancia Amazon EC2 subyacente que ejecuta el host bastión.

Recomendamos seguir el tutorial de Amazon MWAA para configurar un host bastión de Linux: Tutorial: Configuración del acceso a la red privada mediante un host bastión de Linux.

Uso de un equilibrador de carga (avanzado)

En la siguiente sección, se muestran las configuraciones que necesitará aplicar a un equilibrador de carga de aplicación.

1. Grupos de destino. Deberá usar grupos de destino que apunten a las direcciones IP privadas de su servidor web Apache Airflow y a su punto de conexión de interfaz de VPC. Le recomendamos que especifique ambas direcciones IP privadas como destinos registrados, ya que usar solo una puede reducir la disponibilidad. Para obtener más información sobre cómo identificar las direcciones IP privadas, consulte the section called “Identificación de direcciones IP privadas del servidor web Apache Airflow y el punto de conexión de VPC”.
2. Códigos de estado. Le recomendamos que utilice los códigos de estado `200` y `302` en la configuración del grupo de destino. De lo contrario, es posible que los destinos se marquen como en mal estado si el punto de conexión de VPC del servidor web Apache Airflow responde con un error `302 Redirect`.
3. Oyente HTTPS. Deberá especificar el puerto de destino del servidor web Apache Airflow. Por ejemplo:

Protocolo	Puerto
HTTPS	443

4. Nuevo dominio de ACM. Si quiere asociar un certificado SSL/TLS en AWS Certificate Manager, debe crear un nuevo dominio para el oyente HTTPS del equilibrador de carga.
5. Región de certificado ACM. Si desea asociar un certificado SSL/TLS en AWS Certificate Manager, debe cargarlo en la misma región de Región de AWS que su entorno. Por ejemplo:

    - Example región para cargar el certificado

    ```
    aws acm import-certificate --certificate fileb://Certificate.pem --certificate-
    chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --
    region us-west-2
    ```

Creación de los puntos de conexión del servicio de VPC necesarios en una Amazon VPC con enrutamiento privado

Una red de Amazon VPC existente sin acceso a Internet necesita otros puntos de conexión de servicio de VPC (AWS PrivateLink) para usar Apache Airflow en Amazon Managed Workflows para Apache Airflow. En esta página, se describen los puntos de conexión de VPC necesarios para los servicios de AWS que utiliza Amazon MWAA, los puntos de conexión de VPC necesarios para Apache Airflow, y cómo crear y conectar los puntos de conexión de VPC a una Amazon VPC existente con enrutamiento privado.

Contenido

Precios

- AWS PrivateLink Precios de

Red privada y enrutamiento privado

El modo de acceso de red privada limita el acceso a la UI de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno.

Al crear un entorno con acceso mediante red privada al servidor web, debe empaquetar todas sus dependencias en un archivo wheel de Python (`.whl`) y luego hacer referencia al `.whl` en

su `requirements.txt`. Para obtener instrucciones sobre cómo empaquetar e instalar sus dependencias mediante el archivo wheel, consulte cómo administrar dependencias con archivos wheel de Python.

En la siguiente imagen, se muestra dónde se encuentra la opción de red privada en la consola de Amazon MWAA.

- Enrutamiento privado. Una Amazon VPC sin acceso a Internet limita el tráfico de red dentro de la VPC. En esta página, se asume que su Amazon VPC no tiene acceso a Internet y requiere puntos de conexión de VPC para cada servicio de AWS que use su entorno, así como puntos de conexión de VPC para Apache Airflow en la misma Región de AWS y Amazon VPC que su entorno de Amazon MWAA.

(Obligatorio) Puntos de conexión de VPC

En la siguiente sección, se muestran los puntos de conexión de VPC necesarios para una Amazon VPC sin acceso a Internet. Enumera los puntos de conexión de VPC de cada servicio de AWS que utiliza Amazon MWAA, incluidos los puntos de conexión de VPC necesarios para Apache Airflow.

```
com.amazonaws.us-east-1.s3
com.amazonaws.us-east-1.monitoring
com.amazonaws.us-east-1.logs
com.amazonaws.us-east-1.sqs
com.amazonaws.us-east-1.kms
```

> **Note**
>
> Si utiliza una puerta de enlace de tránsito o cualquier otra ruta que no vaya directamente a los puntos de conexión de la API de AWS, le recomendamos agregar AWS PrivateLink a las subredes privadas de Amazon MWAA para los siguientes servicios:
>
> - Amazon S3
> - Amazon SQS
> - Registros de CloudWatch
> - Métricas de CloudWatch
> - AWS KMS (si corresponde)

Esto garantiza que su entorno de Amazon MWAA pueda comunicarse de forma segura y eficiente con estos servicios sin enrutar el tráfico a través de la Internet pública, lo que mejora la seguridad y el rendimiento.

Conexión de los puntos de conexión de VPC necesarios

En esta sección se describen los pasos para conectar los puntos de conexión de VPC necesarios para una Amazon VPC con enrutamiento privado.

Puntos de conexión de VPC necesarios para los servicios de AWS

En la siguiente sección, se muestran los pasos para conectar los puntos de conexión de VPC de los servicios de AWS usados por un entorno a una Amazon VPC existente.

Cómo conectar puntos de conexión de VPC a sus subredes privadas

1. Abra la página Puntos de conexión de la consola de Amazon VPC.
2. Seleccione su Región de AWS.
3. Cree el punto de conexión de Amazon S3:
    a. Elija Crear punto de conexión.
    b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.s3`** y, a continuación, pulse Intro en el teclado.
    c. Se recomienda elegir el punto de conexión del servicio que aparece en la lista para el tipo de puerta de enlace.

       Por ejemplo: ., **`com.amazonaws.us-west-2.s3 amazon Gateway`**
    d. Elija la Amazon VPC de su entorno en VPC.
    e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que ese DNS privado esté habilitado seleccionando Habilitar nombre de DNS.
    f. Elija los grupos de seguridad Amazon VPC de su entorno.
    g. Seleccione Acceso completo en Política.
    h. Seleccione Crear punto de conexión.

4. Cree el punto de conexión para CloudWatch Logs:
    a. Elija Crear punto de conexión.
    b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.logs`** y, a continuación, pulse Intro en el teclado.
    c. Seleccione el punto de conexión del servicio.
    d. Elija la Amazon VPC de su entorno en VPC.
    e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.
    f. Elija los grupos de seguridad Amazon VPC de su entorno.
    g. Seleccione Acceso completo en Política.
    h. Seleccione Crear punto de conexión.
5. Cree el punto de conexión para Monitoreo de CloudWatch:
    a. Elija Crear punto de conexión.
    b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.monitoring`** y, a continuación, pulse Intro en el teclado.
    c. Seleccione el punto de conexión del servicio.
    d. Elija la Amazon VPC de su entorno en VPC.
    e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.
    f. Elija los grupos de seguridad Amazon VPC de su entorno.
    g. Seleccione Acceso completo en Política.
    h. Seleccione Crear punto de conexión.
6. Cree el punto de conexión para Amazon SQS:
    a. Elija Crear punto de conexión.
    b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.sqs`** y, a continuación, pulse Intro en el teclado.
    c. Seleccione el punto de conexión del servicio.
    d. Elija la Amazon VPC de su entorno en VPC.
    e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.

f. Elija los grupos de seguridad Amazon VPC de su entorno.

g. Seleccione Acceso completo en Política.

h. Seleccione Crear punto de conexión.

7. Cree el punto de conexión para AWS KMS:

a. Elija Crear punto de conexión.

b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.kms`** y, a continuación, pulse Intro en el teclado.

c. Seleccione el punto de conexión del servicio.

d. Elija la Amazon VPC de su entorno en VPC.

e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.

f. Elija los grupos de seguridad Amazon VPC de su entorno.

g. Seleccione Acceso completo en Política.

h. Seleccione Crear punto de conexión.

Puntos de conexión de VPC necesarios para Apache Airflow

En la siguiente sección, se muestran los pasos para conectar los puntos de conexión de VPC de Apache Airflow a una Amazon VPC existente.

Cómo conectar puntos de conexión de VPC a sus subredes privadas

1. Abra la página Puntos de conexión de la consola de Amazon VPC.
2. Seleccione su Región de AWS.
3. Cree el punto de conexión para la API Apache Airflow:

a. Elija Crear punto de conexión.

b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.airflow.api`** y, a continuación, pulse Intro en el teclado.

c. Seleccione el punto de conexión del servicio.

d. Elija la Amazon VPC de su entorno en VPC.

e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.

f. Elija los grupos de seguridad Amazon VPC de su entorno.

g. Seleccione Acceso completo en Política.

h. Seleccione Crear punto de conexión.

4. Cree el primer punto de conexión para el entorno Apache Airflow:

a. Elija Crear punto de conexión.

b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.airflow.env`** y, a continuación, pulse Intro en el teclado.

c. Seleccione el punto de conexión del servicio.

d. Elija la Amazon VPC de su entorno en VPC.

e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.

f. Elija los grupos de seguridad Amazon VPC de su entorno.

g. Seleccione Acceso completo en Política.

h. Seleccione Crear punto de conexión.

5. Cree el segundo punto de conexión para las operaciones de Apache Airflow:

a. Elija Crear punto de conexión.

b. En el campo de texto Filtrar por atributos o buscar por palabra clave, escriba: **`.airflow.ops`** y, a continuación, pulse Intro en el teclado.

c. Seleccione el punto de conexión del servicio.

d. Elija la Amazon VPC de su entorno en VPC.

e. Asegúrese de que sus dos subredes privadas en distintas zonas de disponibilidad estén seleccionadas y de que Habilitar nombre de DNS esté activado.

f. Elija los grupos de seguridad Amazon VPC de su entorno.

g. Seleccione Acceso completo en Política.

h. Seleccione Crear punto de conexión.

(Opcional) Habilite las direcciones IP privadas para el punto de conexión de la interfaz de VPC de Amazon S3

Los puntos de conexión de la interfaz de Amazon S3 no admiten DNS privados. Las solicitudes de punto de conexión de S3 aún se resuelven en una dirección IP pública. Para resolver la dirección

S3 a una dirección IP privada, debe agregar una zona alojada privada en Route 53 para el punto de conexión regional de S3.

Uso de Route 53

En esta sección, se describen los pasos para habilitar las direcciones IP privadas para un punto de conexión de la interfaz S3 mediante Route 53.

1. Cree una zona alojada privada para el punto de conexión de la interfaz de Amazon S3 VPC (como s3.eu-west-1.amazonaws.com) y asóciela a su Amazon VPC.
2. Cree un registro ALIAS A para el punto de conexión de la interfaz de VPC de Amazon S3 (como s3.eu-west-1.amazonaws.com) que se resuelva al nombre de DNS del punto de conexión de la interfaz de VPC.
3. Cree un registro comodín ALIAS A para el punto de conexión de la interfaz de Amazon S3 (por ejemplo, *. s3.eu-west-1.amazonaws.com) que se resuelva al nombre de DNS del punto de conexión de la interfaz de la VPC.

VPC con DNS personalizado

Si su Amazon VPC utiliza un enrutamiento de DNS personalizado, debe realizar los cambios en su solucionador de DNS (distinto a Route 53, que normalmente es una instancia de EC2 que ejecuta un servidor DNS) mediante la creación de un registro CNAME. Por ejemplo:

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value:  *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

Administración de sus propios puntos de conexión de Amazon VPC en Amazon MWAA

Amazon MWAA utiliza los puntos de conexión de Amazon VPC para integrarse con varios servicios de AWS necesarios para configurar un entorno de Apache Airflow. La administración de sus propios puntos de conexión tiene dos casos de uso principales:

1. Esto significa que puede crear entornos de Apache Airflow en una Amazon VPC compartida cuando utiliza una AWS Organizations para administrar varias Cuentas de AWS y compartir recursos.

2. Le permite usar políticas de acceso más restrictivas, ya que limita sus permisos a los recursos específicos que usan sus puntos de conexión.

Si decide administrar sus propios puntos de conexión de VPC, debe crear sus propios puntos de conexión para la base de datos de RDS para PostgreSQL y para el servidor web del entorno.

Para obtener más información sobre cómo Amazon MWAA implementa Apache Airflow en la nube, consulte el diagrama de arquitectura de Amazon MWAA.

> **Important**
>
> Amazon MWAA no valida la selección del tipo de dirección IP (`AddressType`) para los puntos de conexión administrados por el cliente, por lo que debe especificar el `AddressType` correctamente (las opciones válidas son IPv4 o IPv6).

Creación de entornos en una Amazon VPC compartida

Si usa AWS Organizations para administrar varias cuentas de Cuentas de AWS que comparten recursos, puede usar puntos de conexión de VPC administrados por el cliente con Amazon MWAA para compartir los recursos del entorno con otra cuenta en su organización.

Cuando configura el acceso compartido a la VPC, la cuenta propietaria de la Amazon VPC principal (propietaria) comparte las dos subredes privadas requeridas por Amazon MWAA con otras cuentas (participantes) que pertenecen a la misma organización. Las cuentas participantes que comparten esas subredes pueden ver, crear, modificar y eliminar entornos en la Amazon VPC compartida.

Suponga que tiene una cuenta, `Owner`, que actúa como la cuenta `Root` de la organización y es propietaria de los recursos de Amazon VPC, y una cuenta participante, `Participant`, que es miembro de la misma organización. Cuando `Participant` crea un nuevo Amazon MWAA en la Amazon VPC que comparte con `Owner`, Amazon MWAA primero creará los recursos de la VPC de servicio y luego entrará en un estado `PENDING` durante un máximo de 72 horas.

Después de que el estado del entorno cambia de `CREATING` a `PENDING`, una entidad principal que actúa en nombre de `Owner` crea los puntos de conexión requeridos. Para ello, Amazon MWAA muestra la base de datos y el punto de conexión del servidor web en la consola de Amazon MWAA. También puede llamar a la acción de la API `GetEnvironment` para obtener los puntos de conexión del servicio.

> **Note**
>
> Si la Amazon VPC que utiliza para compartir recursos es una Amazon VPC privada, debe seguir los pasos descritos en the section called “Administración del acceso a puntos de conexión de VPC”. El tema trata sobre la configuración de un conjunto diferente de puntos de conexión de Amazon VPC relacionados con otros servicios de AWS con los que AWS se integra, como Amazon ECR, Amazon ECS y Amazon SQS. Estos servicios son fundamentales para operar y administrar su entorno de Apache Airflow en la nube.

Requisitos previos

Antes de crear un entorno de Amazon MWAA en una VPC compartida, necesita los siguientes recursos:

- Una Cuenta de AWS, `Owner`, que se usará como la cuenta propietaria de la Amazon VPC.
- Una unidad organizativa AWS Organizations, `MyOrganization`, creada como una raíz.
- Una segunda Cuenta de AWS, `Participant`, en `MyOrganization` para servir a la cuenta participante que crea el nuevo entorno.

Además, le recomendamos que se familiarice con las responsabilidades y los permisos para propietarios y participantes a la hora de compartir recursos en la Amazon VPC.

Creación de la Amazon VPC

En primer lugar, cree una nueva Amazon VPC que compartirán las cuentas propietaria y participante:

1. Inicie sesión en la consola mediante `Owner` y, a continuación, abra la consola de CloudFormation. Use la siguiente plantilla para crear una pila. Esta pila proporciona una serie de recursos de red, incluida una Amazon VPC y las subredes que las dos cuentas compartirán en este escenario.

```
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. Después de que se hayan aprovisionado los nuevos recursos de la Amazon VPC, navegue a la consola de AWS Resource Access Manager y luego elija Crear recurso compartido.
3. Elija las subredes que creó en el primer paso de la lista de subredes disponibles que puede compartir con `Participant`.

Crear el entorno

Complete los siguientes pasos para crear entornos de Amazon MWAA con puntos de conexión de Amazon VPC administrados por el cliente.

1. Inicie sesión mediante `Participant` y abra la consola de Amazon MWAA. Complete el primer paso: especificar los detalles para especificar un bucket de Amazon S3, una carpeta DAG y las dependencias para su nuevo entorno. Para obtener más información, consulte cómo empezar.
2. En la página Configurar ajustes avanzados, en Redes, seleccione las subredes de la Amazon VPC compartida.
3. En Administración del punto de conexión, seleccione CLIENTE en la lista desplegable.
4. Mantenga el valor predeterminado para el resto de las opciones de la página y, a continuación, seleccione Crear entorno en la página Revisar y crear.

El entorno comienza en un estado `CREATING` y, a continuación, cambia a `PENDING`. Cuando el entorno esté en `PENDING`, anote el nombre del servicio de punto de conexión de la base de datos y el nombre del servicio de punto de conexión del servidor web (si configuró un servidor web privado) mediante la consola.

Cuándo crea un entorno nuevo con la consola de Amazon MWAA. Amazon MWAA crea un nuevo grupo de seguridad con las reglas de entrada y salida necesarias. Anote el ID del grupo de seguridad.

En la siguiente sección, `0wner` utilizará los puntos de conexión del servicio y el ID del grupo de seguridad para crear nuevos puntos de conexión de Amazon VPC en la Amazon VPC compartida.

Creación de los puntos de conexión de Amazon VPC

Complete los siguientes pasos para crear puntos de conexión de Amazon VPC para su entorno.

1. Inicie sesión en la Consola de administración de AWS mediante `0wner` y abra https://console.aws.amazon.com/vpc/.
2. Seleccione Grupos de seguridad en el panel de navegación izquierdo y, a continuación, cree un nuevo grupo de seguridad en la Amazon VPC compartida con las siguientes reglas de entrada y salida:

	Tipo	Protocolo	Tipo de origen	Origen
Entrante	Todo el tráfico	Todos	Todos	El grupo de seguridad de su entorno
Salida	Todo el tráfico	Todos	Todos	`0.0.0.0/0`

> **Warning**
>
> La cuenta `0wner` debe configurar un grupo de seguridad en la cuenta `0wner` para permitir el tráfico desde el nuevo entorno a la Amazon VPC compartida. Para hacerlo, puede crear un nuevo grupo de seguridad en `0wner` o editar uno existente.

3. Elija Endpoints (Puntos de conexión) y, a continuación, cree nuevos puntos de conexión para la base de datos del entorno y el servidor web (si están en modo privado) con los nombres de los servicios de puntos de conexión de los pasos anteriores. Elija la Amazon VPC compartida, las subredes que utilizó para el entorno y el grupo de seguridad del entorno.

Si se ejecuta correctamente, el entorno cambiará de `PENDING` a `CREATING` y, finalmente, a `AVAILABLE`. Cuando esté `AVAILABLE`, podrá iniciar sesión en la consola de Apache Airflow.

Resolución de problemas de la Amazon VPC compartida

Utilice la siguiente referencia para resolver los problemas que surjan cuando crea entornos en una Amazon VPC compartida.

El entorno se encuentra en **`CREATE_FAILED`** luego del estado **`PENDING`**

- Compruebe que `Owner` comparte las subredes con `Participant` utilizando AWS Resource Access Manager.
- Compruebe que los puntos de conexión de Amazon VPC para la base de datos y el servidor web se hayan creado en las mismas subredes asociadas con el entorno.
- Compruebe que el grupo de seguridad que se usa con sus puntos de conexión permita el tráfico de los grupos de seguridad que se usan para el entorno. La cuenta `Owner` crea reglas que hacen referencia al grupo de seguridad en `Participant` como *`123456789012`*/*`security-group-id`*.

Tipo	Protocolo	Tipo de origen	Origen
Todo el tráfico	Todos	Todos	*`123456789012`* /*`sg-0909e8e81919`*

  Para obtener más información, consulte las responsabilidades y permisos para propietarios y participantes

El entorno está atascado en estado **`PENDING`**

Compruebe el estado de cada punto de conexión de VPC para asegurarse de que está `Available`. Si configura un entorno con un servidor web privado, también debe crear un punto de conexión para el servidor web. Si el entorno está atascado en `PENDING`, esto podría indicar que falta el punto de conexión del servidor web privado.

Error **`The Vpc Endpoint Service '`*`vpce-service-name`*`' does not exist`** recibido

Si aparece el siguiente error, compruebe que la cuenta que crea los puntos de conexión en la cuenta `Owner` es la propietaria de la VPC compartida:

```
ClientError: An error occurred (InvalidServiceName) when calling the
 CreateVpcEndpoint operation:
```

```
The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Tutoriales de Amazon Managed Workflows para Apache Airflow

Esta guía incluye tutoriales paso a paso sobre el uso y la configuración de un entorno Amazon Managed Workflows para Apache Airflow.

Temas

Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN

En este tutorial, se explican los pasos necesarios para crear un túnel de VPN desde su equipo hasta el servidor web de Apache Airflow para su entorno de Amazon Managed Workflows para Apache Airflow. Para conectarse a Internet a través de un túnel de VPN, primero tendrá que crear un punto de conexión de AWS Client VPN. Una vez configurado, un punto de conexión Client VPN actúa como servidor VPN, lo que permite una conexión segura desde el equipo a los recursos de la VPC. A continuación, se conectará a la Client VPN desde su equipo mediante la AWS Client VPN para escritorio.

Secciones

Red privada

En este tutorial, se asume que ha elegido el modo de acceso red privada para su servidor web de Apache Airflow.

El modo de acceso de red privada limita el acceso a la UI de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno.

Al crear un entorno con acceso mediante red privada al servidor web, debe empaquetar todas sus dependencias en un archivo wheel de Python (`.whl`) y luego hacer referencia al `.whl` en su `requirements.txt`. Para obtener instrucciones sobre cómo empaquetar e instalar sus dependencias mediante el archivo wheel, consulte cómo administrar dependencias con archivos wheel de Python.

En la siguiente imagen, se muestra dónde se encuentra la opción de red privada en la consola de Amazon MWAA.

Casos de uso

Puede utilizar este tutorial antes o después de haber creado un entorno Amazon MWAA. Debe usar la misma Amazon VPC, los mismos grupos de seguridad de VPC y las mismas subredes privadas que su entorno. Si usa este tutorial después de haber creado un entorno de Amazon MWAA, una vez que haya completado los pasos, podrá volver a la consola de Amazon MWAA y cambiar el modo de acceso al servidor web Apache Airflow a red privada.

Antes de empezar

1. Compruebe los permisos de usuario. Asegúrese de que su cuenta en AWS Identity and Access Management (IAM) tenga los permisos necesarios para crear y administrar los recursos de VPC.

2. Utilice su VPC de Amazon MWAA. En este tutorial, se asume que está asociando la Client VPN a una VPC existente. La VPC de Amazon debe estar en la misma Región de AWS que un entorno de Amazon MWAA y tener dos subredes privadas. Si no ha creado una Amazon VPC, utilice la plantilla de CloudFormation en Opción 3: Creación de una red de Amazon VPC sin acceso a Internet.

Objetivos

En este tutorial, hará lo siguiente:

1. Crear un punto de conexión AWS Client VPN mediante una plantilla de CloudFormation para una Amazon VPC existente.
2. Generar certificados y claves de servidor y cliente y, a continuación, cargar el certificado y la clave del servidor en AWS Certificate Manager en la misma Región de AWS que un entorno de Amazon MWAA.
3. Descargar y modificar un archivo de configuración de punto de conexión de Client VPN para su Client VPN y utilizar el archivo para crear un perfil de VPN para conectarse mediante Client VPN para escritorio.

(Opcional) Paso uno: identificar la VPC, las reglas de CIDR y la seguridad de la VPC

En la siguiente sección, se describe cómo encontrar los ID de su Amazon VPC, su grupo de seguridad de VPC y una forma de identificar las reglas de CIDR que necesitará para crear su Client VPN en los pasos siguientes.

Identificación de las reglas de su CIDR

En la siguiente sección, se muestra cómo identificar las reglas de su CIDR, que necesitará para crear su Client VPN.

Identificación del CIDR de su Client VPN

1. Abra la página Sus Amazon VPC en la consola de Amazon VPC.
2. Use el selector de región de la barra de navegación para elegir la misma Región de AWS que un entorno de Amazon MWAA.

3. Elija su Amazon VPC.
4. Supongamos que los CIDR de sus subredes privadas son:

   - Subred privada 1: 10.192.10.0 /24
   - Subred privada 2: 10.192.11.0 /24

   Si el CIDR de su Amazon VPC es 10.192.0.0/16, el CIDR de IPv4 de cliente que especificaría para su Client VPN sería 10.192.0.0/22.
5. Guarde este valor de CIDR y el valor de su ID de VPC para los pasos siguientes.

Identifique su VPC y los grupos de seguridad

En la siguiente sección, se muestra cómo encontrar el identificador de VPC de Amazon y de sus grupos de seguridad, que necesitará para crear su Client VPN.

> **Note**
>
> Puede que esté usando varios grupos de seguridad. Deberá especificar todos los grupos de seguridad de la VPC en los pasos siguientes.

Pasos para identificar los grupos de seguridad

1. Abra la página Grupos de seguridad en la consola de Amazon VPC.
2. En la barra de navegación, use el selector de región para elegir la Región de AWS.
3. Busque la VPC de Amazon en el identificador de la VPC e identifique los grupos de seguridad asociados a la VPC.
4. Guarde el identificador de sus grupos de seguridad y de la VPC para los pasos siguientes.

Paso dos: crear los certificados de servidor y cliente

Los puntos de enlace de Client VPN solo admiten claves RSA con un tamaño de 1024 bits y 2048 bits. En la sección siguiente, se explica cómo usar easy-rsa de OpenVPN para generar los certificados y las claves del servidor y el cliente y, luego, cargar los certificados en ACM mediante la AWS Command Line Interface (AWS CLI).

Creación de certificados de cliente

1. Siga estos pasos rápidos para crear y cargar los certificados en ACM mediante la AWS CLI en autenticación y autorización de clientes: autenticación mutua.
2. En estos pasos, debe especificar la misma Región de AWS que un entorno de Amazon MWAA en el comando de la AWS CLI al cargar los certificados de servidor y cliente. A continuación, se muestran algunos ejemplos de cómo especificar la región en estos comandos:

   a. Example región para el 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 región para el 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. Tras estos pasos, guarde el valor devuelto en la respuesta de la AWS CLI para los ARN del certificado de servidor y del certificado de cliente. Deberá especificar estos ARN en la plantilla de CloudFormation para crear la Client VPN.
3. En estos pasos, se guardan un certificado de cliente y una clave privada en su equipo. A continuación, se muestra un ejemplo de dónde encontrar estas credenciales:

   a. Example en macOS

   En macOS, el contenido se guarda en `/Users/your-user/custom_folder`. Si incluye todos los contenidos (`ls -a`) de este directorio, debería ver algo parecido a lo siguiente:

   ```
   .
   ..
   ca.crt
   client1.domain.tld.crt
   client1.domain.tld.key
   server.crt
   server.key
   ```

b. Tras estos pasos, guarde el contenido o anote la ubicación del certificado de cliente en `client1.domain.tld.crt` y la clave privada en `client1.domain.tld.key`. Deberá agregar estos valores al archivo de configuración de su Client VPN.

Paso tres: guardar la plantilla de CloudFormation localmente

La siguiente sección contiene la plantilla de CloudFormation para crear la Client VPN. Debe especificar la misma VPC de Amazon, los mismos grupos de seguridad de VPC y las mismas subredes privadas que su entorno Amazon MWAA.

- Copie el contenido de la siguiente plantilla y guárdelo localmente como `mwaa_vpn_client.yaml`. También puede descargar la plantilla.

  Sustituya los valores siguientes:

  - **`YOUR_CLIENT_ROOT_CERTIFICATE_ARN`**: el ARN de su certificado client1.domain.tld en `ClientRootCertificateChainArn`.
  - **`YOUR_SERVER_CERTIFICATE_ARN`**: el ARN del certificado de su servidor en `ServerCertificateArn`.
  - La regla CIDR de IPv4 del cliente en `ClientCidrBlock`. Se proporciona una regla CIDR de `10.192.0.0/22`.
  - Su ID de Amazon VPC en `VpcId`. Se proporciona un VPC de `vpc-010101010101`.
  - Sus identificadores de grupo de seguridad de VPC en `SecurityGroupIds`. Se proporciona un grupo de seguridad 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

Si utiliza más de un grupo de seguridad para su entorno, puede especificar varios grupos de seguridad en el siguiente formato:

```
SecurityGroupIds:
 - sg-0112233445566778b
 - sg-0223344556677889f
```

Paso cuatro: crear la pila CloudFormation de Client VPN

Cómo crear el AWS Client VPN

1. Abra la consola de AWS CloudFormation.
2. Seleccione La plantilla está lista y Cargar un archivo de plantilla.
3. Seleccione Elegir archivo y seleccione su archivo `mwaa_vpn_client.yaml`.
4. Seleccione Siguiente, Siguiente.
5. Seleccione la casilla de confirmación y, a continuación, Crear pila.

Paso cinco: asociar subredes a su Client VPN

Cómo asociar subredes privadas a la AWS Client VPN

1. Abra la consola de Amazon VPC.
2. Seleccione la página Puntos de enlace de Client VPN.
3. Seleccione su Client VPN y, a continuación, elija la pestaña Asociaciones, Asociar.
4. Elija lo siguiente en la lista desplegable:
   - Su Amazon VPC en VPC.
   - Una de sus subredes privadas en Elegir una subred para asociar.
5. Elija Asociar.

> **Note**
>
> La VPC y la subred tardan varios minutos en asociarse a Client VPN.

Paso seis: agregar una regla de entrada de autorizaciones al Client VPN

Debe agregar una regla de entrada de autorización mediante la regla CIDR para su VPC a su Client VPN. Si desea autorizar usuarios o grupos específicos de su grupo de Active Directory o proveedor de identidad (IdP) basado en SAML, consulte las reglas de autorización en la guía de Client VPN.

Cómo añadir el CIDR a la AWS Client VPN

1. Abra la consola de Amazon VPC.
2. Seleccione la página Puntos de enlace de Client VPN.
3. Seleccione su Client VPN y, a continuación, elija la pestaña Autorización, Autorizar entrada.
4. Especifique lo siguiente:
   - La regla CIDR de su Amazon VPC en Red de destino que se habilitará. Por ejemplo:

     ```
     10.192.0.0/16
     ```
   - En Conceder acceso a, elija Permitir el acceso a todos los usuarios.

- En Descripción, escriba un nombre descriptivo.

5. Seleccione Agregar regla de autorización.

> **Note**
>
> Según los componentes de red de su VPC de Amazon, es posible que también necesite esta regla de autorización de entrada a su lista de control de acceso a la red (NACL).

Paso siete: descargar el archivo de configuración del punto de conexión de Client VPN

Cómo descargar el archivo de configuración

1. Siga estos pasos rápidos para descargar el archivo de configuración de Client VPN en Descargar el archivo de configuración de punto de conexión de Client VPN.
2. En estos pasos, se le pedirá que añada una cadena al nombre de DNS del punto de conexión de Client VPN. A continuación se muestra un ejemplo:

   - Example nombre de DNS del punto de conexión

     Si el nombre de DNS del punto de conexión de Client VPN tiene el siguiente aspecto:

     ```
     remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
     ```

     Puede agregar una cadena para identificar el punto de conexión de Client VPN de la siguiente manera:

     ```
     remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-
     west-1.amazonaws.com 443
     ```

3. En estos pasos, se le pide que agregue el contenido del certificado de cliente entre un nuevo conjunto de etiquetas `<cert></cert>` y el contenido de la clave privada entre un nuevo conjunto de etiquetas `<key></key>`. A continuación se muestra un ejemplo:

   a. Abra un símbolo del sistema y cambie los directorios a la ubicación del certificado de cliente y la clave privada.

b. Example macOS client1.domain.tld.crt

Para mostrar el contenido del archivo `client1.domain.tld.crt` en macOS, puede usar `cat client1.domain.tld.crt`.

Copie el valor del terminal y péguelo en `downloaded-client-config.ovpn` así:

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

c. Example macOS client1.domain.tld.key

Para mostrar el contenido de `client1.domain.tld.key`, puede utilizar `cat client1.domain.tld.key`.

Copie el valor del terminal y péguelo en `downloaded-client-config.ovpn` así:

```
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>
```

Paso ocho: conectarse a la AWS Client VPN

El cliente para AWS Client VPN se proporciona de forma gratuita. Puede conectar su equipo directamente a AWS Client VPN para disfrutar de una experiencia de VPN integral.

Conexión a Client VPN

1. Descargue e instale AWS Client VPN para escritorio.
2. Abra la AWS Client VPN.
3. Seleccione Archivo, Perfiles administrados en el menú del cliente VPN.
4. Seleccione Agregar perfil y, a continuación, elija `downloaded-client-config.ovpn`.
5. Introduzca un nombre descriptivo en Nombre público.
6. Seleccione Agregar perfil, Listo.
7. Elija Conectar.

Después de conectarse a Client VPN, tendrá que desconectarse de otras VPN para ver cualquiera de los recursos de su VPC de Amazon.

> **Note**
>
> Es posible que tenga que salir del cliente y empezar de nuevo antes de poder conectarse.

Siguientes pasos

- Obtenga información sobre cómo crear un entorno Amazon MWAA en Introducción a Amazon Managed Workflows para Apache Airflow. Debe crear un entorno en la misma Región de AWS que Client VPN y usar la misma VPC, las mismas subredes privadas y el mismo grupo de seguridad que Client VPN.

Tutorial: Configuración del acceso a la red privada mediante un host bastión de Linux

En este tutorial, se explican los pasos necesarios para crear un túnel SSH desde su equipo hasta el servidor web de Apache Airflow para su entorno de Amazon Managed Workflows para Apache Airflow. Se asume que ya ha creado un entorno Amazon MWAA. Una vez configurado, un host bastión de Linux actúa como jump server que permite una conexión segura desde el equipo a los recursos de la VPC. A continuación, debe utilizar un complemento de administración de proxy SOCKS para controlar la configuración del proxy en su navegador y acceder a la interfaz de usuario de Apache Airflow.

Secciones

Red privada

En este tutorial, se asume que ha elegido el modo de acceso red privada para su servidor web de Apache Airflow.

El modo de acceso de red privada limita el acceso a la UI de Apache Airflow a los usuarios de su Amazon VPC a los que se les ha concedido acceso a la política de IAM de su entorno.

Al crear un entorno con acceso mediante red privada al servidor web, debe empaquetar todas sus dependencias en un archivo wheel de Python (`.whl`) y luego hacer referencia al `.whl` en su `requirements.txt`. Para obtener instrucciones sobre cómo empaquetar e instalar sus dependencias mediante el archivo wheel, consulte cómo administrar dependencias con archivos wheel de Python.

En la siguiente imagen, se muestra dónde se encuentra la opción de red privada en la consola de Amazon MWAA.

Casos de uso

Puede utilizar este tutorial después de haber creado un entorno Amazon MWAA. Debe usar la misma Amazon VPC, los mismos grupos de seguridad de VPC y las mismas subredes públicas que su entorno.

Antes de empezar

1. Compruebe los permisos de usuario. Asegúrese de que su cuenta en AWS Identity and Access Management (IAM) tenga los permisos necesarios para crear y administrar los recursos de VPC.
2. Utilice su VPC de Amazon MWAA. En este tutorial, se asume que está asociando el host bastión a una VPC existente. La Amazon VPC debe estar en la misma región que su entorno de Amazon MWAA y tener dos subredes privadas, tal y como se define en Creación de la red de VPC.
3. Cree una clave de SSH. Debe crear una clave SSH de Amazon EC2 (.pem) en la misma región que su entorno de Amazon MWAA para conectarse a los servidores virtuales. Si no tiene una clave SSH, consulte cómo crear o importar un par de claves en la guía del usuario de Amazon EC2.

Objetivos

En este tutorial, hará lo siguiente:

1. Crear una instancia de host bastión de Linux mediante una plantilla de CloudFormation para una VPC existente.
2. Autorizar el tráfico entrante al grupo de seguridad de la instancia bastión mediante una regla de entrada en el puerto 22.
3. Autorizar el tráfico de entrada del grupo de seguridad de un entorno de Amazon MWAA al grupo de seguridad de la instancia de bastión.
4. Crear un túnel SSH hasta la instancia de bastión.
5. Instalar y configurar el complemento FoxyProxy para el navegador Firefox para ver la UI de Apache Airflow.

Paso uno: crear la instancia del bastión

En la siguiente sección, se describen los pasos para crear la instancia de bastión de Linux mediante una plantilla de CloudFormation para una VPC existente en la consola de CloudFormation.

Creación del host bastión de Linux

1. Abra la página implementar el inicio rápido en la consola CloudFormation.
2. Use el selector de región de la barra de navegación para elegir la misma Región de AWS que su entorno de Amazon MWAA.
3. Elija Siguiente.
4. Escriba un nombre en el campo de texto Stack name (Nombre de pila), como `mwaa-linux-bastion`.
5. En el panel Parámetros, Configuración de red, elija las siguientes opciones:
   a. Elija el ID de VPC de su entorno de Amazon MWAA.
   b. Elija el ID de subred pública 1 de su entorno de Amazon MWAA.
   c. Elija el ID de subred pública 2 de su entorno de Amazon MWAA.
   d. Introduzca el rango de direcciones más acotado posible (por ejemplo, un rango CIDR interno) en CIDR de acceso externo al bastión permitido.

   > **Note**
   >
   > La forma más sencilla de identificar un rango es usar el mismo rango de CIDR que las subredes públicas. Por ejemplo, las subredes públicas en la plantilla CloudFormation en la página Creación de la red de VPC son `10.192.10.0/24` y `10.192.11.0/24`.

6. En el panel Configuración de Amazon EC2, elija lo siguiente:
   a. Elija su clave SSH en la lista desplegable Nombre del par de claves.
   b. Introduzca un nombre en Nombre del host bastión.
   c. En Reenvío de TCP, elija verdadero.

> **Warning**
>
> En este paso, el reenvío de TCP debe establecerse como verdadero. De lo contrario, no podrá crear un túnel SSH en el siguiente paso.

7. Seleccione Siguiente, Siguiente.
8. Seleccione la casilla de confirmación y, a continuación, Crear pila.

Para obtener más información sobre la arquitectura de su host bastión de Linux, consulte Hosts bastión de Linux en la nube de AWS: arquitectura.

Paso dos: crear el túnel SSH

Los siguientes pasos describen cómo crear el túnel SSH en el bastión de Linux. Un túnel SSH recibe la solicitud de su dirección IP local al bastión de Linux, que es por lo que en los pasos anteriores el reenvío TCP para el bastión de Linux se configuró como `true`.

macOS/Linux

Pasos para crear un túnel con la línea de comandos

1. Abra la página Instancias de la consola de Amazon EC2.
2. Elija una instancia.
3. Copie la dirección en DNS IPv4 público. Por ejemplo, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. En el símbolo del sistema, vaya hasta el directorio en el que está guardada la clave SSH.
5. Ejecute el comando siguiente para conectarse a la instancia del bastión mediante SSH. Sustituya el valor de muestra por el nombre de la clave SSH en `mykeypair.pem`.

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

Creación de un túnel con PuTTY

1. Abra la página Instancias de la consola de Amazon EC2.

2. Elija una instancia.
3. Copie la dirección en DNS IPv4 público. Por ejemplo, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Abra PuTTY y seleccione Sesión.
5. En Nombre de host, ingrese ec2-user@*YOUR_PUBLIC_IPV4_DNS* como nombre del host y 22 como puerto.
6. Expanda la pestaña SSH y seleccione Autent. En Archivo de clave privada para la autenticación, elija su archivo “ppk” local.
7. En SSH, seleccione la pestaña Túneles y, a continuación, seleccione las opciones Dinámico y Autom.
8. En Puerto de origen, agregue el puerto 8157 (o cualquier otro puerto que no se utilice) y, a continuación, deje el puerto Destino en blanco. Elija Agregar.
9. Seleccione la pestaña Sesión e introduzca un nombre de sesión. Por ejemplo: . `SSH Tunnel`.
10. Elija Guardar y Abrir.

> **Note**
>
> Puede que tenga que introducir una contraseña para la clave pública.

> **Note**
>
> Si recibe un error `Permission denied (publickey)`, le recomendamos que utilice la herramienta AWSSupport-TroubleShootSSH y seleccione Ejecutar esta automatización (consola) para solucionar los problemas de la configuración de SSH.

Paso tres: configurar el grupo de seguridad del bastión como regla de entrada

El acceso a los servidores y el acceso regular a Internet desde los servidores se permite con un grupo de seguridad de mantenimiento especial adjunto a esos servidores. Los siguientes pasos describen cómo configurar el grupo de seguridad bastión como fuente de tráfico entrante al grupo de seguridad de VPC de un entorno.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. En el panel Redes, elija Grupo de seguridad de VPC.
4. Elija Editar reglas de entrada.
5. Seleccione Agregar regla.
6. Elija el ID del grupo de seguridad de la lista desplegable Origen.
7. Deje las opciones restantes en blanco o establézcalas en sus valores predeterminados.
8. Seleccione Guardar reglas.

Paso cuatro: copiar la URL de Apache Airflow

En los siguientes pasos, se describe cómo abrir la consola de Amazon MWAA y copiar la URL en la interfaz de usuario de Apache Airflow.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Copie la URL en la interfaz de usuario de Airflow para los pasos siguientes.

Paso cinco: configurar los ajustes del proxy

Si utiliza un túnel de SSH con enrutamiento de puertos dinámico, debe utilizar un complemento de administración de proxy de SOCKS para controlar los ajustes del proxy en el navegador. Por ejemplo, puede usar la característica `--proxy-server` de Chromium para iniciar una sesión de navegador o usar la extensión FoxyProxy en el navegador Mozilla Firefox.

Opción uno: configurar un túnel SSH utilizando el enrutamiento de puertos local

Si no desea utilizar un proxy SOCKS, puede configurar un túnel SSH al a través del enrutamiento de puertos local. El siguiente comando de ejemplo accede a la interfaz web ResourceManager de Amazon EC2 mediante el enrutamiento del tráfico por el puerto local 8157.

1. Abra una nueva ventana del símbolo del sistema.
2. Escriba el siguiente comando para abrir un túnel SSH.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-vpce.us-
east-1.airflow.amazonaws.com:443 ubuntu@YOUR_PUBLIC_IPV4_DNS.us-
east-1.compute.amazonaws.com
```

`-L` hace referencia al uso de enrutamiento de puertos local que le permite especificar un puerto local usado para reenviar datos al puerto remoto identificado en el servidor web local del nodo.

3. Introduzca `http://localhost:8157/` en su navegador.

> **Note**
>
> Es posible que necesite usar `https://localhost:8157/`.

Opción dos: proxies con la línea de comandos

La mayoría de los navegadores web permiten configurar los proxies mediante una línea de comandos o un parámetro de configuración. Por ejemplo, con Chromium puede iniciar el navegador con el comando siguiente:

```
chromium --proxy-server="socks5://localhost:8157"
```

Esto inicia una sesión de navegador que utiliza el túnel SSH creado en los pasos anteriores para enviar sus solicitudes por proxy. Puede abrir la URL de su entorno privado de Amazon MWAA (con https://) de la siguiente manera:

```
https://YOUR_VPC_ENDPOINT_ID-vpce.us-east-1.airflow.amazonaws.com/home.
```

Opción tres: proxies que utilizan FoxyProxy para Mozilla Firefox

En el siguiente ejemplo se muestra una configuración de FoxyProxy Standard (versión 7.5.1) para Mozilla Firefox. FoxyProxy proporciona un conjunto de herramientas de administración de proxy. Le permite utilizar un servidor proxy para las URL que coincidan con los patrones correspondientes a los dominios utilizados por la interfaz de usuario de Apache Airflow.

1. En Firefox, abra la página de la extensión FoxyProxy Standard.
2. Seleccione Agregar a Firefox.
3. Elija Agregar.

4. Elija el icono de FoxyProxy en la barra de herramientas de su navegador y seleccione Opciones.
5. Copie el siguiente código y guárdelo localmente como `mwaa-proxy.json`. Sustituya el valor de ejemplo de *YOUR_HOST_NAME* por la URL de 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. En el panel Importar ajustes desde FoxyProxy 6.0+, seleccione Importar ajustes y seleccione el archivo `mwaa-proxy.json`.
7. Seleccione Aceptar.

Paso seis: abrir la interfaz de usuario de Apache Airflow

Los siguientes pasos describen cómo abrir la interfaz de usuario de Apache Airflow.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Elija Abrir interfaz de usuario de Airflow.

Siguientes pasos

- Aprenda a ejecutar los comandos de la CLI de Airflow en un túnel SSH para un host bastión en Referencia de los comandos de la CLI de Apache Airflow.
- Aprenda cómo cargar el código DAG a su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Tutorial: Restricciones de acceso de los usuarios de Amazon MWAA a un subconjunto de DAG

Amazon MWAA gestiona el acceso a su entorno asignando sus entidades principales de IAM a uno o más roles predeterminados de Apache Airflow. En el siguiente tutorial, se muestra cómo restringir

a los usuarios individuales de Amazon MWAA para que solo vean o interactúen con un DAG o un conjunto de DAG en concreto.

> **Note**
>
> Los pasos de este tutorial se pueden seguir con acceso federado, siempre que haya roles de IAM.

Temas

Requisitos previos

Para completar los pasos de este tutorial, deberá tener lo siguiente:

- Un entorno de Amazon MWAA con varios DAG
- Una entidad principal de IAM, `Admin` con permisos de AdministratorAccess, y un usuario de IAM, `MWAAUser`, como entidad principal para la que puede limitar el acceso al DAG. Para obtener más información sobre los roles de administrador, consulte la función de trabajo de administrador en la guía del usuario de IAM

  > **Note**
  >
  > No adjunte políticas de permisos directamente a sus usuarios de IAM. Recomendamos que configure roles de IAM que los usuarios puedan asumir para obtener acceso temporal a sus recursos de Amazon MWAA.

- Versión 2 de la AWS Command Line Interface instalada.

Paso 1: dé acceso al servidor web de Amazon MWAA a su entidad principal de IAM con el rol predeterminado **`Public`** de Apache Airflow.

Cómo conceder permiso mediante la Consola de administración de AWS

1. Inicie sesión en la Cuenta de AWS con permisos de `Admin` y abra la consola de IAM.
2. En el panel de navegación izquierdo, elija Usuarios y, a continuación, elija su usuario de IAM de Amazon MWAA en la tabla de usuarios.
3. En la página de información de usuario, en Resumen, vaya a la pestaña Permisos, luego a Políticas de permisos para expandir la tarjeta y seleccione Agregar permisos.
4. En la sección Configurar permisos, elija Adjuntar directamente las políticas existentes y, a continuación, Crear política.
5. En la página Crear política, elija JSON y, a continuación, copie y pegue la siguiente política de permisos de JSON en el editor de políticas. Esta política da acceso al servidor web al usuario con el rol predeterminado `Public` de Apache Airflow.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
            "arn:aws:airflow:us-
east-1:111122223333:role/YOUR_ENVIRONMENT_NAME/Public"
            ]
        }
    ]
}
```

Paso 2: crear un nuevo rol personalizado de Apache Airflow

Pasos para crear un nuevo rol mediante la interfaz de usuario de Apache Airflow

1. Con su rol de IAM de administrador, abra la consola de Amazon MWAA e inicie la interfaz de usuario de Apache Airflow de su entorno.
2. En el panel de navegación de la parte superior, pase el cursor por Security (Seguridad) para abrir la lista desplegable y, a continuación, seleccione List Roles (Mostrar roles) para ver los roles predeterminados de Apache Airflow.
3. En la lista de roles, elija User (Usuario) y, en la parte superior de la página, Actions (Acciones) para abrir el menú desplegable. Elija Copiar rol y confirme con Aceptar

   > **Note**
   >
   > Copie los roles de Operaciones o Lector para conceder más o menos accesos, respectivamente.

4. Localice el nuevo rol que creó en la tabla y elija Editar el registro.
5. En la página Cambiar el rol, haga lo siguiente:

   - En Name (Nombre), indique un nombre nuevo para el rol en el campo de texto. Por ejemplo, **`Restricted`**.
   - En la lista Permisos, elimine `can read on DAGs` y `can edit on DAGs`, y, a continuación, agregue los permisos de lectura y escritura para el conjunto de DAG al que desee proporcionar acceso. Por ejemplo, para un DAG, `example_dag.py`, añada **`can read on DAG:`*`example_dag`*** y **`can edit on DAG:`*`example_dag`***.

   Seleccione Save. Ahora debería tener un nuevo rol que limite el acceso a un subconjunto de DAG disponibles en su entorno de Amazon MWAA. Ya puede asignar este rol a cualquier usuario existente de Apache Airflow.

Paso 3: asigne el rol que creó a su usuario de Amazon MWAA

Pasos para asignar el nuevo rol

1. Con las credenciales de acceso de `MWAAUser`, ejecute el siguiente comando de la CLI para recuperar la URL del servidor web del entorno.

   ```
   aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq
    '.Environment.WebserverUrl'
   ```

   Si todo va bien, obtendrá el siguiente resultado:

   ```
   "ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
   ```

2. Una vez que `MWAAUser` haya iniciado sesión en la Consola de administración de AWS, abra una nueva ventana del navegador y acceda a la URL siguiente. Modifique el `Webserver-URL` añadiendo su información.

   ```
   https://<Webserver-URL>/home
   ```

   Si todo va bien, verá una página de error de `Forbidden`. Eso es porque `MWAAUser` todavía no tiene permiso para acceder a la UI de Apache Airflow.

3. Una vez que `Admin` haya iniciado sesión en la Consola de administración de AWS, vuelva a abrir la consola de Amazon MWAA e inicie la interfaz de usuario de Apache Airflow de su entorno.

4. Desde el panel de la interfaz de usuario, expanda el menú desplegable Seguridad y, esta vez, seleccione Mostrar usuarios.

5. En la tabla de usuarios, busque el nuevo usuario de Apache Airflow y seleccione Editar el registro. El nombre del usuario coincidirá con su nombre de usuario de IAM según el patrón siguiente: `user/mwaa-user`.

6. En la página Editar el usuario, en la sección Rol, añada el nuevo rol personalizado que ha creado y, finalmente, proceda a Guardar.

   > **Note**
   >
   > El campo de apellido es obligatorio, pero se puede rellenar añadiendo solo un espacio.

La entidad principal `Public` de IAM concede el permiso `MWAAUser` para acceder a la UI de Apache Airflow, mientras que el nuevo rol proporciona los permisos adicionales necesarios para ver sus DAG.

> **Important**
>
> Cualquiera de los 5 roles predeterminados (por ejemplo, `Admin`) no autorizados por IAM y que se agreguen mediante la UI de Apache Airflow se eliminará la próxima vez que el usuario inicie sesión.

Pasos a seguir a continuación

- Consulte the section called “Acceso a un entorno de Amazon MWAA” para obtener más información sobre la gestión del acceso a su entorno Amazon MWAA y para ver ejemplos de políticas de IAM en JSON que puede usar para los usuarios de su entorno.

Recursos relacionados

- Control de acceso (documentación de Apache Airflow): obtenga más información sobre los roles predeterminados de Apache Airflow en el sitio web de documentación de Apache Airflow.

Tutorial: cómo automatizar la administración de sus propios puntos de conexión de entorno en Amazon MWAA

Si usa AWS Organizations para administrar varias Cuentas de AWS que comparten recursos, Amazon MWAA le permite crear y administrar sus propios puntos de conexión de Amazon VPC. Esto significa que puede usar políticas de seguridad más estrictas que le permitan acceder únicamente a los recursos que necesite su entorno.

Cuando crea un entorno en una Amazon VPC compartida, la cuenta propietaria de la Amazon VPC principal (propietaria) comparte las dos subredes privadas que requiere Amazon MWAA con otras cuentas (participantes) que pertenecen a la misma organización. A continuación, las cuentas

participantes que comparten esas subredes pueden ver, crear, modificar y eliminar entornos en la VPC compartida.

Cuando cree un entorno en una Amazon VPC compartida o de algún otro modo restringida por políticas, Amazon MWAA primero creará los recursos de la VPC del servicio y, a continuación, entrará en un estado `PENDING` por hasta 72 horas.

Cuando el estado del entorno cambia de `CREATING` a `PENDING`, Amazon MWAA envía una notificación de Amazon EventBridge sobre el cambio de estado. Esto le permite a la cuenta propietaria crear los puntos de conexión necesarios en nombre de los participantes, según la información del servicio de punto de conexión desde la consola o API de Amazon MWAA, o mediante programación. En el siguiente ejemplo, creamos nuevos puntos de conexión de Amazon VPC mediante una función de Lambda y una regla de EventBridge que escucha notificaciones de cambios de estado de Amazon MWAA.

Aquí creamos los nuevos puntos de conexión en la misma Amazon VPC que el entorno. Para configurar una Amazon VPC compartida, cree la regla de EventBridge y la función de Lambda en la cuenta propietaria, y el entorno de Amazon MWAA en la cuenta participante.

Temas

Requisitos previos

Para completar los pasos en este tutorial, deberá hacer lo siguiente:

- ...

Creación de la Amazon VPC

Utilice la siguiente plantilla de CloudFormation y el siguiente comando de la AWS CLI para crear una nueva Amazon VPC. La plantilla configura los recursos de Amazon VPC y modifica la política de punto de conexión para restringir el acceso a una cola específica.

1. Descargue la plantilla de CloudFormation y, a continuación, descomprima el archivo `.yml`.
2. En una nueva ventana de petición de comandos, navegue hasta la carpeta donde guardó la plantilla y, a continuación, use `create-stack` para crear la pila. El indicador `--template-body` especifica la ruta a la plantilla.

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

En la siguiente sección, creará la función de Lambda.

Crear la función de Lambda

Utilice el siguiente código Python y la política JSON de IAM para crear una nueva función de Lambda y un rol de ejecución. Esta función crea puntos de conexión de Amazon VPC para un servidor web Apache Airflow privado y una cola de Amazon SQS. Amazon MWAA utiliza Amazon SQS para poner en cola tareas con Celery entre varios procesos de trabajo a la hora de escalar su entorno.

1. Descargue el código de la función Python.
2. Descargue la política de permisos de IAM y, a continuación, descomprima el archivo.
3. Abra un comando de petición y, a continuación, navegue hasta la carpeta en la que guardó la política de permisos de JSON. Use el comando de IAM `create-role` para crear el nuevo rol.

```
aws iam create-role --role-name function-role \
      --assume-role-policy-document file://lambda-mwaa-vpce-policy.json
```

   Anote el ARN del rol de la respuesta AWS CLI. En el siguiente paso, especificamos este nuevo rol como el rol de ejecución de la función mediante su ARN.
4. Navegue hasta la carpeta en la que guardó el código de la función y, a continuación, utilice el comando `create-function` para crear una función nueva.

```
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 la función ARN de la respuesta AWS CLI. En el siguiente paso, especificamos el ARN para configurar la función como destino para una nueva regla de EventBridge.

En la siguiente sección, creará la regla de EventBridge que invoca esta función cuando el entorno entre en estado `PENDING`.

Crear la regla de EventBridge

Haga lo siguiente para crear una nueva regla que escuche las notificaciones de Amazon MWAA y se dirija a su nueva función de Lambda.

1. Use el comando `put-rule` de EventBridge para crear una nueva regla de EventBridge.

   ```
   aws events put-rule --name "mwaa-lambda-rule" \
   --event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment
    Status Change\"]}"
   ```

   El patrón de eventos escucha las notificaciones que Amazon MWAA envía cada vez que cambia el estado de un entorno.

   ```
   {
        "source": ["aws.airflow"],
        "detail-type": ["MWAA Environment Status Change"]
        }
   ```

2. Utilice el comando `put-targets` para agregar la función de Lambda como destino de la nueva regla.

   ```
   aws events put-targets --rule "mwaa-lambda-rule" \
   --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:mwaa-vpce-
   lambda"
   ```

Ya está listo para crear un nuevo entorno de Amazon MWAA con puntos de conexión de Amazon VPC administrados por el cliente.

Crear el entorno de Amazon MWAA

Utilice la consola de Amazon MWAA para crear un nuevo entorno con puntos de conexión de Amazon VPC administrados por el cliente.

1. Abra la consola de Amazon MWAA y seleccione Crear un entorno.
2. En Nombre, escriba un nombre único.

3. En Versión de Airflow, seleccione la versión más reciente.
4. Elija un bucket de Amazon S3 y una carpeta de DAG, como `dags/`, para utilizarlos con el entorno y, a continuación, seleccione Siguiente.
5. En la página Configurar los ajustes avanzados, haga lo siguiente:
   a. En Nube privada virtual, elija la Amazon VPC que creó en el paso anterior.
   b. En Webserver access (Acceso del servidor web), seleccione Public network (internet accessible) (Red pública (con acceso a Internet)).
   c. En Grupos de seguridad, seleccione el grupo de seguridad que creó con CloudFormation. Como los grupos de seguridad de los puntos de conexión AWS PrivateLink del paso anterior se autorreferencian, debe elegir el mismo grupo de seguridad para su entorno.
   d. En Administración de puntos de conexión, seleccione Puntos de conexión administrados por el cliente.
6. Conserve el resto de la configuración predeterminada y, a continuación, seleccione Siguiente.
7. Revise su selección y, a continuación, elija Crear entorno.

> **Tip**
>
> Para obtener más información sobre cómo configurar un nuevo entorno, consulte Introducción a Amazon MWAA.

Cuando el estado del entorno es `PENDING`, Amazon MWAA envía una notificación que coincide con el patrón de eventos que usted estableció para la regla. La regla invoca su función de Lambda. La función analiza el evento de notificación y obtiene la información de punto de conexión requerida para el servidor web y la cola de Amazon SQS. A continuación, crea los puntos de conexión en su Amazon VPC.

Cuando los puntos de conexión están disponibles, Amazon MWAA reanuda la creación del entorno. Una vez que haya finalizado, el estado del entorno cambiará a `AVAILABLE` y se podrá acceder al servidor web Apache Airflow mediante la consola de Amazon MWAA.

Códigos de ejemplo de Amazon Managed Workflows para Apache Airflow

Esta guía contiene código de ejemplos, incluidos los DAG y complementos personalizados, que puede utilizar en un entorno de Amazon Managed Workflows para Apache Airflow. Para ver más ejemplos del uso de Apache Airflow con servicios de AWS, consulte el directorio de `dags` en el repositorio GitHub de Apache Airflow.

Muestras

- Uso de un DAG para importar variables en la CLI
- Creación de una conexión SSH mediante SSHOperator
- Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow Snowflake
- Uso de un DAG para escribir métricas personalizadas en CloudWatch
- Limpieza de bases de datos de Aurora PostgreSQL en un entorno de Amazon MWAA
- Exportación de metadatos del entorno a archivos CSV en Amazon S3
- Uso de una clave secreta en AWS Secrets Manager para una variable de Apache Airflow
- Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow
- Creación de un complemento personalizado con Oracle
- Cómo cambiar la zona horaria de un DAG en Amazon MWAA
- Actualización de un token de CodeArtifact
- Creación de un complemento personalizado con Apache Hive y Hadoop
- Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator
- Invocación de DAG con una función de Lambda
- Invocación de DAG en diferentes entornos de Amazon MWAA
- Uso de Amazon MWAA con Amazon RDS para Microsoft SQL Server
- Uso de imágenes de Amazon ECR con Amazon EKS
- Conexión a Amazon ECS mediante el ECSOperator
- Uso de dbt con Amazon MWAA
- Blogs y tutoriales de AWS

Uso de un DAG para importar variables en la CLI

La siguiente muestra de código importa variables mediante la CLI de Amazon Managed Workflows para Apache Airflow.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Permisos

Su Cuenta de AWS necesita acceso a la política `AmazonMWAAAirflowCliAccess`. Consulte Política de la CLI de Apache Airflow: AmazonMWAAAirflowCliAccess para obtener más información.

Dependencias

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Código de ejemplo

En la siguiente muestra de código, se requieren tres entradas: el nombre de su entorno de Amazon MWAA (en `mwaa_env`), la Región de AWS de su entorno (en `aws_region`) y el archivo local que contiene las variables que desea importar (en `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)
```

Siguientes pasos

- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Creación de una conexión SSH mediante **`SSHOperator`**

El siguiente ejemplo describe cómo puede utilizar el `SSHOperator` en un gráfico acíclico dirigido (DAG) para conectarse a una instancia remota de Amazon EC2 desde su entorno Amazon Managed Workflows para Apache Airflow. Puede utilizar un enfoque similar para conectarse a cualquier instancia remota con acceso SSH.

En el siguiente ejemplo, se cargará una clave secreta SSH (`.pem`) en el directorio `dags` de su entorno en Amazon S3. A continuación, instale las dependencias necesarias mediante

`requirements.txt` y cree una nueva conexión de Apache Airflow en la interfaz de usuario. Por último, escribe un DAG que crea una conexión SSH con la instancia remota.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- Una clave secreta de SSH. En el código de muestra se supone que tiene una instancia de Amazon EC2 y una `.pem` en la misma región que su entorno de Amazon MWAA. Si no tiene una clave SSH, consulte Crear o importar un par de claves en la guía del usuario de Amazon EC2.

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Requisitos

Añada el siguiente parámetro a `requirements.txt` para instalar el paquete `apache-airflow-providers-ssh` en el servidor web. Una vez que su entorno se actualice y Amazon MWAA instale correctamente la dependencia, verá un nuevo tipo de conexión SSH en la UI.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
apache-airflow-providers-ssh
```

> **Note**
>
> `-c` define la URL de las restricciones en `requirements.txt`. Esto garantiza que Amazon MWAA instale la versión de paquete correcta para su entorno.

Cómo copiar su clave secreta en Amazon S3

Utilice el siguiente comando de la AWS Command Line Interface para copiar la clave `.pem` en el directorio `dags` de su entorno en Amazon S3.

```
aws s3 cp your-secret-key.pem s3://amzn-s3-demo-bucket/dags/
```

Amazon MWAA copia el contenido en `dags`, incluida la clave `.pem`, en el directorio local `/usr/local/airflow/dags/`. De este modo, Apache Airflow puede acceder a la clave.

Creación de una nueva conexión con Apache Airflow

Creación de una nueva conexión SSH mediante la interfaz de usuario de Apache Airflow

1. Abra la página Entornos en la consola de Amazon MWAA.
2. En la lista de entornos, elija Abrir la interfaz de usuario de Airflow para su entorno.
3. En la página de UI de Apache Airflow, seleccione Admin (Administrador) en la barra de navegación superior para ampliar la lista desplegable y, a continuación, seleccione Connections (Conexiones).
4. En la página Listar conexiones, seleccione + o el botón Añadir un nuevo registro para añadir una nueva conexión.
5. En la página Conectar a AD, proporcione la siguiente información:
   a. En Nombre de conexión introduzca **`ssh_new`**.
   b. Para el tipo de conexión, seleccione SSH en la lista desplegable.

> **Note**
>
> Si el tipo de conexión SSH no está disponible en la lista, Amazon MWAA no ha instalado el paquete `apache-airflow-providers-ssh` necesario. Actualice el archivo `requirements.txt` para incluir este paquete e inténtelo de nuevo.

c. Para Host, introduzca la dirección IP de la instancia de Amazon EC2 a la que desee conectarse. Por ejemplo, **`12.345.67.89`**.

d. En Nombre de usuario, introduzca **`ec2-user`** si se está conectando a una instancia de Amazon EC2. Su nombre de usuario puede ser diferente, según el tipo de instancia remota a la que desee que se conecte Apache Airflow.

e. Para Extra, introduzca el siguiente par clave-valor en formato JSON:

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

Este par clave-valor indica a Apache Airflow que busque la clave secreta en el directorio local `/dags`.

Código de ejemplo

El siguiente DAG utiliza `SSHOperator` para conectarse a la instancia de Amazon EC2 de destino y, a continuación, ejecuta el comando de Linux `hostname` para imprimir el nombre de la instancia. Puede modificar el DAG para ejecutar cualquier comando o script en la instancia remota.

1. Abra una terminal y navegue hasta el directorio en el que está almacenado el código del DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si se ejecuta correctamente, verá un resultado similar al siguiente en los registros de tareas del `ssh_task` en el 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
```

Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow Snowflake

En el siguiente ejemplo, se llama a AWS Secrets Manager para obtener una clave secreta para una conexión de Apache Airflow Snowflake en Amazon Managed Workflows para Apache Airflow. Se asume que ha realizado los pasos que se detallan en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- El backend de Secrets Manager como opción de configuración de Apache Airflow, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.
- Una cadena de conexión de Apache Airflow en Secrets Manager, como aparece en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Permisos

- Permisos de Secrets Manager, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Requisitos

Para usar el código de ejemplo de esta página, agregue las siguientes dependencias a su `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
apache-airflow-providers-snowflake==1.3.0
```

Código de ejemplo

En los siguientes pasos se describe cómo crear el código DAG que llama a Secrets Manager para recibir el secreto.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

   ```
   cd dags
   ```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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",
    )
```

Siguientes pasos

- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Uso de un DAG para escribir métricas personalizadas en CloudWatch

Puede usar el siguiente código de ejemplo para escribir un gráfico acíclico dirigido (DAG) que ejecute un `PythonOperator` para recuperar métricas a nivel de sistema operativo para un entorno de Amazon MWAA. A continuación, el DAG publica los datos como métricas personalizadas en Amazon CloudWatch.

Las métricas personalizadas a nivel del sistema operativo proporcionan una visibilidad adicional sobre la forma en que los procesos de trabajo de su entorno utilizan recursos como la memoria virtual y la CPU. Puede utilizar esta información para seleccionar la clase de entorno que mejor se adapte a su carga de trabajo.

Temas

- Versión
- Requisitos previos
- Permisos
- Dependencias

- Ejemplo de código

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el ejemplo de código en esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Dependencias

- No se necesitan dependencias adicionales para usar el código de ejemplo de esta página.

Ejemplo de código

1. En el símbolo del sistema, vaya hasta la carpeta en la que se encuentra almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del siguiente código de ejemplo y guárdelo localmente como `dag-custom-metrics.py`. Sustituya `MWAA-ENV-NAME` por el nombre de su entorno.

```
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. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si el DAG se ejecuta correctamente, debería aparecer algo similar a lo siguiente en los registros de 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
```

Limpieza de bases de datos de Aurora PostgreSQL en un entorno de Amazon MWAA

Amazon Managed Workflows para Apache Airflow utiliza una base de datos Aurora PostgreSQL como base de datos de metadatos de Apache Airflow, donde se ejecuta el DAG y se almacenan las instancias de tareas. La siguiente muestra de código borra periódicamente las entradas de la base de datos Aurora PostgreSQL dedicada a su entorno Amazon MWAA.

Temas

Versión

Los ejemplos de código de esta página son específicos de Apache Airflow v2, compatible con Amazon MWAA. Consulte las versiones de Apache Airflow compatibles.

> **Tip**
>
> Para los usuarios de Apache Airflow v3: si desea limpiar una base de datos (purgar los registros antiguos de las tablas del almacén de metadatos), ejecute el comando `db clean` en la CLI.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Dependencias

Para usar este código de ejemplo con Apache Airflow v2, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Código de ejemplo

El siguiente DAG limpia la base de datos de metadatos de las tablas especificadas en `TABLES_TO_CLEAN`. En el ejemplo, se eliminan los datos de las tablas especificadas con más de 30 días de antigüedad. Para ajustar el tiempo de eliminación de las entradas, establezca `MAX_AGE_IN_DAYS` en un valor diferente.

Apache Airflow v2.4 to 2.10.3

```
from airflow import DAG
from airflow.models.param import Param
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

from datetime import datetime, timedelta
```

```
# Note: Database commands might time out if running longer than 5 minutes. If this
 occurs, please increase the MAX_AGE_IN_DAYS (or change
# timestamp parameter to an earlier date) for initial runs, then reduce on
 subsequent runs until the desired retention is met.

MAX_AGE_IN_DAYS = 30

# To clean specific tables, please provide a comma-separated list per
# https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-
ref.html#clean
# A value of None will clean all tables

TABLES_TO_CLEAN = None

with DAG(
    dag_id="clean_db_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1),
    params={
        "timestamp": Param(
            default=(datetime.now()-timedelta(days=MAX_AGE_IN_DAYS)).strftime("%Y-
%m-%d %H:%M:%S"),
            type="string",
            minLength=1,
            maxLength=255,
        ),
    }
) as dag:
    if TABLES_TO_CLEAN:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --tables '"+TABLES_TO_CLEAN+"' --skip-archive --yes"
    else:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --skip-archive --yes"

    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=bash_command
    )
```

Apache Airflow v2.2 and earlier

```
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 30 days. Adjust MAX_AGE_IN_DAYS to set how far back
 this DAG cleans the database.
MAX_AGE_IN_DAYS = 30
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] >= earliest_date).filter(x[1] <=
 oldest_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()
```

Exportación de metadatos del entorno a archivos CSV en Amazon S3

Use el siguiente código de muestra para crear un gráfico acíclico dirigido (DAG) que consulte en la base de datos un rango de información de ejecución del DAG y escriba los datos en los archivos `.csv` almacenados en Amazon S3.

Es posible que deba exportar información de la base de datos Aurora PostgreSQL de su entorno para inspeccionar los datos localmente, archivarlos en un almacenamiento de objetos o combinarlos con herramientas como el operador Amazon S3 a Amazon Redshift y la limpieza de la base de datos, a fin de sacar los metadatos de Amazon MWAA del entorno y conservarlos para futuros análisis.

Puede consultar en la base de datos cualquiera de los objetos enumerados en los modelos de Apache Airflow. En este código de muestra, se usan tres modelos, `DagRun`, `TaskFail` y `TaskInstance`, que proporcionan información relevante para las ejecuciones del DAG.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- Un nuevo bucket de Amazon S3 a donde desea exportar su información de metadatos.

Permisos

Amazon MWAA necesita permiso para que la acción `s3:PutObject` de Amazon S3 escriba la información de metadatos consultada en Amazon S3. Añada la siguiente instrucción de política al rol de ejecución de su entorno.

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
}
```

Esta política limita el acceso de escritura únicamente a *amzn-s3-demo-bucket*.

Requisitos

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Código de ejemplo

Los siguientes pasos describen cómo puede crear un DAG que consulte Aurora PostgreSQL y escriba el resultado en su nuevo bucket de Amazon S3.

1. En su terminal, vaya hasta el directorio en el que está almacenado el código de DAG. Por ejemplo:

   ```
   cd dags
   ```

2. Copie el contenido del siguiente código de ejemplo y guárdelo localmente como `metadata_to_csv.py`. Puede cambiar el valor asignado `MAX_AGE_IN_DAYS` para controlar la antigüedad de los registros más antiguos que el DAG consulta en la base de datos de metadatos.

   ```
   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. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si se ejecuta correctamente, verá un resultado similar al siguiente en los registros de tareas para la tarea `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
```

Ahora puede acceder a los archivos `.csv` exportados y descargarlos en su nuevo bucket de Amazon S3 en `/files/export/`.

Uso de una clave secreta en AWS Secrets Manager para una variable de Apache Airflow

El siguiente ejemplo, llama a AWS Secrets Manager para obtener una clave secreta para una variable de Apache Airflow en Amazon Managed Workflows para Apache Airflow. Se asume que ha realizado los pasos que se detallan en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- El backend de Secrets Manager como opción de configuración de Apache Airflow, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.
- Una cadena de variables de Apache Airflow en Secrets Manager, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Permisos

- Permisos de Secrets Manager, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Requisitos

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Código de ejemplo

En los siguientes pasos se describe cómo crear el código DAG que llama a Secrets Manager para recibir el secreto.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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
    )
```

Siguientes pasos

- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow

El siguiente ejemplo, se llama a AWS Secrets Manager para obtener una clave secreta para una conexión de Apache Airflow en Amazon Managed Workflows para Apache Airflow. Se asume que ha realizado los pasos que se detallan en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- El backend de Secrets Manager como opción de configuración de Apache Airflow, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.
- Una cadena de conexión de Apache Airflow en Secrets Manager, como aparece en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Permisos

- Permisos de Secrets Manager, como se muestra en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager.

Requisitos

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Código de ejemplo

En los siguientes pasos se describe cómo crear el código DAG que llama a Secrets Manager para recibir el secreto.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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(region_name='us-east-1')
    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
    )
```

Siguientes pasos

- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.

Creación de un complemento personalizado con Oracle

En el siguiente ejemplo, se explican los pasos necesarios para crear un complemento personalizado con Oracle para Amazon MWAA y se puede combinar con otros complementos y binarios personalizados en el archivo plugins.zip.

Contenido

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- El registro de procesos de trabajo habilitado en cualquier nivel de registro, `CRITICAL` o en la sección anterior a su entorno. Para obtener más información sobre los tipos de registros de Amazon MWAA y sobre cómo administrar sus grupos de registros, consulte the section called "Visualización de registros de Airflow"

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Requisitos

Para usar el código de ejemplo de esta página, agregue las siguientes dependencias a su `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

Código de ejemplo

En los siguientes pasos se explica cómo crear el código DAG que probará el complemento personalizado.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

   ```
   cd dags
   ```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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
    )
```

Creación del complemento personalizado

En esta sección se describe cómo descargar las dependencias, crear el complemento personalizado y el archivo plugins.zip.

Descarga de dependencias

Amazon MWAA extraerá el contenido del archivo plugins.zip en `/usr/local/airflow/plugins` en cada contenedor de procesos de trabajo y programador de Amazon MWAA. Se utiliza para añadir binarios a su entorno. En los siguientes pasos se describe cómo ensamblar los archivos necesarios para el complemento personalizado.

Extracción de la imagen del contenedor Linux de Amazon

1. En el símbolo del sistema, extraiga la imagen del contenedor Linux de Amazon y ejecútelo localmente. Por ejemplo:

```
docker pull amazonlinux
      docker run -it amazonlinux:latest /bin/bash
```

El símbolo del sistema puede invocar un símbolo del sistema bash. Por ejemplo:

```
bash-4.2#
```

2. Instale la característica de E/S asíncrona nativa de Linux (libaio).

```
yum -y install libaio
```

3. Mantenga esta ventana abierta para los pasos siguientes. Copiaremos los siguientes archivos en el sistema local: `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1`.

Descarga de la carpeta del cliente

1. Instale el paquete de descompresión localmente. Por ejemplo:

```
sudo yum install unzip
```

2. Cree un directorio de `oracle_plugin`. Por ejemplo:

```
mkdir oracle_plugin
cd oracle_plugin
```

3. Utilice el siguiente comando curl para descargar el archivo instantclient-basic-linux.x64-18.5.0.0.0dbru.zip de Oracle Instant Client Downloads para Linux x86-64 (64 bits).

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. Descomprima el archivo `client.zip`. Por ejemplo:

```
unzip *.zip
```

Extracción de archivos de Docker

1. En un nuevo símbolo del sistema, muestre y anote su ID de contenedor de Docker. Por ejemplo:

   ```
   docker container ls
   ```

   El símbolo del sistema debería mostrar todos los contenedores y sus ID. Por ejemplo:

   ```
   debc16fd6970
   ```

2. En su directorio `oracle_plugin`, extraiga los archivos `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1` en la carpeta `instantclient_18_5` local. Por ejemplo:

   ```
   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/
   ```

Complemento personalizado

Apache Airflow ejecutará el contenido de los archivos de Python en la carpeta de complementos durante el arranque. Esto se usa para establecer y modificar variables de entorno. En los siguientes pasos se describe el código de muestra del complemento personalizado.

- Copie el contenido del código de ejemplo siguiente y guárdelo 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

En los siguientes pasos, se muestra cómo crear el `plugins.zip`. El contenido de este ejemplo se puede combinar con sus otros complementos y binarios en un solo archivo `plugins.zip`.

Compresión del contenido del directorio de complementos

1. En una línea de comando, vaya al directorio `oracle_plugin`. Por ejemplo:

```
cd oracle_plugin
```

2. Comprima el directorio `instantclient_18_5` en plugins.zip. Por ejemplo:

```
zip -r ../plugins.zip ./
```

   La línea de comandos muestra lo siguiente:

```
oracle_plugin$ ls
client.zip  instantclient_18_5
```

3. Elimine el archivo `client.zip`. Por ejemplo:

```
rm client.zip
```

Comprima el archivo env_var_plugin_oracle.py

1. Agregue el archivo `env_var_plugin_oracle.py` a la raíz de plugins.zip. Por ejemplo:

```
zip plugins.zip env_var_plugin_oracle.py
```

2. Ahora su archivo plugins.zip debería incluir lo siguiente:

```
env_var_plugin_oracle.py
instantclient_18_5/
```

Opciones de configuración de Airflow

Si utiliza Apache Airflow v2, agregue `core.lazy_load_plugins : False` como opción de configuración de Apache Airflow. Para obtener más información, consulte Uso de las opciones de configuración para cargar complementos en la versión 2.

Siguientes pasos

- Aprenda a cargar el archivo `requirements.txt` de este ejemplo a su bucket de Amazon S3 en Instalación de dependencias de Python.
- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.
- Obtenga más información sobre cómo cargar el archivo `plugins.zip` de este ejemplo a su bucket de Amazon S3 en Instalación de complementos personalizados.

Cómo cambiar la zona horaria de un DAG en Amazon MWAA

Apache Airflow programa su gráfico acíclico dirigido (DAG) en UTC+0 de forma predeterminada. En los siguientes pasos, se muestra cómo puede cambiar la zona horaria en la que Amazon MWAA ejecuta sus DAG con Pendulum. Opcionalmente, en este tema se muestra cómo puede crear un complemento personalizado para cambiar la zona horaria de los registros de Apache Airflow de su entorno.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Creación de un complemento para cambiar la zona horaria en los registros de Airflow

Apache Airflow ejecutará los archivos de Python en el directorio de `plugins` al inicio. Con el siguiente complemento, puede anular la zona horaria del ejecutor, lo que modifica la zona horaria en la que Apache Airflow escribe los registros.

1. Cree un directorio llamado `plugins` para su complemento personalizado y vaya al directorio. Por ejemplo:

```
$ mkdir plugins
$ cd plugins
```

2. Copie el contenido de la siguiente muestra de código y guárdelo localmente como `dag-timezone-plugin.py` en la carpeta `plugins`.

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. En el directorio `plugins`, cree un archivo de Python vacío llamado `__init__.py`. Su directorio `plugins` debería ser similar a lo siguiente:

```
plugins/
 |-- __init__.py
 |-- dag-timezone-plugin.py
```

Creación de un `plugins.zip`

En los siguientes pasos, se explica cómo crear `plugins.zip`. El contenido de esta muestra se puede combinar con otros complementos y binarios en un solo archivo `plugins.zip`.

1. En el símbolo del sistema, vaya hasta el directorio `plugins` del paso anterior. Por ejemplo:

```
cd plugins
```

2. Comprima el contenido en su directorio `plugins`.

```
zip -r ../plugins.zip ./
```

3. Carga de `plugins.zip` en el bucket S3

```
aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

Código de ejemplo

Para cambiar la zona horaria predeterminada (UTC+0) en la que se ejecuta el DAG, utilizaremos una biblioteca llamada Pendulum, una biblioteca de Python para trabajar con una datetime y conocer la zona horaria.

1. En el símbolo del sistema, vaya hasta el directorio en el que están almacenados los DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido de la siguiente muestra y guárdelo 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. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si se ejecuta correctamente, obtendrá un resultado similar al siguiente en los registros para `tz_aware_task` en el `tz_test` DAG:

```
[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
```

Siguientes pasos

- Obtenga más información sobre cómo cargar el archivo `plugins.zip` de este ejemplo a su bucket de Amazon S3 en Instalación de complementos personalizados.

Actualización de un token de CodeArtifact

Si utiliza CodeArtifact para instalar dependencias de Python, Amazon MWAA requiere un token activo. Para permitir que Amazon MWAA acceda a un repositorio de CodeArtifact en tiempo de ejecución, puede usar un script de inicio y configurar `PIP_EXTRA_INDEX_URL` con el token.

En el siguiente tema se describe cómo crear un script de inicio que utilice la operación de la API de CodeArtifact `get_authorization_token` para recuperar un token nuevo cada vez que el entorno se inicie o se actualice.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- Un repositorio de CodeArtifact en el que almacenar las dependencias de su entorno.

Permisos

Para actualizar el token CodeArtifact y escribir el resultado en Amazon S3, Amazon MWAA debe tener los siguientes permisos en el rol de ejecución.

- La acción `codeartifact:GetAuthorizationToken` permite a Amazon MWAA recuperar un nuevo token de CodeArtifact. La siguiente política otorga permisos a todos los dominios de CodeArtifact que cree. Puede restringir aún más el acceso a sus dominios modificando el valor del recurso en la instrucción y especificando solo los dominios a los que desee que acceda su entorno.

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- La acción `sts:GetServiceBearerToken` es necesaria para llamar a la operación de la API `GetAuthorizationToken` de CodeArtifact. Esta operación devuelve el token que debe usarse cuando se usa un administrador de paquetes como `pip` con CodeArtifact. Para usar un administrador de paquetes con un repositorio de CodeArtifact, el rol de ejecución de su entorno debe permitir `sts:GetServiceBearerToken`, tal y como se muestra en la siguiente instrucción de política.

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

Código de ejemplo

Los siguientes pasos describen cómo crear un script de inicio que actualice el token CodeArtifact.

1. Copie el contenido del código de ejemplo siguiente y guárdelo localmente como `code_artifact_startup_script.sh`.

```
#!/bin/sh
```

```
# Startup script for MWAA, refer to 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. Vaya a la carpeta en la que guardó el script. Utilice `cp` en una nueva ventana de símbolo del sistema para cargar el script en su bucket. Sustituya *amzn-s3-demo-bucket* por su información.

```
aws s3 cp code_artifact_startup_script.sh s3://amzn-s3-demo-bucket/
code_artifact_startup_script.sh
```

Si se ejecuta correctamente, Amazon S3 muestra la ruta URL del objeto:

```
upload: ./code_artifact_startup_script.sh to s3://amzn-s3-demo-bucket/
code_artifact_startup_script.sh
```

Tras cargar el script, su entorno actualiza y ejecuta el script al iniciarse.

Siguientes pasos

- Aprenda a usar los scripts de inicio para personalizar su entorno en the section called “Uso de un script de inicio”.

- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.
- Obtenga más información sobre cómo cargar el archivo `plugins.zip` de este ejemplo a su bucket de Amazon S3 en Instalación de complementos personalizados.

Creación de un complemento personalizado con Apache Hive y Hadoop

Amazon MWAA extrae el contenido de un `plugins.zip` a `/usr/local/airflow/plugins`. Esto se puede utilizar para añadir archivos binarios a sus contenedores. Además, Apache Airflow ejecuta el contenido de los archivos de Python de la carpeta `plugins` en el inicio, lo que permite establecer y modificar las variables de entorno. En el siguiente ejemplo se explican los pasos necesarios para crear un complemento personalizado con Apache Hive y Hadoop en un entorno de Amazon Managed Workflows para Apache Airflow. Además, se puede combinar con otros complementos y binarios personalizados.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Requisitos

Para usar el código de ejemplo de esta página, agregue las siguientes dependencias a su `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Descarga de dependencias

Amazon MWAA extraerá el contenido del archivo plugins.zip en `/usr/local/airflow/plugins` en cada contenedor de procesos de trabajo y programador de Amazon MWAA. Se utiliza para añadir binarios a su entorno. En los siguientes pasos se describe cómo ensamblar los archivos necesarios para el complemento personalizado.

1. En el símbolo del sistema, vaya hasta el directorio en el que desee crear el complemento. Por ejemplo:

   ```
   cd plugins
   ```

2. Descargue Hadoop desde una réplica, por ejemplo:

   ```
   wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
   ```

3. Descargue Hive desde una replica, por ejemplo:

   ```
   wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
   ```

4. Cree un directorio. Por ejemplo:

```
mkdir hive_plugin
```

5. Extraiga Hadoop.

```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. Extraiga Hive.

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

Complemento personalizado

Apache Airflow ejecutará el contenido de los archivos de Python en la carpeta de complementos durante el arranque. Esto se usa para establecer y modificar variables de entorno. En los siguientes pasos se describe el código de muestra del complemento personalizado.

1. En una línea de comando, vaya al directorio `hive_plugin`. Por ejemplo:

```
cd hive_plugin
```

2. Copie el contenido del siguiente ejemplo de código y guárdelo localmente como `hive_plugin.py` en el directorio `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 el contenido del siguiente texto y guárdelo localmente como `.airflowignore` en el directorio `hive_plugin`.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

En los siguientes pasos, se explica cómo crear `plugins.zip`. El contenido de este ejemplo se puede combinar con otros complementos y archivos binarios en un solo archivo `plugins.zip`.

1. En el símbolo del sistema, vaya hasta el directorio `hive_plugin` del paso anterior. Por ejemplo:

```
cd hive_plugin
```

2. Comprima el contenido de la carpeta `plugins`.

```
zip -r ../hive_plugin.zip ./
```

Código de ejemplo

En los siguientes pasos se explica cómo crear el código DAG que probará el complemento personalizado.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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'
    )
```

Opciones de configuración de Airflow

Si utiliza Apache Airflow v2, agregue `core.lazy_load_plugins : False` como opción de configuración de Apache Airflow. Para obtener más información, consulte Uso de las opciones de configuración para cargar complementos en la versión 2.

Siguientes pasos

- Aprenda a cargar el archivo `requirements.txt` de este ejemplo a su bucket de Amazon S3 en Instalación de dependencias de Python.
- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.
- Obtenga más información sobre cómo cargar el archivo `plugins.zip` de este ejemplo a su bucket de Amazon S3 en Instalación de complementos personalizados.

Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator

En el siguiente ejemplo, se muestra cómo parchear `PythonVirtualenvOperator` de Apache Airflow con un complemento personalizado en Amazon Managed Workflows para Apache Airflow.

Temas

- Siguientes pasos

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Permisos

No se necesitan permisos adicionales para usar el código de ejemplo de esta página.

Requisitos

Para usar el código de ejemplo de esta página, agregue las siguientes dependencias a su `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
virtualenv
```

Código de muestra de complemento personalizado

Apache Airflow ejecutará el contenido de los archivos de Python en la carpeta de complementos durante el arranque. Este complemento parcheará la versión integrada `PythonVirtualenvOperator` durante el proceso de arranque para que sea compatible con Amazon MWAA. En los siguientes pasos, se describe el código de muestra del complemento personalizado.

1. En el símbolo del sistema, vaya hasta el directorio `plugins` del paso anterior. Por ejemplo:

   ```
   cd plugins
   ```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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'
```

Plugins.zip

En los siguientes pasos, se muestra cómo crear el `plugins.zip`.

1. En el símbolo del sistema, vaya hasta el directorio que contiene `virtual_python_plugin.py` del paso anterior. Por ejemplo:

```
cd plugins
```

2. Comprima el contenido de la carpeta `plugins`.

```
zip plugins.zip virtual_python_plugin.py
```

Código de ejemplo

Los siguientes pasos describen cómo crear el código de DAG para el complemento personalizado.

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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,
    )
```

Opciones de configuración de Airflow

Si utiliza Apache Airflow v2, agregue `core.lazy_load_plugins : False` como opción de configuración de Apache Airflow. Para obtener más información, consulte Uso de las opciones de configuración para cargar complementos en la versión 2.

Siguientes pasos

- Aprenda a cargar el archivo `requirements.txt` de este ejemplo a su bucket de Amazon S3 en Instalación de dependencias de Python.
- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.
- Obtenga más información sobre cómo cargar el archivo `plugins.zip` de este ejemplo a su bucket de Amazon S3 en Instalación de complementos personalizados.

Invocación de DAG con una función de Lambda

El siguiente código de ejemplo utiliza una función AWS Lambda para obtener un token CLI de Apache Airflow e invocar un gráfico acíclico dirigido (DAG) en un entorno de Amazon MWAA.

Temas

- Versión

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y con Apache Airflow v3 en Python 3.11.

Requisitos previos

Para utilizar este ejemplo de código, debe:

- Utilizar el modo de acceso a la red pública para su entorno Amazon MWAA.
- Tener una función de Lambda utilizando el último tiempo de ejecución de Python.

> **Note**
>
> Si la función de Lambda y su entorno Amazon MWAA están en la misma VPC, puede usar este código en una red privada. Para esta configuración, el rol de ejecución de la función de Lambda necesita permiso para llamar a la operación de la API de CreateNetworkInterface de Amazon Elastic Compute Cloud (Amazon EC2). Puede proporcionar este permiso mediante la política `AWSLambdaVPCAccessExecutionRole` administrada por AWS.

Permisos

Para usar el ejemplo de código de esta página, el rol de ejecución de su entorno Amazon MWAA necesita acceso para realizar la acción `airflow:CreateCliToken`. Puede proporcionar este permiso mediante la política `AmazonMWAAAirflowCliAccess` administrada por AWS.

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Para obtener más información, consulta Política de la CLI de Apache Airflow: AmazonMWAAAirflowCliAccess.

Dependencias

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Ejemplo de código

1. Abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija su función de Lambda en la lista de funciones.
3. En la página de funciones, copie el código siguiente y sustituya lo siguiente por los nombres de sus recursos:

   - YOUR_ENVIRONMENT_NAME: el nombre del entorno de Amazon MWAA.
   - YOUR_DAG_NAME: el nombre del DAG que desea 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. Elija Implementar.
5. Elija Probar para invocar la función mediante la consola Lambda.
6. Para comprobar que su Lambda ha invocado correctamente su DAG, utilice la consola Amazon MWAA para navegar hasta la interfaz de usuario de Apache Airflow de su entorno y, a continuación, haga lo siguiente:

   a. En la página DAG, busque su nuevo DAG de destino en la lista de DAG.

   b. En Última ejecución, compruebe la marca de tiempo de la última ejecución del DAG. Esta marca de tiempo debe acercarse lo máximo posible a la última marca de tiempo para `invoke_dag` en su otro entorno.

   c. En Tareas recientes, compruebe que la última ejecución se haya realizado correctamente.

Invocación de DAG en diferentes entornos de Amazon MWAA

El siguiente código de ejemplo crea un token CLI de Apache Airflow. A continuación, el código utiliza un gráfico acíclico dirigido (DAG) en un entorno de Amazon MWAA para invocar un DAG en un entorno distinto de Amazon MWAA.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y con Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el ejemplo de código en esta página, necesitará lo siguiente:

- Dos entornos de Amazon MWAA con acceso a un servidor web de red pública, incluido su entorno actual.
- Un DAG de muestra cargado en el bucket de Amazon Simple Storage Service (Amazon S3) de su entorno de destino.

Permisos

Para usar el código de ejemplo de esta página, el rol de ejecución de su entorno debe tener permiso para crear un token CLI de Apache Airflow. Puede adjuntar la política administrada de AWS `AmazonMWAAAirflowCliAccess` para conceder este permiso.

JSON

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

```
}
```

Para obtener más información, consulta Política de la CLI de Apache Airflow: AmazonMWAAAirflowCliAccess.

Dependencias

Para usar este código de ejemplo con Apache Airflow v2 y versiones posteriores, no se necesitan dependencias adicionales. Use aws-mwaa-docker-images para instalar Apache Airflow.

Ejemplo de código

En el siguiente código de ejemplo se supone que está utilizando un DAG en su entorno actual para invocar un DAG en otro entorno.

1. En su terminal, vaya hasta el directorio en el que está almacenado el código de DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del siguiente código de ejemplo y guárdelo localmente como `invoke_dag.py`. Reemplace los valores siguientes por sus propios valores.

   - `your-new-environment-name`: nombre del otro entorno en el que desea invocar el DAG.
   - `your-target-dag-id`: ID del DAG del otro entorno que desea 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. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si el DAG se ejecuta correctamente, verá un resultado similar al siguiente en los registros de tareas 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 comprobar que el DAG se haya invocado correctamente, vaya a la interfaz de usuario de Apache Airflow del nuevo entorno y, a continuación, haga lo siguiente:

a. En la página DAG, busque su nuevo DAG de destino en la lista de DAG.

b. En Última ejecución, compruebe la marca de tiempo de la última ejecución del DAG. Esta marca de tiempo debe acercarse lo máximo posible a la última marca de tiempo para `invoke_dag` en su otro entorno.

c. En Tareas recientes, compruebe que la última ejecución se haya realizado correctamente.

Uso de Amazon MWAA con Amazon RDS para Microsoft SQL Server

Puede utilizar Amazon Managed Workflows para Apache Airflow para conectarse a un RDS para SQL Server. El siguiente código de ejemplo utiliza los DAG de un entorno Amazon Managed Workflows para Apache Airflow para conectarse a un servidor Amazon RDS para Microsoft SQL Server y ejecutar consultas en él.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- Amazon MWAA y el RDS para SQL Server deben ejecutarse en la misma Amazon VPC.
- Los grupos de seguridad de VPC de Amazon MWAA y el servidor deben configurarse con las siguientes conexiones:
  - Una regla de entrada para el puerto `1433` abierto para Amazon RDS en el grupo de seguridad de Amazon MWAA
  - O una regla de salida para el puerto `1433` abierto de Amazon MWAA a RDS

- Apache Airflow Connection para RDS para SQL Server refleja el nombre de host, el puerto, el nombre de usuario y la contraseña de la base de datos del servidor SQL de Amazon RDS creada en el proceso anterior.

Dependencias

Para usar el código de ejemplo de esta sección, agregue la siguiente dependencia a su `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
apache-airflow-providers-microsoft-mssql==1.0.1
   apache-airflow-providers-odbc==1.0.1
   pymssql==2.2.1
```

Conexión Apache Airflow v2

Si utiliza una conexión en Apache Airflow v2, asegúrese de que el objeto de conexión Airflow incluya los siguientes pares clave-valor:

1. ID de conexión: mssql_default
2. Tipo de conexión: Amazon Web Services
3. Host: `YOUR_DB_HOST`
4. Esquema:
5. Inicio de sesión: admin
6. Contraseña:
7. Puerto: 1433
8. Extra:

Código de ejemplo

1. En el símbolo del sistema, vaya hasta el directorio en el que esté almacenado el código DAG. Por ejemplo:

   ```
   cd dags
   ```

2. Copie el contenido del código de ejemplo siguiente y guárdelo 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
```

Siguientes pasos

- Aprenda a cargar el archivo `requirements.txt` de este ejemplo a su bucket de Amazon S3 en Instalación de dependencias de Python.
- Aprenda a cargar el código el DAG de este ejemplo en la carpeta `dags` de su bucket de Amazon S3 en Cómo añadir o actualizar DAG.
- Explore ejemplos de scripts y otros ejemplos de módulos pymssql.
- Obtenga más información sobre la ejecución de código SQL en una base de datos Microsoft SQL específica mediante mssql_operator en la guía de referencia de Apache Airflow.

Uso de imágenes de Amazon ECR con Amazon EKS

En el siguiente ejemplo se muestra cómo usar Amazon Managed Workflows para Apache Airflow con Amazon EKS.

Temas

- Versión
- Requisitos previos
- Creación de una clave pública para Amazon EC2
- Creación del clúster
- Creación de un espacio de nombres de mwaa
- Creación de un rol para el espacio de nombres de mwaa
- Creación del rol de IAM del clúster de Amazon EKS
- Creación del archivo requirements.txt
- Creación de un mapeo de identidad para Amazon EKS
- Creación del kubeconfig

- Creación de un DAG
- Adición del DAG y kube_config.yaml al bucket de Amazon S3
- Habilitación y activación del ejemplo

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y con Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el ejemplo de este tema, necesitará lo siguiente:

- Un entorno de Amazon MWAA.
- eksctl. Para obtener más información, consulte cómo instalar eksctl.
- kubectl. Para obtener más información, consulte cómo instalar y configurar kubectl. En algunos casos, se instala con eksctl.
- Un par de claves de EC2 en la región donde se crea su entorno de Amazon MWAA. Para obtener más información, consulte cómo crear o importar un par de claves.

> **Note**
>
> Cuando utilice un comando `eksctl`, puede incluir un `--profile` para especificar un perfil distinto del predeterminado.

Creación de una clave pública para Amazon EC2

Utilice el siguiente comando para crear una clave pública a partir de su clave privada.

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

Para obtener más información, consulte cómo recuperar la clave pública de su par de claves.

Creación del clúster

Utilice el siguiente comando para crear el clúster. Si desea usar un nombre personalizado para el clúster o crearlo en una región diferente, sustituya los valores del nombre y la región. Debe crear el clúster en la misma región en la que haya creado el entorno de Amazon MWAA. Sustituya los valores de las subredes para que coincidan con las subredes de la red de Amazon VPC que utilice para Amazon MWAA. Sustituya el valor por `ssh-public-key` para que coincida con la clave que utilice. Puede utilizar una clave existente de Amazon EC2 que se encuentre en la misma región o crear una nueva en la misma región donde creó su entorno de 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"
```

La creación del clúster tarde un tiempo en completarse. Una vez que termine el proceso, puede comprobar que el clúster se haya creado correctamente y que tiene el proveedor OIDC de IAM configurado mediante el siguiente comando:

```
eksctl utils associate-iam-oidc-provider \
--region us-west-2 \
--cluster mwaa-eks \
--approve
```

Creación de un espacio de nombres de **mwaa**

Tras confirmar que el clúster se ha creado correctamente, utilice el siguiente comando para crear un espacio de nombres para los pods.

```
kubectl create namespace mwaa
```

Creación de un rol para el espacio de nombres de **mwaa**

Tras crear el espacio de nombres, cree un rol y un enlace de rol para un usuario de Amazon MWAA en EKS que pueda ejecutar pods en un espacio de nombres de MWAA. Si utilizó un nombre diferente para el espacio de nombres, reemplace mwaa en `-n` *mwaa* por el nombre que usó.

```
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 que el nuevo rol puede acceder al clúster de Amazon EKS ejecutando el siguiente comando. Asegúrese de utilizar el nombre correcto si no usó *mwaa*:

```
kubectl get pods -n mwaa --as mwaa-service
```

Deberá aparecer un mensaje que indique lo siguiente:

```
No resources found in mwaa namespace.
```

Creación del rol de IAM del clúster de Amazon EKS

Debe crear un rol de IAM y, a continuación, vincularlo al clúster de Amazon EKS (k8s) para que se pueda usar para la autenticación a través de IAM. El rol solo se usa para iniciar sesión en el clúster y no tiene ningún permiso en lo que respecta a la consola o las llamadas a la API.

Cree un nuevo rol para el entorno de Amazon MWAA siguiendo los pasos que se indican en Rol de ejecución de Amazon MWAA. Sin embargo, en lugar de crear y adjuntar las políticas descritas en ese tema, adjunte la siguiente política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:us-east-1:111122223333: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:us-east-1:111122223333: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:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/
${EKS_CLUSTER_NAME}"
        }
    ]
}
```

Una vez que haya creado el rol, edite el entorno de Amazon MWAA para usar el rol que haya creado como rol de ejecución para el entorno. Para cambiar el rol, edite el entorno que se vaya a utilizar. Seleccione el rol de ejecución en Permisos.

Problemas conocidos:

- Existe un problema conocido relacionado con los ARN de los roles por el que las subrutas no pueden autenticarse con Amazon EKS. La solución consiste en crear el rol de servicio de forma manual, en lugar de utilizar el que creó Amazon MWAA. Para obtener más información, consulte

los roles con rutas que no funcionan cuando la ruta está incluida en su ARN en el mapa de configuración de aws-auth

- Si la lista de servicios de Amazon MWAA no está disponible en IAM, debe elegir una política de servicio alternativa, como Amazon EC2, y, a continuación, actualizar la política de confianza del rol para que coincida con lo siguiente:

  JSON

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

  Para obtener más información, consulte cómo usar las políticas de confianza con roles de IAM.

Creación del archivo requirements.txt

Para usar el código de ejemplo de esta sección, compruebe que ha añadido una de las siguientes opciones de base de datos a sus `requirements.txt`. Consulte Instalación de dependencias de Python para obtener más información.

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Creación de un mapeo de identidad para Amazon EKS

Utilice el ARN del rol que creó en el siguiente comando para crear un mapeo de identidad para Amazon EKS. Cambie la región *`us-east-1`* a la región donde haya creado el entorno. Sustituya el ARN por el rol y, por último, sustituya *`mwaa-execution-role`* por el rol de ejecución de su entorno.

```
eksctl create iamidentitymapping \
--region us-east-1 \
--cluster mwaa-eks \
--arn arn:aws:iam::123456789012:role/mwaa-execution-role \
--username mwaa-service
```

Creación del **kubeconfig**

Utilice el siguiente comando para crear el rol `kubeconfig`:

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

Si utilizó un perfil específico al ejecutar `update-kubeconfig`, tendrá que eliminar la sección `env:` añadida al archivo kube_config.yaml para que funcione correctamente con Amazon MWAA. Para ello, elimine lo siguiente del archivo y guárdelo:

```
env:
 - name: AWS_PROFILE
 value: profile_name
```

Creación de un DAG

Utilice el siguiente ejemplo de código para crear un archivo de Python, por ejemplo, `mwaa_pod_example.py` para el DAG.

```
"""
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'
                       )
```

Adición del DAG y **`kube_config.yaml`** al bucket de Amazon S3

Coloque el DAG que creó y el archivo `kube_config.yaml` en el bucket de Amazon S3 para el entorno de Amazon MWAA. Puede colocar los archivos en el bucket a través de la consola de Amazon S3 o el AWS Command Line Interface.

Habilitación y activación del ejemplo

En Apache Airflow, habilite el ejemplo y actívelo.

Cuando se ejecute y se complete correctamente, utilice el siguiente comando para verificar el pod:

```
kubectl get pods -n mwaa
```

Obtendrá un resultado similar al siguiente:

```
NAME READY STATUS RESTARTS AGE
mwaa-pod-test-aa11bb22cc334444555566667778888 0/1 Completed 0 2m23s
```

A continuación, puede verificar la salida del pod con el siguiente comando. Reemplace el nombre del valor por el valor devuelto por el comando anterior:

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc334444555566667778888
```

Conexión a Amazon ECS mediante el **ECSOperator**

En esta página, se describe cómo puede usar el `ECSOperator` para conectarse a un contenedor de Amazon Elastic Container Service (Amazon ECS) desde Amazon MWAA. En los siguientes pasos, añadirá los permisos necesarios al rol de ejecución de su entorno, utilizará una plantilla de CloudFormation para crear un clúster de Fargate de Amazon ECS y, por último, creará y cargará un DAG que se conecte a su nuevo clúster.

Temas

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y con Apache Airflow v3 en Python 3.11.

Requisitos previos

Para usar el código de muestra de esta página, necesitará lo siguiente:

- Un entorno de Amazon MWAA.

Permisos

- El rol de ejecución de su entorno necesita permiso para ejecutar tareas en Amazon ECS. Puede adjuntar la política AmazonECS_FullAccess gestionada por AWS a su rol de ejecución o crear y adjuntar la política siguiente a su rol de ejecución.

  JSON

```
{
    "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"
                }
            }
```

```
        }
    ]
}
```

- Además de agregar los permisos necesarios para ejecutar tareas en Amazon ECS, también debe modificar la declaración de política de CloudWatch Logs en su rol de ejecución de Amazon MWAA para permitir el acceso al grupo de registro de tareas de Amazon ECS, como se muestra a continuación. El grupo de registro de Amazon ECS se crea mediante la plantilla de CloudFormation en the section called “Creación de un clúster de Amazon ECS.”.

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
  ],
  "Resource": [
    "arn:aws:logs:us-east-1:123456789012:log-group:airflow-environment-name-*",
    "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
  ]
}
```

Para obtener más información sobre el rol de ejecución de Amazon MWAA y sobre cómo adjuntar una política, consulte Rol de ejecución.

Creación de un clúster de Amazon ECS.

Con la siguiente plantilla de CloudFormation, creará un clúster de Fargate de Amazon ECS para usarlo con su flujo de trabajo de Amazon MWAA. Para obtener más información, consulte cómo crear una definición de tarea en la guía para desarrolladores de Amazon Elastic Container Service.

1. Cree un archivo JSON con el siguiente contenido y guárdelo 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. Para crear una pila nueva, utilice el comando de la AWS CLI siguiente en el símbolo del sistema. Debe reemplazar los valores `SecurityGroups` y `SubnetIds` por los valores de los grupos de seguridad y las subredes de su entorno de 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
```

También puede utilizar el siguiente script de shell: El script recupera los valores necesarios para los grupos de seguridad y las subredes de su entorno mediante el comando de la AWS CLI de `get-environment` y, a continuación, crea la pila en consecuencia. Para ejecutar el script, siga los pasos que se detallan a continuación.

a. Copie y guarde el script como `ecs-stack-helper.sh` en el mismo directorio que su plantilla de 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. Ejecute el script mediante los comandos siguientes. Reemplace el `environment-name` y el `stack-name` con su información.

```
chmod +x ecs-stack-helper.sh
./ecs-stack-helper.bash environment-name stack-name
```

Si se realiza correctamente, aparecerá el siguiente resultado en el que se muestra el identificador de su nueva pila de CloudFormation.

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

Una vez que haya completado su pila de CloudFormation y AWS haya aprovisionado sus recursos de Amazon ECS, estará listo para crear y cargar su DAG.

Código de ejemplo

1. Abra un símbolo del sistema y vaya hasta el directorio en el que está almacenado el código DAG. Por ejemplo:

```
cd dags
```

2. Copie el contenido del siguiente código de ejemplo y guárdelo localmente como `mwaa-ecs-operator.py`; a continuación, cargue su nuevo DAG a 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

En el ejemplo del DAG, para `awslogs_group`, puede que tenga que modificar el grupo de registro con el nombre de su grupo de registro de tareas de Amazon ECS. El ejemplo asume un grupo de registro denominado “`mwaa-ecs-zero`”. Para `awslogs_stream_prefix`, utilice el prefijo del flujo de registro de tareas de Amazon ECS. El ejemplo asume un prefijo de flujo de registro, `ecs`.

3. Ejecute el siguiente comando de la AWS CLI para copiar el DAG en el bucket de su entorno y, a continuación, active el DAG mediante la interfaz de usuario de Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si se ejecuta correctamente, verá un resultado similar al siguiente en los registros de tareas del `ecs_operator_task` en el 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
```

Uso de dbt con Amazon MWAA

En este tema se muestra cómo puede utilizar dbt y Postgres con Amazon MWAA. En los siguientes pasos, añadirá las dependencias necesarias a su `requirements.txt` y cargará un ejemplo de proyecto de dbt en el bucket de Amazon S3 de su entorno. A continuación, utilizará un ejemplo de DAG para comprobar que Amazon MWAA ha instalado las dependencias y, por último, utilizará el `BashOperator` para ejecutar el proyecto dbt.

Temas

- Uso de un DAG para verificar la instalación de la dependencia de dbt
- Uso de un DAG para ejecutar un proyecto dbt

Versión

Puede usar el código de ejemplo que aparece en esta página con Apache Airflow v2 en Python 3.10 y Apache Airflow v3 en Python 3.11.

Requisitos previos

Para completar los siguientes pasos, necesitará lo siguiente:

- Un entorno de Amazon MWAA que utilice Apache Airflow v2.2.2. Este ejemplo se escribió y probó con la versión 2.2.2. Es posible que tenga que modificar el ejemplo para usarlo con otras versiones de Apache Airflow.
- Un ejemplo de proyecto de dbt. Para empezar a usar dbt con Amazon MWAA, puede crear una ramificación y clonar el proyecto de inicio de dbt desde el repositorio dbt-labs de GitHub.

Dependencias

Para usar Amazon MWAA con dbt, agregue el siguiente script de inicio a su entorno. Para obtener más información, consulte Cómo usar un script de inicio con 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
```

En las secciones siguientes, cargará el directorio de su proyecto dbt a Amazon S3 y ejecutará un DAG que valide si Amazon MWAA ha instalado las dependencias dbt requeridas correctamente.

Carga de un proyecto de dbt en Amazon S3

Para poder utilizar un proyecto dbt con su entorno Amazon MWAA, cargue todo el directorio del proyecto a la carpeta de `dags` de su entorno. Cuando el entorno se actualice, Amazon MWAA descargará el directorio dbt en la carpeta local de `usr/local/airflow/dags/`.

Pasos para cargar un proyecto de dbt en Amazon S3

1. Vaya al directorio en el que clonó el proyecto de inicio de dbt.
2. Ejecute el siguiente comando de la AWS CLI de Amazon S3 para copiar el contenido del proyecto en la carpeta de `dags` de su entorno de forma recursiva mediante el parámetro `--recursive`. El comando creará un subdirectorio llamado "`dbt`" que puede usar para todos sus proyectos de dbt. Si el subdirectorio ya existe, los archivos del proyecto se copiarán en el directorio existente, no se creará un nuevo directorio. El comando también creará un subdirectorio dentro del directorio de `dbt` para este proyecto inicial específico.

   ```
   aws s3 cp dbt-starter-project s3://amzn-s3-demo-bucket/dags/dbt/dbt-starter-project
    --recursive
   ```

   Puede utilizar diferentes nombres para los subdirectorios de los proyectos a fin de organizar varios proyectos de dbt dentro del directorio principal de `dbt`.

Uso de un DAG para verificar la instalación de la dependencia de dbt

El siguiente DAG utiliza un `BashOperator` y un comando de bash para comprobar si Amazon MWAA ha instalado correctamente las dependencias dbt especificadas en el archivo `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/python3-virtualenv/dbt-env/bin/dbt --version"
   )
```

Realice los siguientes pasos para ver los registros de tareas y comprobar que dbt y sus dependencias se hayan instalado.

1. Vaya a la consola de Amazon MWAA y, a continuación, seleccione Abrir interfaz de usuario de Airflow en la lista de entornos disponibles.

2. En la UI de Apache Airflow, busque el DAG de `dbt-installation-test` en la lista y, a continuación, elija la fecha que aparece debajo de la columna `Last Run` para abrir la última tarea completada.
3. Con Vista de gráfico, elija la tarea `bash_command` para abrir los detalles de la instancia de la tarea.
4. Seleccione Registro para abrir los registros de tareas y, a continuación, compruebe que muestran la versión de dbt especificada en `requirements.txt` correctamente.

Uso de un DAG para ejecutar un proyecto dbt

El siguiente DAG utiliza un `BashOperator` para copiar los proyectos dbt que ha cargado en Amazon S3 desde el directorio local `usr/local/airflow/dags/` al directorio accesible para escritura `/tmp` y, a continuación, ejecuta el proyecto dbt. Los comandos de bash asumen un proyecto dbt inicial titulado "`dbt-starter-project`". Modifique el nombre del directorio de acuerdo con el nombre del directorio de su proyecto.

```
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"
```

```
)
```

Blogs y tutoriales de AWS

- Uso de Amazon EKS y Amazon MWAA for Apache Airflow v2.x

Prácticas recomendadas para el uso de Amazon Managed Workflows para Apache Airflow

En esta guía se describen las prácticas recomendadas para utilizar Amazon Managed Workflows para Apache Airflow.

Temas

Ajuste del desempeño de Apache Airflow en Amazon MWAA

En esta página, se describe cómo ajustar el rendimiento de un entorno de Amazon Managed Workflows para Apache Airflow mediante Uso de las opciones de configuración de Apache Airflow en Amazon MWAA.

Contenido

Adición de una opción de configuración de Apache Airflow

Siga los siguientes pasos para agregar una opción de configuración de Apache Airflow a su entorno.

1. Abra la página Entornos en la consola de Amazon MWAA.

2. Seleccione un entorno.
3. Elija Editar.
4. Elija Siguiente.
5. Seleccione Agregar configuración personalizada en el panel Opciones de configuración de Airflow.
6. En la lista desplegable, elija una opción de configuración e introduzca un valor. También puede escribir una configuración personalizada e introducir un valor.
7. Seleccione Agregar configuración personalizada para cada configuración que desee agregar.
8. Seleccione Save.

Consulte Uso de las opciones de configuración de Apache Airflow en Amazon MWAA para obtener más información.

Programador de Apache Airflow

El programador de Apache Airflow es un componente básico de Apache Airflow. Si hay algún problema en el programador, es posible que no se analicen los DAG ni se programen las tareas. Para obtener más información sobre cómo ajustar el programador de Apache Airflow, consulte cómo ajustar el funcionamiento del programador en el sitio web de documentación de Apache Airflow.

Parameters

En esta sección, se describen las opciones de configuración disponibles para el programador de Apache Airflow (Apache Airflow v2 y versiones posteriores) y sus casos de uso.

Apache Airflow v3

Configuración	Caso de uso
celery.sync_parallelism Cantidad de procesos que utiliza el ejecutor Celery para sincronizar el estado de las tareas. Predeterminado: 1	Esta opción puede utilizarse para evitar conflictos en las colas ya que limita los procesos que utiliza el ejecutor Celery. El valor 1 se utiliza de forma predeterminada para evitar errores al entregar los registros de tareas a los registros de CloudWatch. Si se emplea el valor `0`, se estará usando la

Configuración	Caso de uso
	cantidad máxima de procesos, lo que podría provocar errores al entregar los registros de tareas.
scheduler.scheduler_idle_sleep_time Cantidad de segundos que debe esperar entre procesamientos consecutivos de archivos DAG en el “bucle” del programador. Predeterminado: 1	Se puede usar esta opción para reducir el uso de la CPU por parte del programador aumentando el tiempo de inactividad del programador una vez que haya terminado de recuperar los resultados del análisis de los DAG, de buscar las tareas y ponerlas en cola y de ejecutar las tareas en cola en el ejecutor. Al aumentar este valor, se consume la cantidad de hilos del programador que se ejecutan en un entorno en `dag_processor.parsing_processes` en el caso de Apache Airflow v2 y v3. Esto podría reducir la capacidad de los programadores de analizar los DAG y aumentar el tiempo que tardan los DAG en propagarse en el servidor web.
scheduler.max_dagruns_to_create_per_loop Cantidad máxima de DAG por “bucle” del programador para los que se pueden crear DagRuns. Valor predeterminado: 10	Se puede usar esta opción para liberar recursos destinados a la programación de tareas reduciendo la cantidad máxima de DagRuns para el “bucle” del programador.

Configuración	Caso de uso
dag_processor.parsing_processes Cantidad de hilos que el programador puede ejecutar en paralelo para programar los DAG. De forma predeterminada: use `(2 * number of vCPUs) - 1`	Se puede usar esta opción para liberar recursos reduciendo la cantidad de procesos que el programador ejecuta en paralelo para analizar los DAG. Recomendamos utilizar una cantidad baja si el análisis de los DAG afecta a la programación de tareas. Debe especificar un valor inferior al recuento de la CPU virtual (vCPU) de su entorno. Consulte los límites para obtener más información.

Apache Airflow v2

Configuración	Caso de uso
celery.sync_parallelism Cantidad de procesos que utiliza el ejecutor Celery para sincronizar el estado de las tareas. Predeterminado: 1	Esta opción puede utilizarse para evitar conflictos en las colas ya que limita los procesos que utiliza el ejecutor Celery. El valor 1 se utiliza de forma predeterminada para evitar errores al entregar los registros de tareas a los registros de CloudWatch. Si se emplea el valor `0`, se estará usando la cantidad máxima de procesos, lo que podría provocar errores al entregar los registros de tareas.
scheduler.idle_sleep_time Cantidad de segundos que debe esperar entre procesamientos consecutivos de archivos DAG en el “bucle” del programador. Predeterminado: 1	Se puede usar esta opción para reducir el uso de la CPU por parte del programador aumentando el tiempo de inactividad del programador una vez que haya terminado de recuperar los resultados del análisis de los DAG, de buscar las tareas y ponerlas en cola y de ejecutar las tareas en cola en el

Configuración	Caso de uso
	ejecutor. Al aumentar este valor, se consume la cantidad de hilos del programador que se ejecutan en un entorno en `scheduler .parsing_processes` en el caso de Apache Airflow v2 y v3. Esto podría reducir la capacidad de los programadores de analizar los DAG y aumentar el tiempo que tardan los DAG en propagarse en el servidor web.
scheduler.max_dagruns_to_create_per_loop Cantidad máxima de DAG por “bucle” del programador para los que se pueden crear DagRuns. Valor predeterminado: 10	Se puede usar esta opción para liberar recursos destinados a la programación de tareas reduciendo la cantidad máxima de DagRuns para el “bucle” del programador.
scheduler.parsing_processes Cantidad de hilos que el programador puede ejecutar en paralelo para programar los DAG. De forma predeterminada: use `(2 * number of vCPUs) - 1`	Se puede usar esta opción para liberar recursos reduciendo la cantidad de procesos que el programador ejecuta en paralelo para analizar los DAG. Recomendamos utilizar una cantidad baja si el análisis de los DAG afecta a la programación de tareas. Debe especificar un valor inferior al recuento de la CPU virtual (vCPU) de su entorno. Consulte los límites para obtener más información.

Límites

En esta sección, se describen los límites que debe tener en cuenta al ajustar los parámetros predeterminados del programador.

scheduler.parsing_processes, scheduler.max_threads (solo para la versión 2)

Se permiten dos hilos por vCPU en cada clase de entorno. Reserve al menos un hilo para el programador en cada clase de entorno. Si nota un retraso en la programación de las tareas, es posible que deba aumentar su clase de entorno. Por ejemplo, un entorno grande tendrá una instancia de contenedor de Fargate de 4 vCPU para el programador. Esto significa que hay un máximo de 7 hilos disponibles en total para que los utilicen otros procesos. Es decir: hay dos hilos para cada una de las cuatro vCPU, menos uno que es para el programador en sí. Tal como se muestra a continuación, el valor que especifique en `scheduler.max_threads` (solo para la versión 2) y en `scheduler.parsing_processes` no debe ser superior a la cantidad de hilos que tenga disponibles en una clase de entorno.

- mw1.small: no debe destinarse más de 1 hilo al resto de procesos. El hilo restante debe reservarse para el programador.
- mw1.medium: no deben destinarse más de 3 hilos al resto de procesos. El hilo restante debe reservarse para el programador.
- mw1.large: no deben destinarse más de 7 hilos al resto de procesos. El hilo restante debe reservarse para el programador.

Carpetas de los DAG

El programador de Apache Airflow analiza de forma continua la carpeta de los DAG de su entorno. en busca de cualquier archivo que contenga `plugins.zip`, o cualquier archivo Python (`.py`) que contenga en sus instrucciones de importación la palabra “airflow”. Los objetos DAG en Python que se creen se incluirán en una DagBag para que el programador pueda procesarla a fin de determinar qué tareas deben programarse, en caso necesario. El análisis de los archivos DAG se realiza independientemente de si los archivos contienen objetos DAG viables o no.

Parameters

En esta sección, se describen las opciones de configuración disponibles para la carpeta de los DAG (Apache Airflow v2 y versiones posteriores) y sus casos de uso.

Apache Airflow v3

Configuración	Caso de uso
dag_processor.refresh_interval	

Configuración	Caso de uso
Cantidad de segundos durante los cuales debe escanearse la carpeta de los DAG en busca de archivos nuevos. Valor predeterminado: 300 segundos	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos destinados a analizar la carpeta de los DAG. Recomendamos aumentar este valor si observa que los tiempos de análisis en `total_parse_time metrics` son muy largos; esto puede deberse a que hay una gran cantidad de archivos en su carpeta de DAG.
dag_processor.min_file_process_interval Cantidad de segundos que transcurren desde que el programador analiza un DAG hasta que se reflejan las actualizaciones del mismo. Valor predeterminado: 30 segundos	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que el programador espera antes de analizar un DAG. Por ejemplo, si especifica un valor de `30`, el archivo DAG se analizará cada 30 segundos. Recomendamos mantener elevado dicho valor para reducir el uso de la CPU en su entorno.

Apache Airflow v2

Configuración	Caso de uso
scheduler.dag_dir_list_interval Cantidad de segundos durante los cuales debe escanearse la carpeta de los DAG en busca de archivos nuevos. Valor predeterminado: 300 segundos	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos destinados a analizar la carpeta de los DAG. Recomendamos aumentar este valor si observa que los tiempos de análisis en `total_parse_time metrics` son muy largos; esto puede deberse a que hay una gran cantidad de archivos en su carpeta de DAG.

Configuración	Caso de uso
scheduler.min_file_process_interval Cantidad de segundos que transcurren desde que el programador analiza un DAG hasta que se reflejan las actualizaciones del mismo. Valor predeterminado: 30 segundos	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que el programador espera antes de analizar un DAG. Por ejemplo, si especifica un valor de `30`, el archivo DAG se analizará cada 30 segundos. Recomendamos mantener elevado dicho valor para reducir el uso de la CPU en su entorno.

Archivos DAG

Como parte del bucle del programador de Apache Airflow, los archivos DAG individuales se analizan para extraer los objetos DAG en Python. En Apache Airflow v2 y versiones posteriores, el programador analiza simultáneamente un número máximo de procesos de análisis. Para que se vuelva a analizar el mismo archivo, deben transcurrir primero los segundos especificados en `scheduler.min_file_process_interval` (versión 2) o en `dag_processor.min_file_process_interval` (versión 3).

Parameters

En esta sección, se describen las opciones de configuración disponibles para los archivos DAG de Apache Airflow (Apache Airflow v2 y versiones posteriores) y sus casos de uso.

Apache Airflow v3

Configuración	Caso de uso
dag_processor.dag_file_processor_timeout Cantidad de segundos que transcurren antes de que el DAGFileProcessor interrumpa el procesamiento de un archivo DAG. Valor predeterminado: 50 segundos	Esta opción puede utilizarse para liberar recursos aumentando el tiempo que transcurr e hasta la interrupción del DAGFileProcessor. Le recomendamos que aumente este valor si observa interrupciones en los registros de procesamiento de los DAG que impiden que se carguen DAG viables.

Configuración	Caso de uso
core.dagbag_import_timeout Cantidad segundos que transcurren antes de que se interrumpe la importación de un archivo en Python. Valor predeterminado: 30 segundos	Se puede usar esta opción para liberar recursos aumentando el tiempo que transcurre hasta que el programador interrumpe la importación de un archivo en Python para extraer los objetos DAG. Esta opción se procesa como parte del "bucle" del programador y debe incluir un valor inferior al especificado en `dag_processor.dag_file_processor_timeout` .
core.min_serialized_dag_update_interval Cantidad mínima de segundos que transcurren hasta que se actualizan los DAG serializados en la base de datos. Valor predeterminado: 30	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que deben transcurrir hasta que se actualizan los DAG serializados en la base de datos. Recomendamos aumentar este valor si dispone de una gran cantidad de DAG o DAG complejos. Al aumentar este valor, se reduce la carga del programador y de la base de datos, ya que los DAG se serializan.
core.min_serialized_dag_fetch_interval Cantidad de segundos que se tarda en volver a extraer un DAG serializado de la base de datos una vez que ya se ha cargado en la DagBag. Valor predeterminado: 10	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que se tarda en volver a extraer un DAG serializado. El valor debe ser superior al valor especificado en `core.min_serialized_dag_update_interval` para reducir las tasas de "escritura" de la base de datos. Al aumentar este valor, se reduce la carga del servidor web y de la base de datos, ya que los DAG se serializan.

Apache Airflow v2

Configuración	Caso de uso
core.dag_file_processor_timeout Cantidad de segundos que transcurren antes de que el DAGFileProcessor interrumpa el procesamiento de un archivo DAG. Valor predeterminado: 50 segundos	Esta opción puede utilizarse para liberar recursos aumentando el tiempo que transcurre hasta la interrupción del DAGFileProcessor. Le recomendamos que aumente este valor si observa interrupciones en los registros de procesamiento de los DAG que impiden que se carguen DAG viables.
core.dagbag_import_timeout Cantidad segundos que transcurren antes de que se interrumpe la importación de un archivo en Python. Valor predeterminado: 30 segundos	Se puede usar esta opción para liberar recursos aumentando el tiempo que transcurre hasta que el programador interrumpe la importación de un archivo en Python para extraer los objetos DAG. Esta opción se procesa como parte del “bucle” del programador y debe incluir un valor inferior al especificado en `core.dag_file_processor_timeout` .
core.min_serialized_dag_update_interval Cantidad mínima de segundos que transcurren hasta que se actualizan los DAG serializados en la base de datos. Valor predeterminado: 30	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que deben transcurrir hasta que se actualizan los DAG serializados en la base de datos. Recomendamos aumentar este valor si dispone de una gran cantidad de DAG o DAG complejos. Al aumentar este valor, se reduce la carga del programador y de la base de datos, ya que los DAG se serializan.
core.min_serialized_dag_fetch_interval	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de segundos que se tarda en volver a extraer

Configuración	Caso de uso
Cantidad de segundos que se tarda en volver a extraer un DAG serializado de la base de datos una vez que ya se ha cargado en la DagBag. Valor predeterminado: 10	un DAG serializado. El valor debe ser superior al valor especificado en `core.min_serialized_dag_update_interval` para reducir las tasas de “escritura” de la base de datos. Al aumentar este valor, se reduce la carga del servidor web y de la base de datos, ya que los DAG se serializan.

Tareas

Tanto el programador como los procesos de trabajo de Apache Airflow intervienen en la puesta de tareas en la cola y en su retirada de la misma. El programador toma las tareas analizadas que ya están a punto para ser programadas y modifica su estado de Ninguno a Programado. El ejecutor, que también se ejecuta en el contenedor del programador de Fargate, pone esas tareas en cola y cambia su estado a En cola. Cuando los procesos de trabajo tengan capacidad para ello, tomarán la tarea de la cola y cambiarán su estado a En ejecución, que posteriormente cambiará a Correcto o Error en función de si se ha logrado llevar a cabo la tarea correctamente o si algo ha fallado.

Parameters

En esta sección se describen las opciones de configuración disponibles para las tareas de Apache Airflow y sus casos de uso.

Las opciones de configuración predeterminadas que Amazon MWAA anula están marcadas en *`rojo`*.

Apache Airflow v3

Configuración	Caso de uso
core.parallelism Cantidad máxima de instancias de tareas que pueden tener el estado `Running`.	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de instancias de tareas que se pueden ejecutar simultáneamente. El valor que se especifique debe ser igual al número de procesos de trabajo

Configuración	Caso de uso
Valor predeterminado: Se establece dinámicamente en función de `(maxWorkers * maxCeleryWorkers) / schedulers * 1.5` .	disponibles por la densidad de tareas de los procesos de trabajo. Recomendamos cambiar este valor únicamente cuando haya una gran cantidad de tareas atascadas en los estados de ejecución o de cola.
core.execute_tasks_new_python_interpreter Determina si Apache Airflow ejecuta tareas mediante la bifurcación del proceso principal o la creación de un nuevo proceso de Python. Valor predeterminado: `True`	Cuando este parámetro esté establecido en `True`, Apache Airflow reconocerá los cambios que realice en sus complementos como un nuevo proceso en Python creado para ejecutar tareas.
celery.worker_concurrency Amazon MWAA anula la instalación base de Airflow para que esta opción escale procesos de trabajo como parte de su componente de escalado automático. Valor predeterminado: No corresponde	*Se pasa por alto cualquier valor especificado para esta opción.*

Configuración	Caso de uso
celery.worker_autoscale Simultaneidad de tareas de los procesos de trabajo. Valores predeterminados: • mw1.micro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mw1.large - 20,0 • mw1.xlarge - 40,0 • mw1.2xlarge - 80,0	Se puede usar esta opción para liberar recursos reduciendo la simultaneidad `minimum` y `maximum` de tareas de los procesos de trabajo. Los procesos de trabajo aceptarán el `maximum` de tareas simultáne as configuradas, independientemente de si hay recursos suficientes para realizarlas. Si las tareas se programan sin que haya recursos suficientes, se producirá un error de inmediato. Recomendamos cambiar este valor en el caso de que las tareas consuman muchos recursos; reduzca los valores por debajo de los valores predeterminados para que haya una mayor capacidad por tarea.

Apache Airflow v2

Configuración	Caso de uso
core.parallelism Cantidad máxima de instancias de tareas que pueden tener el estado `Running`. Valor predeterminado: Se establece dinámicamente en función de `(maxWorke rs * maxCeleryWorkers) / schedulers * 1.5` .	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de instancia s de tareas que se pueden ejecutar simultáne amente. El valor que se especifique debe ser igual al número de procesos de trabajo disponibles por la densidad de tareas de los procesos de trabajo. Recomendamos cambiar este valor únicamente cuando haya una gran cantidad de tareas atascadas en los estados de ejecución o de cola.
core.dag_concurrency	Esta opción puede utilizarse para liberar recursos aumentando la cantidad de instancia

Configuración	Caso de uso
Cantidad de instancias de tareas que se pueden ejecutar simultáneamente para cada DAG. Predeterminado: 10000	s de tareas que pueden ejecutarse simultáne amente. Por ejemplo, si tiene cien DAG con diez tareas paralelas y quiere que todos los DAG se ejecuten simultáneamente, puede calcular el paralelismo máximo multiplicando el número de procesos de trabajo disponibles por la densidad de las tareas de los procesos de trabajo en `celery.worker_conc urrency` y dividirlo por la cantidad de DAG.
core.execute_tasks_new_python_interpreter Determina si Apache Airflow ejecuta tareas mediante la bifurcación del proceso principal o la creación de un nuevo proceso de Python. Valor predeterminado: `True`	Cuando este parámetro esté establecido en `True`, Apache Airflow reconocerá los cambios que realice en sus complementos como un nuevo proceso en Python creado para ejecutar tareas.
celery.worker_concurrency Amazon MWAA anula la instalación base de Airflow para que esta opción escale procesos de trabajo como parte de su componente de escalado automático. Valor predeterminado: No corresponde	*Se pasa por alto cualquier valor especificado para esta opción.*

Configuración	Caso de uso
celery.worker_autoscale Simultaneidad de tareas de los procesos de trabajo. Valores predeterminados: • mw1.micro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mw1.large - 20,0 • mw1.xlarge - 40,0 • mw1.2xlarge - 80,0	Se puede usar esta opción para liberar recursos reduciendo la simultaneidad `minimum` y `maximum` de tareas de los procesos de trabajo. Los procesos de trabajo aceptarán el `maximum` de tareas simultáne as configuradas, independientemente de si hay recursos suficientes para realizarlas. Si las tareas se programan sin que haya recursos suficientes, se producirá un error de inmediato. Recomendamos cambiar este valor en el caso de que las tareas consuman muchos recursos; reduzca los valores por debajo de los valores predeterminados para que haya una mayor capacidad por tarea.

Administración de las dependencias de Python en requirements.txt

En esta página, se describe cómo instalar y administrar las dependencias de Python en un archivo `requirements.txt` para entornos de Amazon Managed Workflows para Apache Airflow.

Contenido

- Visualización de registros en la consola de registros de CloudWatch
- Visualización de los errores en la UI de Apache Airflow
  - Inicio de sesión en Apache Airflow
- Ejemplos de escenarios de requirements.txt

Pruebas de los DAG mediante la utilidad de la CLI de Amazon MWAA

- La utilidad de la interfaz de la línea de comandos (CLI) replica entornos en Amazon Managed Workflows para Apache Airflow de forma local.
- La CLI crea localmente una imagen de contenedor de Docker similar a una imagen de producción de Amazon MWAA. Esto le permite ejecutar un entorno local de Apache Airflow para desarrollar y probar los DAG, los complementos personalizados y las dependencias antes de implementarlos en Amazon MWAA.
- Para ejecutar la CLI, consulte aws-mwaa-docker-images en GitHub.

Instalación de dependencias de Python mediante el formato de archivo de requisitos PyPI.org

En la siguiente sección se describen las distintas maneras de instalar las dependencias de Python según el archivo de requisitos Requirements file format de PyPi.org.

Opción uno: dependencias de Python desde el Índice de paquetes de Python

La siguiente sección describe cómo especificar las dependencias de Python desde el Python Package Index en un archivo `requirements.txt`.

Apache Airflow v3

1. Hacer una prueba local. Añada bibliotecas adicionales de forma iterativa para encontrar la combinación adecuada de paquetes y sus versiones antes de crear un archivo `requirements.txt`. Para ejecutar la utilidad de la CLI de Amazon MWAA, consulte aws-mwaa-docker-images en GitHub.
2. Revise los extras del paquete Apache Airflow. Para acceder a una lista de los paquetes instalados para Apache Airflow v3 en Amazon MWAA, consulte aws-mwaa-docker-images `requirements.txt` en el sitio web de GitHub.

3. Añada instrucciones respecto a las restricciones. Añada el archivo de restricciones para su entorno Apache Airflow v3 en la parte superior del archivo `requirements.txt`. Los archivos de restricciones de Apache Airflow especifican las versiones de proveedores disponibles en el momento de la publicación de Apache Airflow.

   En el siguiente ejemplo, sustituya *{environment-version}* con el número de versión de su entorno y *{Python-version}* con la versión de Python compatible con su entorno.

   Para obtener información sobre la versión de Python compatible con su entorno de Apache Airflow, consulte las versiones de Apache Airflow.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   Si el archivo de restricciones determina que el paquete `xyz==1.0` no es compatible con otros paquetes de su entorno, `pip3 install` no podrá impedir que se instalen bibliotecas incompatibles en su entorno. Si se genera un error al instalar algún paquete, podrá ver los registros de errores de cada componente de Apache Airflow (el programador, el proceso de trabajo y el servidor web) en el flujo de registro correspondiente en los registros de CloudWatch. Para obtener más información sobre los tipos de registros, consulte the section called “Visualización de registros de Airflow”.

4. Paquetes de Apache Airflow. Añada los extras del paquete y la versión (==). Esto ayuda a evitar que se instalen en su entorno paquetes del mismo nombre, pero de una versión diferente.

   ```
   apache-airflow[package-extra]==2.5.1
   ```

5. Bibliotecas Python. Añada el nombre del paquete y la versión (==) al archivo `requirements.txt`. Esto permite evitar que se lleve a cabo una actualización futura de última hora de PyPi.org automáticamente.

   ```
   library == version
   ```

   Example Boto3 y psycopg2-binary

   Este caso se proporciona como ejemplo. Las bibliotecas boto y psycopg2-binary vienen incluidas en la instalación base de Apache Airflow v3, por lo que no es necesario especificarlas en un archivo `requirements.txt`.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Si un paquete se especifica sin versión, Amazon MWAA instalará la última versión del paquete que encuentre en PyPi.org. Esta versión puede entrar en conflicto con otros paquetes de su `requirements.txt`.

Apache Airflow v2

1. Hacer una prueba local. Añada bibliotecas adicionales de forma iterativa para encontrar la combinación adecuada de paquetes y sus versiones antes de crear un archivo `requirements.txt`. Para ejecutar la utilidad de la CLI de Amazon MWAA, consulte aws-mwaa-docker-images en GitHub.
2. Revise los extras del paquete Apache Airflow. Para acceder a una lista de los paquetes instalados para Apache Airflow v2 en Amazon MWAA, consulte aws-mwaa-docker-images `requirements.txt` en el sitio web de GitHub.
3. Añada instrucciones respecto a las restricciones. Añada el archivo de restricciones para su entorno Apache Airflow v2 en la parte superior del archivo `requirements.txt`. Los archivos de restricciones de Apache Airflow especifican las versiones de proveedores disponibles en el momento de la publicación de Apache Airflow.

   A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que utilice.

   En el siguiente ejemplo, sustituya *`{environment-version}`* con el número de versión de su entorno y *`{Python-version}`* con la versión de Python compatible con su entorno.

   Para obtener información sobre la versión de Python compatible con su entorno de Apache Airflow, consulte las versiones de Apache Airflow.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

Si el archivo de restricciones determina que el paquete `xyz==1.0` no es compatible con otros paquetes de su entorno, `pip3 install` no podrá impedir que se instalen bibliotecas incompatibles en su entorno. Si se genera un error al instalar algún paquete, podrá ver los registros de errores de cada componente de Apache Airflow (el programador, el proceso de trabajo y el servidor web) en el flujo de registro correspondiente en los registros de CloudWatch. Para obtener más información sobre los tipos de registros, consulte the section called “Visualización de registros de Airflow”.

4. Paquetes de Apache Airflow. Añada los extras del paquete y la versión (==). Esto ayuda a evitar que se instalen en su entorno paquetes del mismo nombre, pero de una versión diferente.

```
apache-airflow[package-extra]==2.5.1
```

5. Bibliotecas Python. Añada el nombre del paquete y la versión (==) al archivo `requirements.txt`. Esto permite evitar que se lleve a cabo una actualización futura de última hora de PyPi.org automáticamente.

```
library == version
```

Example Boto3 y psycopg2-binary

Este caso se proporciona como ejemplo. Las bibliotecas boto y psycopg2-binary vienen incluidas en la instalación base de Apache Airflow v2, por lo que no es necesario especificarlas en un archivo `requirements.txt`.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Si un paquete se especifica sin versión, Amazon MWAA instalará la última versión del paquete que encuentre en PyPi.org. Esta versión puede entrar en conflicto con otros paquetes de su `requirements.txt`.

Opción dos: archivos wheels de Python (.whl)

El formato wheel de Python es un tipo de paquete diseñado para enviar bibliotecas con artefactos compilados. Los paquetes wheel presentan varias ventajas si se utilizan como método para instalar dependencias en Amazon MWAA:

- Instalación más rápida: los archivos WHL se copian en el contenedor como un único ZIP y se instalan de forma local, sin necesidad de descargarlos uno por uno.
- Menos conflictos: se puede determinar con antelación la compatibilidad de las versiones de los paquetes. De esta manera, no es necesario que `pip` elabore de forma recurrente versiones compatibles.
- Mayor resiliencia: si las bibliotecas se alojan externamente, es posible que los requisitos posteriores cambien con el tiempo, lo que provocará que las versiones sean incompatibles entre los contenedores de un entorno de Amazon MWAA. Al no depender de una fuente externa para las dependencias, todos los contenedores tienen las mismas bibliotecas, independientemente de cuándo se instancie cada contenedor.

Recomendamos seguir los métodos que se indican a continuación para instalar las dependencias de Python en su `requirements.txt` desde un archivo wheel de Python (`.whl`).

Métodos

- Uso del archivo plugins.zip en un bucket de Amazon S3
- Uso de archivos WHL alojados en una URL
- Creación de archivos WHL a partir de DAG

Uso del archivo **`plugins.zip`** en un bucket de Amazon S3

El programador, los procesos de trabajo y el servidor web de Apache Airflow (en el caso de Apache Airflow v2.2.2 y las versiones posteriores) buscan complementos personalizados durante el inicio en el contenedor de Fargate administrado por AWS para su entorno en `/usr/local/airflow/plugins/*`. Este proceso comienza antes del `pip3 install -r requirements.txt` de Amazon MWAA para las dependencias de Python y del inicio del servicio Apache Airflow. Se puede usar un archivo `plugins.zip` con todos aquellos archivos que no quiera que vayan cambiando continuamente durante la ejecución del entorno, o bien, con aquellos a los que no quiera permitir que accedan los usuarios que escriben DAG. Por ejemplo, archivos wheel de la biblioteca Python, archivos PEM de certificados y archivos YAML de configuración.

En la siguiente sección se describe cómo instalar wheels del archivo `plugins.zip` de su bucket de Amazon S3.

1. Descargue los archivos WHL necesarios. Puede usar `pip download` con su `requirements.txt` actual en el aws-mwaa-docker-images de Amazon MWAA o en otro contenedor de Amazon Linux 2 para resolver y descargar los archivos wheel de Python necesarios.

   ```
   pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
   cd "$AIRFLOW_HOME/plugins"
   zip "$AIRFLOW_HOME/plugins.zip" *
   ```

2. Especifique la ruta en su **`requirements.txt`**. Especifique el directorio de complementos al principio de su archivo requirements.txt con `--find-links` e indique a `pip` que instale desde otras fuentes con `--no-index`, tal y como se muestra en el siguiente código:

   ```
   --find-links /usr/local/airflow/plugins
   --no-index
   ```

   Example Ejemplo de wheel en el archivo requirements.txt

   En el siguiente ejemplo, se supone que ha cargado el wheel en un archivo `plugins.zip` en la raíz de su bucket de Amazon S3. Por ejemplo:

   ```
   --find-links /usr/local/airflow/plugins
   --no-index

   numpy
   ```

   Amazon MWAA extrae el wheel `numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl` de la carpeta de `plugins` y lo instala en su entorno.

Uso de archivos WHL alojados en una URL

En la siguiente sección se describe cómo instalar un archivo wheel alojado en una URL. La URL debe ser de acceso público o se debe poder acceder a ella desde la VPC de Amazon personalizada que especificó para su entorno de Amazon MWAA.

- Indique una URL. Indique la URL en la que se encuentra el archivo wheel en su `requirements.txt`.

  Example archivo wheel en una URL pública

  En el siguiente ejemplo, se puede ver cómo se descarga un archivo wheel de un sitio web público.

  ```
  --find-links https://files.pythonhosted.org/packages/
  --no-index
  ```

  Amazon MWAA obtiene el archivo wheel de la URL que había especificado y lo instala en su entorno.

  > **Note**
  >
  > No se puede acceder a las URL desde servidores web privados que requieran instalar requisitos en Amazon MWAA v2.2.2 y versiones posteriores.

Creación de archivos WHL a partir de DAG

Si dispone de un servidor web privado con Apache Airflow v2.2.2 o versiones posteriores y no puede instalar los requisitos porque su entorno no tiene acceso a repositorios externos, puede usar el siguiente DAG para tomar sus requisitos actuales de Amazon MWAA y empaquetarlos en 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://amzn-s3-demo-bucket/{S3_KEY}"
```

```
)
```

Después de ejecutar el DAG, utilice este nuevo archivo como su `plugins.zip` de Amazon MWAA. Si lo desea, puede empaquetarlo con otros complementos. A continuación, actualice su `requirements.txt` que comienza por `--find-links /usr/local/airflow/plugins` y `--no-index` sin agregar `--constraint`.

Este método le permitirá usar las mismas bibliotecas sin conexión.

Opción tres: dependencias de Python alojadas en un repositorio privado conforme con PyPI/PEP-503

En la siguiente sección se describe cómo instalar un elemento adicional de Apache Airflow alojado en una URL privada con autenticación.

1. Añada su nombre de usuario y contraseña como Opciones de configuración de Airflow. Por ejemplo:
   - `foo.user` : *YOUR_USER_NAME*
   - `foo.pass` : *YOUR_PASSWORD*
2. Cree su archivo `requirements.txt`. Sustituya los marcadores de posición del ejemplo que sigue por su URL privada e introduzca el nombre de usuario y la contraseña que haya añadido como Opciones de configuración de Airflow. Por ejemplo:

   ```
   --index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
   ```
3. En caso aplicable, añada las bibliotecas adicionales a su archivo `requirements.txt`. Por ejemplo:

   ```
   --index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
   my-private-package==1.2.3
   ```

Habilitación de registros en la consola de Amazon MWAA

El rol de ejecución de su entorno de Amazon MWAA necesita permiso para enviar los registros a los Registros de Amazon CloudWatch. Para actualizar los permisos de un rol de ejecución, consulte Rol de ejecución de Amazon MWAA.

Puede habilitar los registros de Apache Airflow en los niveles `INFO`, `WARNING`, `ERROR` o `CRITICAL`. A elegir un nivel de registro, Amazon MWAA envía los registros correspondientes a ese nivel y a todos los niveles de gravedad superiores. Por ejemplo, si habilita los registros en el nivel `INFO`, Amazon MWAA envía los registros del nivel `INFO`, así como los registros de los niveles `WARNING`, `ERROR` y `CRITICAL` a los Registros de CloudWatch. Recomendamos habilitar los registros de Apache Airflow en el nivel `INFO` para que el programador pueda ver los registros recibidos para el archivo `requirements.txt`.

Visualización de registros en la consola de registros de CloudWatch

Consulte los registros de Apache Airflow correspondientes al programador encargado de organizar sus flujos de trabajo y analizar su carpeta de `dags`. En los siguientes pasos, se describe cómo abrir el grupo de registros del programador en la consola de Amazon MWAA y cómo ver los registros de Apache Airflow en la consola de registros de CloudWatch.

Para acceder a los registros de un **`requirements.txt`**

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija el Grupo de registro del programador de Airflow en el panel de Monitorización.
4. Seleccione el registro `requirements_install_ip` en los flujos de registro.
5. Consulte la lista de paquetes que se hayan instalado en el entorno en `/usr/local/airflow/.local/bin`. Por ejemplo:

```
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. Consulte la lista de paquetes y compruebe si se produjo algún error en alguno de ellos durante la instalación. Si se produce un error, es posible que aparezca un error similar al siguiente:

```
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))
```

Visualización de los errores en la UI de Apache Airflow

También puede comprobar su UI de Apache Airflow para averiguar si algún error puede estar relacionado con otro problema. El error más común que puede producirse con Apache Airflow en Amazon MWAA es el siguiente:

```
Broken DAG: No module named x
```

Si ve este error en la UI de Apache Airflow, es probable que falte una dependencia obligatoria en su archivo `requirements.txt`.

Inicio de sesión en Apache Airflow

Necesita permisos de Política de acceso a la interfaz de usuario de Apache Airflow: AmazonMWAAWebServerAccess para su Cuenta de AWS en AWS Identity and Access Management (IAM) para ver la UI de Apache Airflow.

Pasos para acceder a la interfaz de usuario de Apache Airflow

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Abrir interfaz de usuario de Airflow.

Ejemplos de escenarios de **`requirements.txt`**

Puede mezclar y combinar diferentes formatos en su `requirements.txt`. En el siguiente ejemplo se utiliza una combinación de las distintas formas de instalar elementos adicionales.

Example Ejemplo de elementos adicionales en PyPI.org y en una URL pública

Utilice la opción `--index-url` al especificar paquetes de PyPI.org y cuando se especifiquen paquetes en una URL pública, como en el caso de las URL de repositorios personalizados que cumplen con el PEP 503.

```
aws-batch == 0.6
```

```
    phoenix-letter >= 0.3

    --index-url http://dist.repoze.org/zope2/2.10/simple
    zopelib
```

Monitorización y métricas de Amazon Managed Workflows para Apache Airflow

La monitorización es un factor importante a la hora de mantener la fiabilidad, la disponibilidad y el rendimiento de Amazon Managed Workflows para Apache Airflow y de las soluciones de AWS. Recomendamos que recopile los datos de supervisión de todas las partes de la solución de AWS para que le resulte más sencillo depurar errores multipuntos en caso de que se produjeran. En este tema se describen los recursos que ofrece AWS para monitorizar su entorno de Amazon MWAA y responder ante posibles inconvenientes.

> **Note**
>
> Las métricas y el registro de Apache Airflow están sujetos a los precios estándar de Amazon CloudWatch.

Para obtener más información sobre la monitorización de Apache Airflow, consulte Registro y monitorización en el sitio web de documentación de Apache Airflow.

Secciones

- Información general sobre la monitorización en Amazon MWAA
- Acceso a los registros de auditoría en AWS CloudTrail
- Visualización de registros en Amazon CloudWatch
- Monitorización de paneles y alarmas en Amazon MWAA
- Métricas del entorno de Apache Airflow en CloudWatch
- Métricas de contenedores, colas y bases de datos para Amazon MWAA

Información general sobre la monitorización en Amazon MWAA

En esta página se describen los servicios de AWS que se utilizan para monitorizar un entorno de Amazon Managed Workflows para Apache Airflow.

Contenido

- Información general sobre Amazon CloudWatch

- AWS CloudTrailInformación general de

Información general sobre Amazon CloudWatch

CloudWatch es un repositorio de métricas para los servicios de AWS que permiten generar estadísticas a partir de las métricas y las dimensiones que haya publicado un servicio. Estas métricas pueden utilizarse para configurar alarmas, calcular estadísticas y presentar los datos en un panel que le ayude a evaluar el estado de su entorno en la consola de Amazon CloudWatch.

Apache Airflow ya está configurado para enviar métricas de un entorno de Amazon Managed Workflows para Apache Airflow a Amazon CloudWatch con StatsD.

Para obtener más información, consulte ¿Qué es Amazon CloudWatch?

AWS CloudTrailInformación general de

CloudTrail es un servicio de auditoría que proporciona un registro de las acciones de un usuario, un rol o un servicio de AWS en Amazon MWAA. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon MWAA, la dirección IP de origen desde la que se realizó la solicitud, quién la realizó, cuándo y otra información de interés.

Para obtener más información, consulta ¿Qué es AWS CloudTrail?.

Acceso a los registros de auditoría en AWS CloudTrail

AWS CloudTrail se habilita en su Cuenta de AWS cuando la crea. CloudTrail registra las actividades que llevan a cabo las entidades de IAM o los servicios de AWS, como Amazon Managed Workflows para Apache Airflow, y las registra como un evento de CloudTrail. Puede consultar, buscar y descargar el historial de eventos de los últimos 90 días desde la consola de CloudTrail. CloudTrail registra todos los eventos que tienen lugar en la consola de Amazon MWAA y todas las llamadas a las API de Amazon MWAA. No registra las acciones de solo lectura, como `GetEnvironment`, ni la acción `PublishMetrics`. En esta página se describe cómo utilizar CloudTrail para monitorizar eventos de Amazon MWAA.

Contenido

- Creación de un registro de seguimiento en CloudTrail
- Consulta de eventos con el historial de eventos de CloudTrail

Creación de un registro de seguimiento en CloudTrail

Para llevar un registro continuo de los eventos de la Cuenta de AWS, incluidos los eventos de Amazon MWAA, debe crear un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. Pero aunque no cree un registro de seguimiento, el historial de eventos estará disponible en la consola de CloudTrail. Por ejemplo, puede determinar qué solicitud se realizó a Amazon MWAA, la dirección IP de origen desde la que se realizó, quién la realizó y cuándo, entre otros, gracias a la información recopilada por CloudTrail. Para obtener más información, consulte cómo crear un registro para su Cuenta de AWS.

Consulta de eventos con el historial de eventos de CloudTrail

Puede resolver problemas operativos e incidentes de seguridad que se hayan producido durante los últimos 90 días en el historial de eventos de la consola de CloudTrail. Por ejemplo, puede consultar los eventos relacionados con la creación, modificación o eliminación de recursos (como usuarios de IAM u otros recursos de AWS) en su Cuenta de AWS por región. Para obtener más información, lea cómo consultar eventos con el historial de eventos de CloudTrail.

1. Abra la consola de CloudTrail.
2. Elija Historial de eventos.
3. Elija los eventos que desee consultar y, a continuación, seleccione Comparar detalles de los eventos.

Ejemplo de registro de seguimiento para **`CreateEnvironment`**

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de registros en un bucket de Amazon S3 que especifique.

Los archivos de registro de CloudTrail pueden contener una o varias entradas de registro. Los eventos son solicitudes específicas realizadas desde una fuente y contienen información sobre la acción solicitada, la fecha y la hora de la acción y los parámetros de la solicitud. Los archivos de registro de CloudTrail no son un seguimiento de pila ordenado de las llamadas a la API públicas y no aparecen en un orden específico. El siguiente ejemplo muestra una entrada de registro

para la acción `CreateEnvironment` que es denegada por falta de permisos. Los valores de `AirflowConfigurationOptions` se han ocultado por motivos de privacidad.


```
{
  "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"
}
```

Siguientes pasos

- Aprenda a configurar otros servicios de AWS para los datos de eventos recopilados en los registros de CloudTrail en Servicios e integración de CloudTrail compatibles.
- Descubra cómo recibir una notificación cada vez que CloudTrail publique nuevos archivos de registro en un bucket de Amazon S3 en Configuración de las notificaciones de Amazon SNS para CloudTrail.

Visualización de registros en Amazon CloudWatch

Amazon MWAA tiene la capacidad de enviar registros de Apache Airflow a Amazon CloudWatch. Puede visualizar los registros de varios entornos desde una única ubicación para así identificar fácilmente los retrasos en las tareas o los errores en el flujo de trabajo de Apache Airflow sin necesidad de usar otras herramientas externas. Los registros de Apache Airflow deben estar habilitados en la consola de Amazon Managed Workflows para Apache Airflow para poder visualizar en CloudWatch los registros del procesamiento de DAG, las tareas, el servidor web y los procesos de trabajo de Apache Airflow.

Contenido

Precios

- Se aplican los cargos estándar de los Registros de CloudWatch. Para obtener más información, consulte los precios de CloudWatch.

Antes de empezar

- Debe tener un rol que tenga acceso a los registros en CloudWatch. Para obtener más información, consulta Acceso a un entorno de Amazon MWAA.

Tipos de registro

Amazon MWAA crea un grupo de registro para cada opción de registro de Airflow que se habilite y envía los registros a los grupos de registros de CloudWatch asociados a los entornos. Se asigna un nombre con el formato `YourEnvironmentName-`**`LogType`** a los grupos de registro. Por ejemplo, si su entorno se denomina "`Airflow-v202-Public`", los registros de las tareas de Apache Airflow se enviarán a `Airflow-v202-Public-`**`Task`**.

Tipo de registro	Descripción
`YourEnvironmentName-` **`DAGProcessing`**	Registros del administrador del procesador de DAG (la parte del programador encargada de procesar los archivos DAG).
`YourEnvironmentName-` **`Scheduler`**	Registros que genera el programador de Airflow.
`YourEnvironmentName-` **`Task`**	Registros de las tareas que genera un DAG.
`YourEnvironmentName-` **`WebServer`**	Registros que genera la interfaz web de Airflow.

Tipo de registro	Descripción
`YourEnvironmentName-` **`Worker`**	Registros que se generan como parte del flujo de trabajo y de la ejecución de los DAG.

Habilitación de los registros de Apache Airflow

Puede habilitar los registros de Apache Airflow en los niveles `INFO`, `WARNING`, `ERROR` o `CRITICAL`. A elegir un nivel de registro, Amazon MWAA envía los registros correspondientes a ese nivel y a todos los niveles de gravedad superiores. Por ejemplo, si habilita los registros en el nivel `INFO`, Amazon MWAA envía los registros del nivel `INFO`, así como los registros de los niveles `WARNING`, `ERROR` y `CRITICAL` a los Registros de CloudWatch.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija Editar.
4. Elija Siguiente.
5. Elija una o varias de las siguientes opciones de registro:
    a. Elija el Grupo de registro del programador de Airflow en el panel de Monitorización.
    b. Elija el grupo de registro del servidor web de Airflow en el panel de monitorización.
    c. Elija el grupo de registro de los procesos de trabajo de Airflow en el panel de monitorización.
    d. Elija el grupo de registro del procesamiento de los DAG de Airflow en el panel de monitorización.
    e. Elija el grupo de registro de las tareas de Airflow en el panel monitorización.
    f. Elija el nivel de registro en el nivel de registro.
6. Elija Siguiente.
7. Seleccione Save.

Visualización de registros de Apache Airflow

En la siguiente sección, se describe cómo ver los registros de Apache Airflow en la consola de CloudWatch.

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija un grupo de registro en el panel de monitorización.
4. Elija un registro en el flujo de registro.

Ejemplos de registros del programador

Consulte los registros de Apache Airflow correspondientes al programador encargado de organizar sus flujos de trabajo y analizar su carpeta de `dags`. En los siguientes pasos, se describe cómo abrir el grupo de registros del programador en la consola de Amazon MWAA y cómo ver los registros de Apache Airflow en la consola de registros de CloudWatch.

Para acceder a los registros de un **`requirements.txt`**

1. Abra la página Entornos en la consola de Amazon MWAA.
2. Seleccione un entorno.
3. Elija el Grupo de registro del programador de Airflow en el panel de Monitorización.
4. Seleccione el registro `requirements_install_ip` en los flujos de registro.
5. Consulte la lista de paquetes que se hayan instalado en el entorno en `/usr/local/airflow/.local/bin`. Por ejemplo:

```
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. Consulte la lista de paquetes y compruebe si se produjo algún error en alguno de ellos durante la instalación. Si algo salió mal, es posible que aparezca un error similar al siguiente:

```
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))
```

Siguientes pasos

- Aprenda a configurar una alarma de CloudWatch en Uso de las alarmas de Amazon CloudWatch.
- Aprenda a crear un panel de CloudWatch en Uso de paneles de Amazon CloudWatch.

Monitorización de paneles y alarmas en Amazon MWAA

Se puede crear un panel personalizado en Amazon CloudWatch y añadir alarmas para métricas concretas a fin de monitorizar el estado de entornos de Amazon Managed Workflows para Apache Airflow. Cuando una alarma está en un panel, se vuelve de color rojo cuando está en el estado `ALARM`, lo que facilita la monitorización del estado de un entorno de Amazon MWAA de forma proactiva.

Apache Airflow muestra las métricas de varios procesos, entre ellos, la cantidad de procesos de los DAG, el tamaño de la DagBag, las tareas en curso, las que se realizaron correctamente y aquellas que tuvieron errores. Al crear un entorno, Airflow está configurado para enviar las métricas de los entornos de Amazon MWAA a CloudWatch de manera automática. En esta página se describe cómo crear un panel de estado de las métricas de Airflow en CloudWatch para los entornos de Amazon MWAA.

Contenido

Métricas

Puede crear paneles y alarmas personalizados para cualquiera de las métricas disponibles en su versión de Apache Airflow. Cada métrica corresponde a un indicador clave de rendimiento de Apache Airflow. Para acceder a una lista de métricas, consulte:

- Métricas del entorno de Apache Airflow en CloudWatch

Información general sobre los estados de las alarmas

Una alarma de métrica tiene los siguientes estados posibles:

- `OK`: la métrica o expresión está dentro del umbral definido.
- `ALARM`: la métrica o expresión está fuera del umbral definido.
- `INSUFFICIENT_DATA`: la alarma acaba de iniciarse, la métrica no está disponible o no hay suficientes datos disponibles en la métrica para determinar el estado de la alarma.

Ejemplos de paneles y alarmas personalizados

Puede crear un panel de monitorización personalizado que muestre gráficos de las métricas seleccionadas para su entorno de Amazon MWAA.

Acerca de las métricas

En la siguiente lista se describen cada una de las métricas que se han creado en el panel personalizado mediante el tutorial y las plantillas de esta sección.

- QueuedTasks: cantidad de tareas en el estado de en cola. Se corresponde con la métrica `executor.queued_tasks` de Apache Airflow.
- TaskSpending: cantidad de tareas pendientes en el ejecutor. Se corresponde con la métrica `scheduler.tasks.pending` de Apache Airflow.

  > **Note**
  >
  > No se aplica a Apache Airflow v2.2 ni versiones posteriores.

- RunningTasks: cantidad de tareas en curso en el ejecutor. Se corresponde con la métrica `executor.running_tasks` de Apache Airflow.

- SchedulerHeartbeat: cantidad de comprobaciones que Apache Airflow realiza en el trabajo del programador. Se corresponde con la métrica `scheduler_heartbeat` de Apache Airflow.
- TotalParseTime: cantidad de segundos que se tarda en analizar e importar todos los archivos DAG una vez. Se corresponde con la métrica `dag_processing.total_parse_time` de Apache Airflow.

Acerca del panel

En la siguiente imagen, se muestra el panel de monitorización que se ha creado al definir el tutorial y la plantilla de esta sección.

Tutoriales de uso de AWS

Puede seguir el tutorial de AWS para crear de forma automática un panel de estado para cualquier entorno de Amazon MWAA que esté implementado actualmente. También puede ver cómo crear alarmas de CloudWatch para procesos de trabajo en mal estado y fallos de latidos del programador en todos los entornos de Amazon MWAA.

- Panel de automatización de Amazon MWAA en CloudWatch

Uso de CloudFormation

Puede utilizar la plantilla de CloudFormation que encontrará en esta sección para crear un panel de monitorización en CloudWatch y para añadir después alarmas en la consola de CloudWatch con el fin de recibir una notificación cada vez que una métrica supere un umbral determinado. Para crear la pila usando esta definición de plantilla, consulte cómo crear una pila en la consola de la CloudFormation. Para agregar una alarma al panel, consulte cómo usar las alarmas.

```
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
                    }
                }
            ]
        }'
```

Eliminación de métricas y paneles

Si elimina un entorno de Amazon MWAA, se eliminará también el panel correspondiente. Las métricas de CloudWatch se almacenan durante quince (15) meses y se no pueden eliminar. La consola de CloudWatch limita la búsqueda de métricas a dos (2) semanas después de la última incorporación de una métrica para garantizar que se muestren las instancias más actualizadas de su entorno de Amazon MWAA. Para obtener más información, consulte las preguntas frecuentes de Amazon CloudWatch.

Siguientes pasos

- Aprenda a crear un DAG que consulte la base de datos de metadatos de Amazon Aurora PostgreSQL de su entorno y publique métricas personalizadas en CloudWatch en Uso de un DAG para escribir métricas personalizadas en CloudWatch.

Métricas del entorno de Apache Airflow en CloudWatch

Apache Airflow v2 y v3 ya están configurados para recopilar y enviar métricas de un entorno de Amazon Managed Workflows para Apache Airflow a Amazon CloudWatch con StatsD. Encontrará la lista completa de métricas que envía Apache Airflow en la página Metrics de la guía de referencia de Apache Airflow. En esta página, se describen las métricas de Apache Airflow disponibles en CloudWatch y cómo acceder a ellas desde la consola de CloudWatch.

Contenido

Términos

Espacio de nombres

Los espacios de nombres son contenedores para métricas de CloudWatch de servicios de AWS. En el caso de Amazon MWAA, el espacio de nombres es AmazonMWAA.

Métricas de CloudWatch

Las métricas representan una serie de puntos de datos ordenados por tiempo que se publican en CloudWatch.

Métricas de Apache Airflow

Las métricas que son específicas de Apache Airflow.

Dimensión

Una dimensión es un par de nombre-valor que forma parte de la identidad de una métrica.

Unidad

Las estadísticas tienen unidades de medida. En el caso de Amazon MWAA, las unidades son recuento, segundos y milisegundos. Además, en Amazon MWAA, las unidades se establecen basándose en las unidades de las métricas de Airflow originales.

Dimensiones

En esta sección, se describen las diferentes agrupaciones de dimensiones de CloudWatch para las métricas de Apache Airflow en CloudWatch.

Dimensión	Descripción			
DAG	Indica un nombre específico para el DAG de Apache Airflow.			
Nombre de archivo del DAG	Indica un nombre de archivo específico para el DAG de Apache Airflow.			
Función	Esta dimensión se utiliza para mejorar la			

Dimensión	Descripción
	agrupación de métricas en CloudWatch.
Trabajo	Indica un trabajo de Apache Airflow ejecutado por el programador. Siempre tiene un valor de `Job`.
Operador	Indica un operador específico de Apache Airflow.
Grupo	Indica un grupo de procesos de trabajo específico de Apache Airflow.
Tarea	Indica una tarea específica de Apache Airflow.
HostName	Indica el nombre de host de un proceso específico que se está ejecutando en Apache Airflow.

Acceso a las métricas a través de la consola de CloudWatch

En esta sección se describe cómo acceder a las métricas de rendimiento de CloudWatch para un DAG específico.

Consulta de métricas de rendimiento de una dimensión

1. Abra la página de métricas en la consola de CloudWatch.
2. Seleccione su Región de AWS.
3. Elija el espacio de nombres AmazonMWAA.

4. En la pestaña Todas las métricas, elija una dimensión. Por ejemplo, DAG, Entorno.
5. Elija una métrica de CloudWatch para la dimensión. Por ejemplo, TaskInstanceSuccesses o TaskInstanceDuration. Elija Representar gráficamente todos los resultados de la búsqueda.
6. Elija la pestaña Graphed metrics (Métricas gráficas) para ver las estadísticas de rendimiento de las métricas de Apache Airflow. Por ejemplo, DAG, Entorno, Tarea.

Métricas de Apache Airflow disponibles en CloudWatch

En esta sección se describen las métricas y las dimensiones de Apache Airflow que se envían a CloudWatch.

Contadores de Apache Airflow

Las métricas de Apache Airflow que figuran en esta sección contienen datos sobre los contadores de Apache Airflow.

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
SLAMissed **Note** Disponible únicamente para Apache Airflow v2.4.3 a v2.10.3.	sla_missed	Recuento	Función, Programador	
FailedSLACallback **Note** Disponible únicamente para Apache Airflow v2.4.3 a v2.10.3.	sla_callback_notification_failure	Recuento	Función, Programador	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
Actualizaciones **Note** Disponible para Apache Airflow v2.6.3 y versiones posteriores.	dataset.updates	Recuento	Función, Programador	
Orphaned **Note** Disponible para Apache Airflow v2.6.3 y versiones posteriores.	dataset.orphaned	Recuento	Función, Programador	
FailedCeleryTaskExecution **Note** Disponible para Apache Airflow v2.4.3 y versiones posteriores.	celery.execute_command.failure	Recuento	Función, Celery	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
FilePathQueueUpdateCount Note Disponible para Apache Airflow v2.6.3 y versiones posteriores.	dag_processing.file_path_queue_update_count	Recuento	Función, Programador	
CriticalSectionBusy	scheduler.critical_section_busy	Recuento	Función, Programador	
DagBagSize	dagbag_size	Recuento	Función, Procesamiento de DAG	
DagCallbackExceptions	dag.callback_exceptions	Recuento	DAG, Todos	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
FailedSLAEmailAttempts **Note** Disponible para Apache Airflow v3.0.6 y versiones posteriores.	sla_email_notification_failure	Recuento	Función, Programador	
TaskInstanceFinished	ti.finish.{dag_id}.{task_id}.{state}	Recuento	DAG, {dag_id} Tarea, {task_id} Estado, {state}	
JobEnd	{job_name}_end	Recuento	Trabajo, {job_name}	
JobHeartbeatFailure	{job_name}_heartbeat_failure	Recuento	Trabajo, {job_name}	
JobStart	{job_name}_start	Recuento	Trabajo, {job_name}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión
ManagerStalls	dag_processing.manager_stalls	Recuento	Función, Procesamiento de DAG
OperatorFailures	operator_failures_{operator_name}	Recuento	Operador, {operator_name}
OperatorSuccesses	operator_successes_{operator_name}	Recuento	Operador, {operator_name}
OtherCallbackCount **Note** Disponible en Apache Airflow v2.6.3 y versiones posteriores.	dag_processing.other_callback_count	Recuento	Función, Programador
Processes	dag_processing.processes	Recuento	Función, Procesamiento de DAG
SchedulerHeartbeat	scheduler_heartbeat	Recuento	Función, Programador

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
StartedTaskInstances	ti.start.{dag_id}.{task_id}	Recuento	DAG, Todos Tarea, Todas	
SlaCallbackCount	dag_processing.sla_callback_count **Note** Disponible para Apache Airflow v2.6.3 y versiones posteriores.	Recuento	Función, Programador	
TasksKilledExternally	scheduler.tasks.killed_externally	Recuento	Función, Programador	
TaskTimeoutError	celery.task_timeout_error	Recuento	Función, Celery	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TaskInstanceCreatedUsingOperator	task_instance_created-{operator_name}	Recuento	Operador, {operator_name}	
TaskInstancePreviouslySucceeded	previously_succeeded	Recuento	DAG, Todos Tarea, Todas	
TaskInstanceFailures	ti_failures	Recuento	DAG, Todos Tarea, Todas	
TaskInstanceSuccesses	ti_successes	Recuento	DAG, Todos Tarea, Todas	
TaskRemovedFromDAG	task_removed_from_dag.{dag_id}	Recuento	DAG, {dag_id}	
TaskRestoredToDAG	task_restored_to_dag.{dag_id}	Recuento	DAG, {dag_id}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TriggersSucceeded **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	triggers.succeeded	Recuento	Función, Disparador	
TriggersFailed **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	triggers.failed	Recuento	Función, Disparador	
TriggersBlockedMainThread **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	triggers.blocked_main_thread	Recuento	Función, Disparador	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TriggerHeartbeat **Note** Disponible para Apache Airflow v2.8.1 y versiones posteriores.	triggerer_heartbeat	Recuento	Función, Disparador	
TaskInstanceCreatedUsingOperator	airflow.task_instance_created_`{operator_name}` **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	Recuento	Operador, `{operator_name}`	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
ZombiesKilled	zombies_killed	Recuento	DAG, Todos Tarea, Todas	

Indicadores de Apache Airflow

Las métricas de Apache Airflow que figuran en esta sección contienen datos sobre los indicadores de Apache Airflow.

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
DAGFileRefreshError	dag_file_refresh_error	Recuento	Función, Procesamiento de DAG	
ImportErrors	dag_processing.import_errors	Recuento	Función, Procesamiento de DAG	
Exception Failures	smart_sensor_operator.exception_failures	Recuento	Función, Operador de sensores inteligentes	
ExecutedTasks	smart_sensor_operator.executed_tasks	Recuento	Función, Operador de sensores inteligentes	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión
InfraFailures	smart_sensor_operator.infra_failures	Recuento	Función, Operador de sensores inteligentes
LoadedTasks	smart_sensor_operator.loaded_tasks	Recuento	Función, Operador de sensores inteligentes
TotalParseTime	dag_processing.total_parse_time	Segundos	Función, Procesamiento de DAG
TriggeredDagRuns **Note** Disponible en Apache Airflow v2.6.3 y versiones posteriores.	dataset.triggered_dagruns	Recuento	Función, Programador

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TriggersRunning Note Disponible en Apache Airflow v2.7.2 y versiones posteriores.	triggers.running.*{hostname}*	Recuento	Función, Disparador Nombre de host, *{hostname}*	
PoolDeferredSlots Note Disponible en Apache Airflow v2.7.2 y versiones posteriores.	pool.deferred_slots.{pool_name}	Recuento	Grupo, {pool_name}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
DAGFilePr ocessingL astRunSec ondsAgo	dag_processing.las t_run.seconds_ago. {dag_filename}	Segundos	Nombre de archivo del DAG, {dag_file name}	
OpenSlots	executor.open_slots	Recuento	Función, Ejecutor	
OrphanedT asksAdopted	scheduler.orphaned _tasks.adopted	Recuento	Función, Programador	
OrphanedT asksCleared	scheduler.orphaned _tasks.cleared	Recuento	Función, Programador	
PokedExce ptions	smart_sensor_opera tor.poked_exception	Recuento	Función, Operador de sensores inteligentes	
PokedSuccess	smart_sensor_opera tor.poked_success	Recuento	Función, Operador de sensores inteligentes	
PokedTasks	smart_sensor_opera tor.poked_tasks	Recuento	Función, Operador de sensores inteligentes	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión
PoolFailures	pool.open_slots.{pool_name}	Recuento	Grupo, {pool_name}
PoolStarvingTasks	pool.starving_tasks.{pool_name}	Recuento	Grupo, {pool_name}
PoolOpenSlots	pool.open_slots.{pool_name}	Recuento	Grupo, {pool_name}
PoolQueuedSlots	pool.queued_slots.{pool_name}	Recuento	Grupo, {pool_name}
PoolRunningSlots	pool.running_slots.{pool_name}	Recuento	Grupo, {pool_name}
ProcessorTimeouts	dag_processing.processor_timeouts	Recuento	Función, Procesamiento de DAG
QueuedTasks	executor.queued_tasks	Recuento	Función, Ejecutor
RunningTasks	executor.running_tasks	Recuento	Función, Ejecutor
TasksExecutable	scheduler.tasks.executable	Recuento	Función, Programador

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TasksPending **Note** No se aplica a Apache Airflow v2.2 ni versiones posteriores.	scheduler.tasks.pending	Recuento	Función, Programador	
TasksRunning	scheduler.tasks.running	Recuento	Función, Programador	
TasksStarving	scheduler.tasks.starving	Recuento	Función, Programador	
TasksWithoutDagRun	scheduler.tasks.without_dagrun	Recuento	Función, Programador	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión
DAGFileProcessingLastNumOfDbQueries Note Disponible en Apache Airflow v2.10.1 y versiones posteriores.	dag_processing.last_num_of_db_queries.{dag_filename}	Recuento	Nombre de archivo del DAG, {dag_filename}
PoolScheduledSlots Note Disponible en Apache Airflow v2.10.1 y versiones posteriores.	pool.scheduled_slots.{pool_name}	Recuento	Grupo, {pool_name}

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión
TaskCpuUsage Note Disponible en Apache Airflow v2.10.1 y versiones posteriores.	cpu.usage.{dag_id}.{task_id}	Porcentaje	DAG, {dag_id} Tarea, {task_id}
TaskMemoryUsage Note Disponible en Apache Airflow v2.10.1 y versiones posteriores.	mem.usage.{dag_id}.{task_id}	Porcentaje	DAG, {dag_id} Tarea, {task_id}

Temporizadores de Apache Airflow

Las métricas de Apache Airflow que figuran en esta sección contienen datos sobre los temporizadores de Apache Airflow.

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
CollectDBDags	colect_db_dags	Milisegundos	Función, Procesamiento de DAG	
CriticalSectionDuration	scheduler.critical_section_duration	Milisegundos	Función, Programador	
CriticalSectionQueryDuration Note Disponible para Apache Airflow v2.5.1 y versiones posteriores.	scheduler.critical_section_query_duration	Milisegundos	Función, Programador	
DAGDependencyCheck	dagrun.dependency-check.{dag_id}	Milisegundos	DAG, {dag_id}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
DAGDurationFailed	dagrun.duration.failed.{dag_id}	Milisegundos	DAG, {dag_id}	
DAGDurationSuccess	dagrun.duration.success.{dag_id}	Milisegundos	DAG, {dag_id}	
DAGFileProcessingLastDuration	dag_processing.last_duration.{dag_filename}	Segundos	Nombre de archivo del DAG, {dag_filename}	
DAGScheduleDelay	dagrun.schedule_delay.{dag_id}	Milisegundos	DAG, {dag_id}	
FirstTaskSchedulingDelay	dagrun.{dag_id}.first_task_scheduling_delay	Milisegundos	DAG, {dag_id}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
Scheduler LoopDuration Note Disponible para Apache Airflow v2.5.1 y versiones posteriores.	scheduler.scheduler_loop_duration	Milisegundos	Función, Programador	
TaskInstanceDuration	dag.{dag_id}.{task_id}.duration	Milisegundos	DAG, {dag_id} Tarea, {task_id}	

Métrica de CloudWatch	Métrica de Apache Airflow	Unidad	Dimensión	
TaskInstanceQueuedDuration	dag.{dag_id}.{t .queued_duration **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	Milisegundos	DAG, {dag_id} Tarea, {task_id}	
TaskInstanceScheduledDuration **Note** Disponible para Apache Airflow v2.7.2 y versiones posteriores.	dag.{dag_id}.{t .scheduled_duration	Milisegundos	DAG, {dag_id} Tarea, {task_id}	

Selección de las métricas se comunican

Puede escoger qué métricas de Apache Airflow quiere que se envíen a CloudWatch o cuáles quiere que Airflow bloquee mediante las siguientes opciones de configuración de Amazon MWAA:

- **`metrics.metrics_allow_list`** — una lista de prefijos separados por comas que puede usar para elegir las métricas que su entorno debe enviar a CloudWatch. Utilice esta opción si no quiere que Apache Airflow envíe todas las métricas disponibles y el subconjunto de elementos que sí quiere enviar. Por ejemplo, `scheduler,executor,dagrun`.
- **`metrics.metrics_block_list`** — una lista de prefijos separados por comas para filtrar las métricas que comienzan con los elementos de la lista. Por ejemplo, `scheduler,executor,dagrun`.

Si configura ambas opciones, la `metrics.metrics_allow_list` y `metrics.metrics_block_list`, Apache Airflow ignorará la `metrics.metrics_block_list`. Si configura la `metrics.metrics_block_list` pero no `metrics.metrics_allow_list`, Apache Airflow filtrará los elementos que haya especificado en la `metrics.metrics_block_list`.

> **Note**
>
> Las opciones de configuración `metrics.metrics_allow_list` y `metrics.metrics_block_list` solo se aplican a Apache Airflow v2.6.3 y versiones posteriores. Para la versión anterior de Apache Airflow, utilice `metrics.statsd_allow_list` y `metrics.statsd_block_list` en su lugar.

Siguientes pasos

- Conozca la operación de la API de Amazon MWAA que se utiliza para publicar las métricas del estado del entorno en PublishMetrics.

Métricas de contenedores, colas y bases de datos para Amazon MWAA

Además de las métricas de Apache Airflow, también puede monitorizar los componentes subyacentes de los entornos de Amazon Managed Workflows para Apache Airflow con Amazon

CloudWatch, que recopila datos sin tratar y los convierte en métricas legibles prácticamente en tiempo real. Gracias a estas métricas del entornos, dispondrá de una mayor visibilidad de los indicadores clave de rendimiento que le resultarán útiles para dimensionar sus entornos de forma adecuada y resolver problemas relacionados con los flujos de trabajo. Estas métricas se aplican a todas las versiones de Apache Airflow compatibles con Amazon MWAA.

Amazon MWAA proporcionará métricas sobre el uso de la CPU y la memoria para cada contenedor de Amazon Elastic Container Service (Amazon ECS) e instancia de Amazon Aurora PostgreSQL, así como métricas de Amazon Simple Queue Service (Amazon SQS) para el número de mensajes y la antigüedad del mensaje más antiguo, métricas de Amazon Relational Database Service (Amazon RDS) para las conexiones de base de datos, la profundidad de la cola de disco, las operaciones de escritura, la latencia y el rendimiento, y métricas de Amazon RDS Proxy. Además, también se incluirá la cantidad de procesos de trabajo base, los procesos de trabajo adicionales, los programadores y los servidores web.

Estas estadísticas se conservarán durante 15 meses, lo que le permitirá acceder a información histórica y dispondrá de una mejor perspectiva acerca de por qué un programa está fallando y podrá solucionar problemas subyacentes. También puede establecer alarmas para supervisar determinados umbrales y enviar notificaciones o tomar medidas cuando se cumplan dichos umbrales. Para obtener más información, consulte la Guía del usuario de Amazon CloudWatch.

Temas

Términos

Espacio de nombres

Los espacios de nombres son contenedores para métricas de CloudWatch de servicios de AWS. El espacio de nombres para Amazon MWAA es `AWS/MWAA`.

Métricas de CloudWatch

Las métricas representan una serie de puntos de datos ordenados por tiempo que se publican en CloudWatch.

Dimensión

Una dimensión es un par de nombre-valor que forma parte de la identidad de una métrica.

Unidad

Las estadísticas tienen unidades de medida. En el caso de Amazon MWAA, la unidad es recuento.

Dimensiones

En esta sección, se describen las diferentes agrupaciones de dimensiones de CloudWatch para las métricas de Amazon MWAA.

Dimensión	Descripción
Clúster	Métricas de los tres contenedores de Amazon ECS, como mínimo, que un entorno de Amazon MWAA utiliza para ejecutar los componentes de Apache Airflow: programador, proceso de trabajo y servidor web.
Cola	Métricas de las colas de Amazon SQS que separan el programador de los procesos de trabajo. Cuando los procesos de trabajo leen un mensaje, se considera que están en tránsito y, por tanto, no está disponible para otros procesos de trabajo. Los mensajes pasan a estar disponibles para que los lean otros procesos de trabajo si no se borran antes del tiempo límite de visibilidad de 12 horas.
Database	Métricas de los clústeres de Aurora que utiliza Amazon MWAA. Esto incluye métricas de la instancia de base de datos principal y una réplica de lectura compatibles con las operaciones de lectura. Amazon MWAA publica

Dimensión	Descripción
	métricas de base de datos para las instancias de LECTOR y ESCRITOR.

Acceso a las métricas a través de la consola de CloudWatch

En esta sección se describe cómo acceder a las métricas de Amazon MWAA en CloudWatch.

Consulta de métricas de rendimiento de una dimensión

1. Abra la página de métricas en la consola de CloudWatch.
2. Seleccione su Región de AWS.
3. Elija el espacio de nombres AWS/MWAA.
4. En la pestaña Todas las métricas, elija una dimensión. Por ejemplo, Clúster.
5. Elija una métrica de CloudWatch para la dimensión. Por ejemplo, NumSchedulers o CPUUtilization. A continuación, elija Representar gráficamente todos los resultados de la búsqueda.
6. Elija la pestaña Graphed metrics (Métricas gráficas) para ver las métricas de rendimiento.

Lista de métricas

En las tablas siguientes se enumeran las métricas de los servicios de clúster, cola y base de datos de Amazon MWAA. Para ver las descripciones de las métricas que se envían directamente desde Amazon ECS, Amazon SQS o Amazon RDS, haga clic en el enlace a la documentación correspondiente.

Temas

- Métricas de clúster
- Métricas de la base de datos
- Métricas de cola
- Métricas del Equilibrador de carga de aplicación

Métricas de clúster

Las métricas siguientes se aplican a todos los programadores, procesos de trabajo base, procesos de trabajo adicionales y servidores web. Para obtener más información y leer las descripciones de cada métrica de clúster, consulte las métricas y dimensiones disponibles en la guía para desarrolladores de Amazon ECS.

Espacio de nombres	Métrica	Unidad
`AWS/MWAA`	`CPUUtilization`	Porcentaje
`AWS/MWAA`	`MemoryUtilization`	Porcentaje

Evaluación de la cantidad de contenedores adicionales para procesos de trabajo y servidores web

Puede usar las métricas de los componentes que se indican en la dimensión Clúster, como se describe en el siguiente procedimiento, para evaluar cuántos procesos de trabajo adicionales o servidores web usa un entorno en un momento dado. Para ello, puede diagramar la métrica CPUUtilization o MemoryUtilization y establecer el tipo de estadística en Número de muestras. El valor resultante será el número total de tareas `RUNNING` del componente `AdditionalWorker`. Comprender la cantidad de instancias de procesos de trabajo adicionales que usa su entorno puede ser útil para evaluar la forma en que este se escala y optimizar la cantidad de procesos de trabajo adicionales.

Workers

Pasos para evaluar la cantidad de procesos de trabajo adicionales que utilizan la Consola de administración de AWS

1. Elija el espacio de nombres AWS/MWAA.
2. En la pestaña Todas las métricas, elija la dimensión Clúster.
3. En la dimensión Clúster, en AdditionalWorker, escoja la métrica CPUUtilization o MemoryUtilization.
4. En la pestaña Métricas diagramadas, cambie Periodo a 1 minuto y Estadística a Número de muestras.

webservers

Pasos para evaluar la cantidad de servidores web adicionales que usan la Consola de administración de AWS

1. Elija el espacio de nombres AWS/MWAA.
2. En la pestaña Todas las métricas, elija la dimensión Clúster.
3. En la dimensión Clúster, en AdditionalWebServers, seleccione la métrica CPUUtilization o MemoryUtilization.
4. En la pestaña Métricas diagramadas, cambie Periodo a 1 minuto y Estadística a Número de muestras.

Para más información, consulte el recuento de tareas del servicio `RUNNING` en la guía para desarrolladores de Amazon Elastic Container Service.

Métricas de la base de datos

Las métricas siguientes se aplican a todas las instancias de bases de datos asociadas al entorno de Amazon MWAA.

Espacio de nombres	Métrica	Unidad
`AWS/MWAA`	`CPUUtilization`	Porcentaje
`AWS/MWAA`	`DatabaseConnections`	Recuento
`AWS/MWAA`	`DiskQueueDepth`	Recuento
`AWS/MWAA`	`FreeableMemory`	Bytes
`AWS/MWAA`	`VolumeWriteIOPS`	Recuento cada 5 minutos
`AWS/MWAA`	`WriteIOPS`	

Espacio de nombres	Métrica	Unidad
		Recuento por segundo
AWS/MWAA	WriteLatency	Segundos
AWS/MWAA	WriteThroughput	Bytes por segundo

Métricas de cola

Para obtener más información sobre las unidades y descripciones de la cola, consulte las métricas de CloudWatch disponibles para Amazon SQS en la guía para desarrolladores de Amazon Simple Queue Service.

Espacio de nombres	Métrica	Unidad
AWS/MWAA	ApproximateAgeOfOldestTask	Segundos
AWS/MWAA	RunningTasks	Recuento
AWS/MWAA	QueuedTasks	Recuento

Métricas del Equilibrador de carga de aplicación

Las métricas del equilibrador de carga de aplicación se aplican a los servidores web que se ejecutan en su entorno. Amazon MWAA utiliza estas métricas para escalar sus servidores web en función de la cantidad de tráfico. Para obtener más información sobre las unidades y descripciones de las siguientes métricas del equilibrador de carga, consulte las métricas de CloudWatch para el equilibrador de carga de aplicación en la guía del usuario de equilibradores de carga de aplicación.

Espacio de nombres	Métrica	Unidad
AWS/MWAA	ActiveConnectionCount	Recuento

Seguridad en Amazon Managed Workflows para Apache Airflow

La seguridad en la nube de AWS es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted (el cliente). En el modelo de responsabilidad compartida, se la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWSProgramas de conformidad de . Para obtener más información sobre los programas de conformidad que se aplican a Amazon MWAA, consulte los servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad se determina según el servicio de AWSque utiliza. También eres responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

La documentación explica cómo se debe aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon Managed Workflows para Apache Airflow. Úselo para configurar Amazon MWAA y lograr sus objetivos de seguridad y conformidad. También puede aprender a usar otros servicios de AWS que ayudan a monitorear y proteger los recursos de Amazon MWAA.

En esta sección:

Protección de datos en Amazon Managed Workflows para Apache Airflow

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en Amazon Managed Workflows para Apache Airflow. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye la configuración de seguridad y las tareas de administración para el Servicios de AWS que usa. Para obtener más información sobre la privacidad de los datos, consulte las preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de Cuenta de AWS y configurar cuentas de usuario individuales con AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de las siguientes maneras:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. Recomendamos TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.

Recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, direcciones de email de sus clientes, en etiquetas o en los campos de formato libre, como el campo Name (Nombre). Esto incluye los casos en los que trabaje con Amazon MWAA u otros servicios de AWS mediante la consola, la API, la AWS CLI o los SDK de AWS. Los datos que ingresa en etiquetas o campos de formato libre usados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado en Amazon MWAA

En los siguientes temas se describe la manera en que Amazon MWAA protege los datos en reposo y los datos en tránsito. Utilice esta información para conocer cómo Amazon MWAA se integra con AWS KMS para cifrar los datos en reposo y cómo se cifran los datos en tránsito mediante el protocolo de seguridad de la capa de transporte (TLS).

Temas

Cifrado en reposo

En Amazon MWAA, los datos en reposo son datos que el servicio guarda en medios persistentes.

Al crear un entorno, puede usar una clave propiedad de AWS para el cifrado de los datos en reposo o, si lo desea, proporcionar una clave administrada por el cliente para otro cifrado. Si opta por usar una clave de KMS administrada por el cliente, tenga en cuenta que deberá estar en la misma cuenta que los demás recursos y servicios de AWS que use en el entorno.

Para usar una clave de KMS administrada por el cliente, debe asociar a su política de claves la instrucción de política necesaria para el acceso de CloudWatch. Si usa una clave de KMS administrada por el cliente para su entorno, Amazon MWAA asocia cuatro concesiones en su nombre. Para obtener más información sobre las concesiones que Amazon MWAA asocia a una clave de KMS administrada por el cliente, consulte las claves administradas por el cliente para el cifrado de datos.

Si no especifica una clave de KMS administrada por el cliente, Amazon MWAA usará de forma predeterminada una clave de KMS propiedad de AWS para cifrar y descifrar los datos. Recomendamos utilizar una clave de KMS propiedad de AWS para administrar el cifrado de datos en Amazon MWAA.

> **Note**
>
> En Amazon MWAA, se paga el almacenamiento y el uso tanto de las claves de KMS propiedad de AWS como de las administradas por el cliente. Para obtener más información, consulte los precios de AWS KMS.

Artefactos de cifrado

Al crear su entorno de Amazon MWAA, deberá especificar los artefactos de cifrado que usará para el cifrado en reposo especificando una clave propiedad de AWS o una clave administrada por el cliente. Amazon MWAA se encargará de añadir la concesiones necesarias a la clave especificada.

Amazon S3: los datos de Amazon S3 se cifran a nivel de objeto mediante el cifrado del servidor (SSE). En Amazon S3, el cifrado y el descifrado se llevan a cabo en el bucket de Amazon S3 en el cual estén almacenados el código de sus DAG y los archivos auxiliares. Los objetos se cifran al cargarlos en Amazon S3 y se descifran al descargarlos de su entorno de Amazon MWAA. Si usa una clave de KMS administrada por el cliente, Amazon MWAA la usará para leer y descifrar los datos de su bucket de Amazon S3 de forma predeterminada.

Registros de CloudWatch: si usa una clave de KMS propiedad de AWS, los registros de Apache Airflow que se envíen a los registros de CloudWatch se cifrarán mediante el cifrado del servidor (SSE) con la clave de KMS propiedad de AWS de los registros de CloudWatch. Si usa una clave de KMS administrada por el cliente, deberá agregar una política de claves a su clave de KMS para permitir que los registros de CloudWatch puedan usarla.

Amazon SQS: Amazon MWAA creará una cola de Amazon SQS para su entorno. Amazon MWAA cifrará los datos que entren y salgan de la cola mediante el SSE con una clave de KMS propiedad de AWS o una clave de KMS administrada por el cliente que usted especifique. Deberá agregar los permisos de Amazon SQS a su rol de ejecución, independientemente de si usa una clave de KMS propiedad de AWS o una clave de KMS administrada por el cliente.

Aurora PostgreSQL: Amazon MWAA creará un clúster de PostgreSQL para su entorno. Aurora PostgreSQL cifrará el contenido mediante el SSE con una clave de KMS propiedad de AWS o una clave de KMS administrada por el cliente. Si usa una clave de KMS administrada por el cliente, Amazon RDS agregará, como mínimo, dos concesiones a la clave: una para el clúster y otra para la instancia de base de datos. Amazon RDS puede crear más concesiones si decide usar la clave de KMS administrada por el cliente en varios entornos. Para obtener más información, consulte la protección de datos en Amazon RDS.

Cifrado en tránsito

Se denomina “datos en tránsito” a los datos que pueden interceptarse mientras se desplazan por la red.

La seguridad de la capa de transporte (TLS) cifra los objetos de Amazon MWAA en tránsito entre los componentes de Apache Airflow de su entorno y otros servicios de AWS que se integran con

Amazon MWAA, como Amazon S3. Para obtener más información sobre el cifrado Amazon S3, consulte cómo proteger los datos mediante cifrado.

Uso de claves maestras de cliente para el cifrado

Si lo desea, puede proporcionar una clave administrada por el cliente para el cifrado de datos de su entorno. Deberá crear la clave de KMS administrada por el cliente en la misma región que su instancia de entorno de Amazon MWAA y su bucket de Amazon S3 en el cual almacena los recursos para sus flujos de trabajo. Si la clave KMS administrada por el cliente que especifica está en una cuenta diferente de la que usa para configurar un entorno, debe especificarla mediante su ARN para el acceso entre cuentas. Para obtener más información sobre la creación de claves, consulte cómo crear claves en la guía para desarrolladores de AWS Key Management Service.

Elementos compatibles

AWS KMSCaracterística de	Compatible
Un ID o un ARN de la clave de AWS KMS.	Sí
Un alias de la clave de AWS KMS.	No
Una clave AWS KMS de varias regiones.	No

Uso de concesiones para el cifrado

En este tema, se describen las concesiones que Amazon MWAA asocia a una clave de KMS administrada por el cliente en su nombre para cifrar y descifrar sus datos.

Funcionamiento

AWS KMS admite dos mecanismos de control de acceso basados en recursos para la clave de KMS administrada por el cliente: una política de claves y una concesión.

La política de claves se utiliza cuando el permiso es principalmente estático y se utiliza en modo de servicio sincrónico, mientras que la concesión se utiliza cuando se requieren permisos más dinámicos y detallados, como cuando un servicio necesita definir diferentes permisos de acceso para sí mismo o para otras cuentas.

Amazon MWAA usa y asocia cuatro políticas de concesión a la clave de KMS administrada por el cliente. Esto se debe a que se necesitan permisos detallados para que un entorno cifre los datos en reposo procedentes de los registros de CloudWatch, la cola de Amazon SQS, la base de datos de Aurora PostgreSQL, la información secreta de Secrets Manager, el bucket de Amazon S3 y las tablas de DynamoDB.

Al crear un entorno de Amazon MWAA y especificar una clave de KMS administrada por el cliente, Amazon MWAA asocia las políticas de concesión a dicha clave. Estas políticas permiten que Amazon MWAA use su clave de KMS administrada por el cliente en `airflow.`*`us-east-1`*`.amazonaws.com` para cifrar en su nombre los recursos que son propiedad de Amazon MWAA.

Amazon MWAA crea y asocia concesiones adicionales a una clave de KMS específica en su nombre. Esto incluye políticas para retirar una concesión en caso de que elimine su entorno, para usar la clave de KMS administrada por el cliente para el cifrado del cliente (CSE) y para el rol de ejecución de AWS Fargate que necesita obtener acceso a los secretos protegidos por su clave administrada por el cliente en Secrets Manager.

Políticas de concesión

Amazon MWAA agrega las concesiones de la política basada en recursos que se indican a continuación en su nombre a una clave de KMS administrada por el cliente. Estas políticas permiten que el beneficiario y la entidad principal (Amazon MWAA) lleven a cabo las acciones definidas en la política.

Concesión 1: se utiliza para crear recursos del plano de datos

```
{
  "Name": "mwaa-grant-for-env-mgmt-role-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
```

```
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Concesión 2: se utiliza para el acceso del **`ControllerLambdaExecutionRole`**

```
{
  "Name": "mwaa-grant-for-lambda-exec-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Concesión 3: se utiliza para el acceso del **`CfnManagementLambdaExecutionRole`**

```
{
  "Name": " mwaa-grant-for-cfn-mgmt-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ]
}
```

Concesión 4: se utiliza para el rol de ejecución de Fargate a fin de acceder a la información secreta del backend

```
{
```

```
  "Name": "mwaa-fargate-access-for-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Asociación de políticas de claves a una clave administrada por el cliente

Si decide usar su propia clave de KMS administrada por el cliente con Amazon MWAA, debe asociar la siguiente política a la clave para que Amazon MWAA pueda usarla para cifrar sus datos.

Si la clave de KMS administrada por el cliente que use para su entorno de Amazon MWAA todavía no está configurada para que trabaje con CloudWatch, deberá actualizar la política de claves para permitir el cifrado de registros de CloudWatch. Para obtener más información, consulte el cifrado de datos de registro en CloudWatch mediante servicios de AWS Key Management Service.

En el siguiente ejemplo se representa una política de claves para los Registros de CloudWatch. Sustituya los valores de ejemplo que se han indicado para la región.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.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-east-1:*:*"
```

```
    }
  }
}
```

AWS Identity and Access Management

AWS Identity and Access Management (IAM) es un servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan a qué personas se puede autenticar (pueden iniciar sesión) y autorizar (tiene permisos) para utilizar recursos de Amazon Managed Workflows para Apache Airflow. IAM es un servicio de AWS que puede usar sin cargo adicional.

En este tema se ofrece información general básica acerca de cómo Amazon MWAA utiliza AWS Identity and Access Management (IAM). Para obtener más información sobre la administración del acceso a Amazon MWAA, consulte Administración del acceso a un entorno de Amazon MWAA.

Contenido

Público

La forma en que utiliza AWS Identity and Access Management (IAM) varía en función del rol:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de Amazon Managed Workflows para Apache Airflow relacionados con la identidad y el acceso)
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Amazon MWAA con IAM)
- Administrador de IAM: escriba políticas para administrar el acceso (consulte Ejemplos de políticas de Amazon MWAA basadas en identidades)

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado como Usuario raíz de la cuenta de AWS, un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de un origen de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales de Google/Facebook. Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Usuario raíz de la Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión llamada Cuenta de AWS del usuario raíz que tiene acceso completo a todos los Servicios de AWS y recursos. Recomendamos encarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales a largo plazo. Para obtener más información, consulte Solicitar que los usuarios humanos utilicen la federación con un proveedor de identidades para acceder a AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un usuario a un rol de IAM (consola) o llamando a una operación

de la API de la AWS CLI o AWS. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuarios federados, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política define permisos cuando se asocia con una identidad o recurso. AWS evalúa estas políticas cuando una entidad principal realiza una solicitud. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Mediante políticas, los administradores especifican quién tiene acceso a qué definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a los roles, que luego los usuarios pueden asumir. Las políticas de IAM definen permisos independientemente del método utilizado para realizar la operación.

Políticas basadas en identidad

Las políticas basadas en identidad son documentos JSON de políticas de permisos que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en la identidad pueden ser políticas integradas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las políticas de confianza de roles de IAM y las políticas de bucket de Amazon

S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política en función de recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas de AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulta Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer el máximo de permisos concedidos por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en la identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations.
- Políticas de control de recursos (RCP): establecen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si permitir o no una

solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo permitir a los usuarios que vean sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de 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": "*"
        }
    ]
}
```

Solución de problemas de Amazon Managed Workflows para Apache Airflow relacionados con la identidad y el acceso

Use la siguiente información para diagnosticar y solucionar los problemas comunes que pueden surgir cuando se trabaja con Amazon MWAA e IAM.

No tengo autorización para realizar una acción en Amazon MWAA

Si la Consola de administración de AWS le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

No tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, las políticas se deben actualizar para permitirle pasar un rol a Amazon MWAA.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado “`marymajor`” intenta utilizar la consola para llevar a cabo una acción en Amazon MWAA. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su gestionador de AWS. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de Amazon MWAA

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que

asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon MWAA admite estas características, consulte Cómo funciona Amazon MWAA con IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte Acceso para un usuario de IAM en otra Cuenta de AWS propia en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a tus recursos a Cuentas de AWS de terceros, consulta Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Cómo funciona Amazon MWAA con IAM

Amazon MWAA utiliza políticas de IAM basadas en la identidad para conceder permisos a las acciones y los recursos de Amazon MWAA. Para ver ejemplos de políticas de IAM personalizadas recomendadas que puede usar para controlar el acceso a los recursos de Amazon MWAA, consulte the section called “Acceso a un entorno de Amazon MWAA”.

Para obtener una perspectiva general sobre cómo funcionan Amazon MWAA y otros servicios de AWS con IAM, consulte los servicios de AWS que funcionan con IAM en la guía del usuario de IAM.

Políticas basadas en identidades de Amazon MWAA

Con las políticas basadas en identidad de IAM, puede especificar las acciones y recursos permitidos o denegados. Amazon MWAA admite acciones, claves de condiciones y recursos específicos.

En los siguientes pasos, se muestra cómo puede crear una nueva política JSON mediante la consola de IAM. Esta política administrada proporciona acceso de solo lectura a los recursos de Amazon MWAA.

Utilización del editor de política de JSON para la creación de una política

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Welcome to Managed Policies (Bienvenido a políticas administradas). Elija Comenzar.
3. En la parte superior de la página, seleccione Crear política.
4. En la sección Editor de políticas, seleccione la opción JSON.
5. Ingrese el siguiente documento de política JSON:

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

6. Elija Siguiente.

> **Note**
>
> Puede alternar entre las opciones Visual y JSON del editor en todo momento. No obstante, si realiza cambios o selecciona Siguiente en la opción Visual del editor, es posible que IAM reestructure la política, con el fin de optimizarla para el editor visual. Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.

7. En la página Revisar y crear, introduzca el Nombre de la política y la Descripción (opcional) para la política que está creando. Revise los Permisos definidos en esta política para ver los permisos que concede la política.
8. Elija Crear política para guardar la nueva política.

Para obtener información sobre todos los elementos que usa en una política JSON, consulte la referencia de los elementos de las políticas JSON de IAM en la guía del usuario de IAM.

Acciones

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las instrucciones de la política deben incluir un elemento `Action` o `NotAction`. El elemento `Action` enumera las acciones permitidas por la política, mientras que el elemento `NotAction` enumera las no permitidas.

Las acciones que se definan para Amazon MWAA reflejarán las tareas que puede realizar mediante Amazon MWAA. Las acciones de políticas en Detective tienen el siguiente prefijo: `airflow:`.

También puede utilizar comodines (*) para especificar varias acciones. En lugar de enumerar estas acciones por separado, puede otorgar acceso a todas las acciones que terminan con una palabra, por ejemplo, `environment`.

Para ver una lista de las acciones de Amazon MWAA, consulte las acciones definidas por Amazon Managed Workflows para Apache Airflow en la guía del usuario de IAM.

Ejemplos de políticas de Amazon MWAA basadas en identidades

Para acceder a las políticas de Amazon MWAA, consulte Administración del acceso a un entorno de Amazon MWAA.

De forma predeterminada, los usuarios y roles de IAM no tienen permiso para crear ni modificar recursos de Amazon MWAA. Tampoco pueden realizar tareas mediante la Consola de administración de AWS, la AWS CLI, o la API de AWS.

Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. En ese caso, el administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

> **Important**
>
> Recomendamos utilizar roles de IAM y credenciales temporales para proporcionar acceso a los recursos de Amazon MWAA. Trate de no asociar directamente políticas de permisos a sus usuarios de IAM.

Para obtener información sobre cómo crear una política basada en identidad de IAM con el ejemplo con documentos de políticas JSON, consulte cómo crear políticas en la pestaña JSON en la guía del usuario de IAM.

Temas

- Prácticas recomendadas sobre las políticas
- Uso de la consola de Amazon MWAA
- Cómo permitir a los usuarios que vean sus propios permisos

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon MWAA de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comienza con las políticas administradas por AWSy continúa con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de tarea, utiliza las políticas administradas por AWS, que conceden permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Se recomienda definir políticas gestionadas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se

pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.

- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puedes escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado como, por ejemplo, CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesite usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para obtener una mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Amazon MWAA

Para utilizar la consola de Amazon MWAA, el usuario o el rol deben poder acceder a las acciones pertinentes, que deben coincidir con las acciones correspondientes de la API.

Para acceder a las políticas de Amazon MWAA, consulte Administración del acceso a un entorno de Amazon MWAA.

Cómo permitir a los usuarios que vean sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política

incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de 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": "*"
        }
    ]
}
```

Validación de conformidad de Amazon Managed Workflows para Apache Airflow

Para saber si un Servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte Servicios de AWS incluidos en el ámbito del programa de conformidad y elija el

programa de conformidad que le interese. Para obtener información general, consulte Programas de conformidad de AWS.

Puedes descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la confidencialidad de los datos, los objetivos de conformidad de su empresa, así como de la legislación y los reglamentos aplicables. Para obtener más información sobre la responsabilidad de conformidad al usar Servicios de AWS, consulte la Documentación de seguridad de AWS.

Resiliencia en Amazon Managed Workflows para Apache Airflow

La infraestructura global de AWS se construye en torno a las Regiones de AWS y a las zonas de disponibilidad. Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja demora. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre las zonas de disponibilidad y Regiones de AWS, consulte la infraestructura global de AWS.

Seguridad de la infraestructura en Amazon MWAA

Como se trata de un servicio administrado, Amazon Managed Workflows para Apache Airflow se encuentra protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y sobre cómo AWS protege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de AWS siguiendo las prácticas recomendadas de seguridad de infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas en AWS para acceder a Amazon MWAA a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.

- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Configuración y análisis de vulnerabilidades en Amazon MWAA

La configuración y los controles de TI son una responsabilidad compartida entre AWS y usted, nuestro cliente.

Amazon Managed Workflows para Apache Airflow revisa y actualiza Apache Airflow en sus entornos de forma periódica. Procure que se usen las políticas de acceso adecuadas para sus VPC.

Para obtener más información, consulte los siguientes recursos:

- Validación de conformidad de Amazon Managed Workflows para Apache Airflow
- Modelo de responsabilidad compartida
- Amazon Web Services: Overview of Security Processes
- Seguridad de la infraestructura en Amazon MWAA
- Prácticas recomendadas de seguridad para Amazon MWAA

Prácticas recomendadas de seguridad para Amazon MWAA

Amazon MWAA proporciona una serie de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

- Utilice las políticas de permisos que sean menos permisivas. Otorgue permisos únicamente a los recursos o a las acciones que los usuarios necesiten para llevar a cabo las tareas.
- Utilice AWS CloudTrail para monitorizar la actividad de los usuarios en su cuenta.
- Asegúrese de que la política de buckets de Amazon S3 y las listas de control de acceso (ACL) a objetos concedan permisos a los usuarios del entorno de Amazon MWAA asociado para que puedan colocar objetos en el bucket. Esto garantizará que los usuarios con permisos para añadir flujos de trabajo al bucket también dispongan de permisos para ejecutar los flujos de trabajo en Airflow.

- Utilice los buckets de Amazon S3 asociados a los entornos de Amazon MWAA. Su bucket de Amazon S3 puede tener cualquier nombre. No almacene otros objetos en el bucket ni lo use con otro servicio.

Prácticas recomendadas de seguridad en Apache Airflow

Apache Airflow no es multitenencia. Si bien Amazon MWAA implementa algunas medidas de control de acceso para limitar algunas características a usuarios específicos, los creadores de DAG tienen la capacidad de escribir DAG que pueden cambiar los privilegios de los usuarios de Apache Airflow e interactuar con la base de metadatos subyacente.

Recomendamos seguir los siguientes consejos al trabajar con Apache Airflow en Amazon MWAA a fin de garantizar la seguridad de la base de metadatos y los DAG de su entorno.

- Utilice entornos separados para equipos distintos que tengan acceso a la escritura de los DAG, o bien tengan la capacidad de añadir archivos a su carpeta `/dags` de Amazon S3, suponiendo que los usuarios que puedan escribir en el entorno también podrían acceder a cualquier a la que se pueda acceder mediante el rol de ejecución de Amazon MWAA o las conexiones de Apache Airflow.
- No proporcione acceso directo a las carpetas de los DAG de Amazon S3. En su lugar, utilice las herramientas de integración continua (CI) y entrega continua (CD) para escribir los DAG en Amazon S3, añadiendo un paso de validación que garantice que el código de los DAG cumple con las directrices de seguridad de su equipo.
- Evite que los usuarios puedan acceder al bucket de Amazon S3. En su lugar, utilice un generador de DAG que genere DAG basados en un archivo YAML, JSON u otro archivo de definición que esté almacenado en una ubicación diferente a la de su bucket de Amazon S3 de Amazon MWAA en el que almacena los DAG.
- Almacene la información secreta en Secrets Manager. Si bien esto no impedirá que los usuarios que pueden escribir DAG lean esa información, sí evitará que puedan modificar la información secreta que utiliza su entorno.

Detección de cambios en los privilegios de los usuarios de Apache Airflow

Puede utilizar la información de los registros de CloudWatch para detectar si algún DAG cambia los privilegios de los usuarios de Apache Airflow. Para ello, puede utilizar una regla programada de EventBridge, una función de Lambda y la información de registros de CloudWatch para que se envíe

una notificación a las métricas de CloudWatch cada vez que uno de sus DAG cambie los privilegios de los usuarios de Apache Airflow.

Requisitos previos

Para completar los pasos que se detallan a continuación, necesita lo siguiente:

- Un entorno de Amazon MWAA que tenga habilitados todos los tipos de registros de Apache Airflow en el nivel de registro `INFO`. Para obtener más información, consulta the section called "Visualización de registros de Airflow".

Configuración de las notificaciones acerca de cualquier cambio que se produzca en los privilegios de los usuarios de Apache Airflow

1. Cree una función de Lambda que ejecute la siguiente cadena de consulta de la información de los registros de CloudWatch en los cinco grupos de registros del entorno Amazon MWAA (`DAGProcessing`, `Scheduler`, `Task`, `WebServer` y `Worker`).

   ```
   fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
    by @log
   ```

2. Cree una regla de EventBridge que se ejecute según una programación, con la función de Lambda que creó en el paso anterior como objetivo de la regla. Configure su programación mediante una expresión de frecuencia o una expresión cron para que se ejecute a intervalos regulares.

Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow

En esta página, se describen las versiones de Apache Airflow compatibles con Amazon Managed Workflows para Apache Airflow y prácticas recomendadas para actualizar a la versión más reciente.

Temas

Acerca de las versiones de Amazon MWAA

Amazon MWAA crea imágenes de contenedores que agrupan las versiones de Apache Airflow con otros binarios y bibliotecas de Python comunes. La imagen usa la instalación básica de Apache Airflow para la versión que usted especifique. Cuando cree un entorno, tiene que especificar la versión de imagen que se va a utilizar. Una vez creado el entorno, se sigue utilizando la versión de imagen especificada hasta que se actualice a una versión posterior.

Última versión

Amazon MWAA admite varias versiones de Apache Airflow. Si no especifica una versión de imagen al crear un entorno, Amazon MWAA crea el entorno con la última versión compatible de Apache Airflow.

Versiones de Apache Airflow

Las siguientes versiones de Apache Airflow son compatibles con Amazon Managed Workflows para Apache Airflow.

Note

- A partir del 30 de diciembre de 2025, Amazon MWAA dejará de ser compatible con las versiones v2.4.3, v2.5.1 y v2.6.3 de Apache Airflow. Para obtener más información, consulta Preguntas frecuentes y compatibilidad de Apache Airflow.
- A partir de Apache Airflow v2.2.2, Amazon MWAA admite la instalación de requisitos de Python, paquetes de proveedores y complementos personalizados directamente en el servidor web Apache Airflow.
- A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que se está usando.

  Para obtener más información sobre la configuración de restricciones en su archivo de requisitos, consulte cómo instalar dependencias de Python.

Versión de Apache Airflow	Fecha de la versión de Apache Airflow	Fecha de disponibilidad en Amazon MWAA	Restricciones de Apache Airflow	Versión de Python
v3.0.6	de agosto de 29, 2025	1 de octubre de 2025	archivo de restricciones v3.0.6	Python 3.12
v2.10.3	de noviembre de 4, 2024	18 de diciembre de 2024	archivo de restricciones v2.10.3	Python 3.11
v2.10.1	de septiembre de 5, 2024	26 de septiembre de 2024	Archivo de restricciones v2.10.1	Python 3.11
v2.9.2	de junio de 10, 2024	9 de julio de 2024	Archivo de restricciones v2.9.2	Python 3.11

Versión de Apache Airflow	Fecha de la versión de Apache Airflow	Fecha de disponibilidad en Amazon MWAA	Restricciones de Apache Airflow	Versión de Python
v2.8.1	de enero de 19, 2024	23 de febrero de 2024	Archivo de restricciones v2.8.1	Python 3.11
v2.7.2	de octubre de 12, 2023	6 de noviembre de 2023	Archivo de restricciones v2.7.2	Python 3.11

Para obtener más información sobre la migración de sus implementaciones autoadministradas de Apache Airflow o la migración de un entorno Amazon MWAA existente, incluidas las instrucciones para realizar copias de seguridad de su base de datos de metadatos, consulte la guía de migración a Amazon MWAA.

Componentes de Apache Airflow

En esta sección se describe el número de programadores y procesos de trabajo de Apache Airflow disponibles para cada versión de Apache Airflow en Amazon MWAA y se muestra una lista con las principales características de Apache Airflow, en la que se indica la versión compatible con cada característica.

Programadores

Programadores para Apache Airflow v2 y versiones posteriores:

Programador (predeter minado)	Programador (mín.)	Programador (máx.)
2	2	5

Procesos de trabajo

Procesos de trabajo de Apache Airflow v2 y versiones posteriores:

Procesos de trabajo (predeter minado)	Procesos de trabajo (mín.)	Procesos de trabajo (máx.)
10	1	25

Actualización de la versión de Apache Airflow

Amazon MWAA admite la actualización a versiones menores. Esto significa que puede actualizar su entorno de la versión x**.1.**z a otra x**.2.**z, pero no a una nueva versión principal, por ejemplo, de **1.**y**.**z a otra **2.**y**.**z.

Para obtener más información e instrucciones detalladas sobre cómo actualizar los recursos de flujo de trabajo y actualizar el entorno a una versión nueva, consulte the section called “Cambio de versión”.

Actualización de la versión de Apache Airflow

Amazon MWAA se puede degradar a una versión anterior que sea compatible en el momento de la degradación. Esto significa que puede degradar su entorno de la versión x**.2.**z a otra x**.1.**z, pero no a una versión principal anterior, por ejemplo, de **2.**y**.**z a otra **1.**y**.**z.

Para obtener más información e instrucciones detalladas sobre cómo actualizar los recursos de flujo de trabajo y actualizar el entorno a una versión nueva, consulte the section called “Cambio de versión”.

Versiones obsoletas de Apache Airflow

En la siguiente tabla, se enumeran las versiones obsoletas de Apache Airflow en Amazon MWAA, junto con las fechas de lanzamiento y finalización de la compatibilidad de cada versión. Para obtener más información sobre la migración a una versión más reciente, consulte la guía de migración de Amazon MWAA.

Versión de Apache Airflow	Fecha de la versión de Apache Airflow	Fecha de disponibilidad en Amazon MWAA	Fecha de finalización de la compatibilidad en Amazon MWAA
v1.10.12	25 de agosto de 2020	24 de noviembre de 2020	21 de febrero de 2024
v2.0.2	19 de abril de 2021	25 de mayo de 2021	29 de abril de 2024
v2.2.2	15 de noviembre de 2021	27 de enero de 2022	27 de junio de 2024
v2.4.3	14 de noviembre de 2022	05 de enero de 2023	30 de diciembre de 2025
v2.5.1	20 de enero de 2023	11 de abril de 2023	30 de diciembre de 2025
v2.6.3	10 de julio de 2023	09 de agosto de 2023	30 de diciembre de 2025

Preguntas frecuentes y compatibilidad de Apache Airflow

De conformidad con el proceso de lanzamiento y la política de versiones de la comunidad de Apache Airflow, Amazon MWAA se compromete a admitir al menos tres versiones secundarias de Apache Airflow en un momento dado. Anunciaremos la fecha de finalización del soporte de una versión secundaria de determinada al menos 180 días antes de la fecha de finalización del soporte.

Preguntas frecuentes

P: ¿Durante cuánto tiempo será compatible Amazon MWAA con una versión de Apache Airflow?

R: Amazon MWAA será compatible con una versión revisada de Apache Airflow durante un mínimo de 12 meses desde que estuviese disponible por primera vez.

P: ¿Se me notifica cuándo finaliza el soporte para una versión de Apache Airflow en Amazon MWAA?

R: Sí. Si alguno de los entornos de Amazon MWAA de su cuenta ejecuta una versión cuyo soporte está a punto de finalizar, Amazon MWAA envía un aviso a través de AWS Health Dashboard para indicar la fecha de finalización del soporte.

P: ¿Qué sucede cuando llega la fecha de finalización del soporte?

R: Cuando llegue la fecha de finalización de la compatibilidad, ya no podrá usar versiones obsoletas para crear nuevos entornos de Amazon MWAA. Podrá seguir accediendo a sus entornos existentes de Amazon MWAA que ejecutan la versión obsoleta de Apache Airflow asociada bajo su propia responsabilidad. Para obtener instrucciones sobre cómo actualizar a una versión más reciente de Apache Airflow en Amazon MWAA, consulte la guía de migración a Amazon MWAA.

Important

Usted es responsable de mantener actualizadas sus versiones de Amazon MWAA. AWS insta a todos los clientes a actualizar sus entornos de Amazon MWAA a la última versión para beneficiarse de las medidas de seguridad, privacidad y disponibilidad más recientes. Si utiliza su entorno en una versión o software no compatibles después de la fecha de caducidad, conocida como la versión heredada, enfrenta una mayor probabilidad de riesgos operativos, de seguridad y de privacidad, incluidos eventos de tiempo de inactividad. El uso de su entorno de Amazon MWAA en una versión heredada significa que confirma que entiende y asume conscientemente estos riesgos y acepta finalizar la actualización a la última versión lo antes posible. La utilización continua de su entorno en una versión heredada está sujeta al acuerdo que rige el uso de los servicios de AWS.

Las versiones heredadas no se consideran generalmente disponibles, y AWS ya no proporciona compatibilidad con la versión heredada. Como resultado, es posible que AWS imponga límites al acceso o uso de cualquier versión heredada en cualquier momento si AWS determina que esta representa un riesgo de seguridad, responsabilidad o daño para los servicios, AWS, sus afiliados o cualquier otro tercero. Su decisión de continuar ejecutando las cargas de trabajo en una versión heredada podría dar como resultado que su contenido deje de estar disponible, se corrompa o no se pueda recuperar. Los entornos que se ejecutan en una versión heredada están sujetos a excepciones del Acuerdo de Nivel de Servicio (SLA).

Los entornos y el software relacionado que se ejecutan en una versión heredada pueden contener errores, defectos y componentes dañinos. En consecuencia, y sin perjuicio de cualquier información en contrario en el acuerdo o en los términos de servicio, AWS proporciona la versión heredada tal cual está.

Para obtener más información sobre el modelo de responsabilidad compartida de AWS, consulte responsabilidad compartida en Well-Architected Framework de AWS.

Puntos de conexión y cuotas de servicio de Amazon Managed Workflows para Apache Airflow

Amazon Managed Workflows para Apache Airflow cuenta con las siguientes cuotas y puntos de conexión de servicio. Service Quotas, también denominadas límites, establecen el número máximo de recursos u operaciones de servicio para su cuenta de Cuenta de AWS.

Contenido

Puntos de conexión de servicio

Para ver una lista con los puntos de conexión de Amazon MWAA, consulte Amazon Managed Workflows para cuotas y puntos de conexión de Apache Airflow.

Service Quotas

Nombre de la cuota	Descripción	Cuota predeterminada	Ajustable
Entornos	El número máximo de entornos de Amazon MWAA por cuenta y región.	10	Sí
Procesos de trabajo por entorno	El número máximo de procesos de trabajo por entorno de Amazon MWAA.	25	Sí
Servidores web por entorno	El número máximo de servidores web por entorno de Amazon MWAA.	5	Sí

Aumento de cuotas

Puede solicitar un aumento de una cuota ajustable enviando una solicitud de aumento de la cuota.

Preguntas frecuentes sobre Amazon MWAA

En esta página, se describen las preguntas más frecuentes que se pueden plantear al usar Amazon Managed Workflows para Apache Airflow.

Contenido

Versiones compatibles

¿Qué es lo que Amazon MWAA admite para Apache Airflow v2?

Para obtener información sobre la compatibilidad de Amazon MWAA, consulte Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow.

¿Qué versión de Python puedo usar?

Las siguientes versiones de Apache Airflow son compatibles con Amazon Managed Workflows para Apache Airflow.

> **Note**
>
> - A partir del 30 de diciembre de 2025, Amazon MWAA dejará de ser compatible con las versiones v2.4.3, v2.5.1 y v2.6.3 de Apache Airflow. Para obtener más información, consulta Preguntas frecuentes y compatibilidad de Apache Airflow.
> - A partir de Apache Airflow v2.2.2, Amazon MWAA admite la instalación de requisitos de Python, paquetes de proveedores y complementos personalizados directamente en el servidor web Apache Airflow.
> - A partir de la versión 2.7.2 de Apache Airflow, su archivo de requisitos debe incluir una instrucción `--constraint`. Si no proporciona ninguna restricción, Amazon MWAA

especificará una para garantizar que los paquetes que figuran en sus requisitos sean compatibles con la versión de Apache Airflow que se está usando.

Para obtener más información sobre la configuración de restricciones en su archivo de requisitos, consulte cómo instalar dependencias de Python.

Versión de Apache Airflow	Fecha de la versión de Apache Airflow	Fecha de disponibilidad en Amazon MWAA	Restricciones de Apache Airflow	Versión de Python
v3.0.6	de agosto de 29, 2025	1 de octubre de 2025	archivo de restricciones v3.0.6	Python 3.12
v2.10.3	de noviembre de 4, 2024	18 de diciembre de 2024	archivo de restricciones v2.10.3	Python 3.11
v2.10.1	de septiembre de 5, 2024	26 de septiembre de 2024	Archivo de restricciones v2.10.1	Python 3.11
v2.9.2	de junio de 10, 2024	9 de julio de 2024	Archivo de restricciones v2.9.2	Python 3.11
v2.8.1	de enero de 19, 2024	23 de febrero de 2024	Archivo de restricciones v2.8.1	Python 3.11
v2.7.2	de octubre de 12, 2023	6 de noviembre de 2023	Archivo de restricciones v2.7.2	Python 3.11

Para obtener más información sobre la migración de sus implementaciones autoadministradas de Apache Airflow o la migración de un entorno Amazon MWAA existente, incluidas las instrucciones

para realizar copias de seguridad de su base de datos de metadatos, consulte la guía de migración a Amazon MWAA.

Casos de uso

¿Puedo usar Amazon MWAA con Amazon SageMaker Unified Studio?

Sí. Con un flujo de trabajo de Amazon SageMaker Unified Studio, puede configurar y ejecutar una serie de tareas en ese entorno. Los flujos de trabajo de Amazon SageMaker Unified Studio usan Apache Airflow para modelar los procedimientos de procesamiento de datos y orquestar los artefactos de código de Amazon SageMaker Unified Studio. Para obtener más información, consulte la sección Flujos de trabajo. Para obtener más información sobre Amazon SageMaker, consulte ¿Qué es Amazon SageMaker?

¿Cuándo debo usar AWS Step Functions en vez de Amazon MWAA?

1. Puede utilizar Step Functions para procesar pedidos de clientes individuales, ya que Step Functions puede ampliarse para satisfacer la demanda de un pedido o un millón de pedidos.
2. Si tiene un flujo de trabajo nocturno que procesa los pedidos del día anterior, puede usar Step Functions o Amazon MWAA. Amazon MWAA le ofrece una opción de código abierto para abstraer el flujo de trabajo de los recursos de AWS que se están usando.

Notificaciones del entorno

¿Cuánto espacio de almacenamiento de tareas está disponible en cada entorno?

El almacenamiento de tareas está limitado a 20 GB y especificado por Amazon ECS Fargate 1.4. La cantidad de RAM viene determinada por la clase de entorno que especifique. Para obtener más información sobre las clases de entorno, consulte Configuración de la clase de entorno Amazon MWAA.

¿Cuál es el sistema operativo predeterminado que se utiliza para los entornos de Amazon MWAA?

Los entornos de Amazon MWAA se crean en instancias que ejecutan Amazon Linux 2 para las versiones 2.6 y anteriores y en instancias que ejecutan Amazon Linux 2023 para las versiones 2.7 y posteriores.

¿Puedo usar una imagen personalizada para mi entorno Amazon MWAA?

Las imágenes personalizadas no son compatibles. Amazon MWAA utiliza imágenes creadas en la AMI de Amazon Linux. Amazon MWAA instala los requisitos adicionales ejecutando `pip3 -r install` para los requisitos especificados en el archivo requirements.txt que añada al bucket de Amazon S3 para el entorno.

¿Cumple Amazon MWAA con la HIPAA?

Amazon MWAA es elegible para Ley de Portabilidad y Responsabilidad de Seguros Médicos de EE. UU (HIPAA). Si cuenta con un apéndice para socios comerciales (BAA) de la HIPAA con AWS, puede utilizar Amazon MWAA para los flujos de trabajo que gestionan la información de salud protegida (PHI) en entornos creados a partir del 14 de noviembre de 2022.

¿Amazon MWAA es compatible con instancias de spot?

Actualmente, Amazon MWAA no es compatible con los tipos de instancias de spot Amazon EC2 bajo demanda para Apache Airflow. Sin embargo, un entorno de Amazon MWAA puede activar instancias de spot en, por ejemplo, Amazon EMR y Amazon EC2.

¿Amazon MWAA es compatible con un dominio personalizado?

Para poder utilizar un dominio personalizado para el nombre de host de Amazon MWAA, realice una de las siguientes acciones:

- Para las implementaciones de Amazon MWAA con acceso a servidores web públicos, puede utilizar Amazon CloudFront con Lambda@Edge para dirigir el tráfico a su entorno y asignar un nombre de dominio personalizado a CloudFront. Para obtener más información y un ejemplo de cómo configurar un dominio personalizado para un entorno público, consulte el ejemplo de dominio personalizado de Amazon MWAA para servidor web público en el repositorio GitHub de ejemplos de Amazon MWAA.

- Para ver las implementaciones de Amazon MWAA con acceso a un servidor web privado, consulte the section called “Configuración de un dominio personalizado”.

¿Puedo usar SSH en mi entorno?

Si bien SSH no es compatible con un entorno Amazon MWAA, es posible utilizar un DAG para ejecutar comandos bash mediante el `BashOperator`. Por ejemplo:

```
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 activar el DAG en la interfaz de usuario de Apache Airflow, utilice:

```
{ "command" : "your bash command"}
```

¿Por qué se requiere una regla de autorreferencia en el grupo de seguridad de VPC?

Si crea una regla de entrada con autorreferencia, puede restringir el origen al mismo grupo de seguridad de la VPC y no está abierto a todas las redes. Consulte the section called “Seguridad en la VPC” para obtener más información.

¿Puedo ocultar entornos de diferentes grupos en IAM?

Puede limitar el acceso especificando un nombre de entorno en AWS Identity and Access Management; sin embargo, el filtrado de visibilidad no está disponible en la consola de AWS. Si un usuario puede ver un entorno, podrá ver todos los entornos.

¿Puedo almacenar datos temporales en el proceso de trabajo de Apache Airflow?

Sus operadores de Apache Airflow pueden almacenar datos temporales sobre los procesos de trabajo. Los procesos de trabajo de Apache Airflow pueden acceder a los archivos temporales de `/tmp` en los contenedores de Fargate de su entorno.

> **Note**
>
> El almacenamiento total de tareas está limitado a 20 GB, según Amazon ECS Fargate 1.4. No hay garantía de que las tareas subsiguientes se ejecuten en la misma instancia de contenedor de Fargate, que podría usar una carpeta de `/tmp` diferente.

¿Puedo especificar más de 25 procesos de trabajo de Apache Airflow?

Sí. Si bien puede especificar hasta 25 procesos de trabajo de Apache Airflow en la consola de Amazon MWAA, puede configurar hasta 50 en un entorno solicitando un aumento de la cuota. Para obtener más información, consulte cómo solicitar un aumento de cuota.

¿Admite Amazon MWAA subredes compartidas o VPC de Amazon compartidas?

Amazon MWAA no admite subredes compartidas o VPC de Amazon compartidas. La VPC de Amazon que seleccione al crear un entorno debe ser propiedad de la cuenta que intenta crear el entorno. Sin embargo, puede enrutar el tráfico desde una VPC de Amazon de la cuenta de Amazon MWAA a una VPC compartida. Para obtener más información y ver un ejemplo de enrutamiento del tráfico a una VPC de Amazon compartida, consulte Enrutamiento saliente centralizado a Internet en la guía de puertas de enlance en tránsito de Amazon VPC.

¿Puedo crear o integrar colas de Amazon SQS personalizadas para administrar la ejecución de tareas y la organización del flujo de trabajo en Apache Airflow?

No, no puede crear, modificar ni usar colas de Amazon SQS personalizadas en Amazon MWAA. Esto se debe a que Amazon MWAA aprovisiona y administra automáticamente su propia cola de Amazon SQS para cada entorno de Amazon MWAA.

Métricas

¿Qué métricas se usan para determinar si se deben escalar procesos de trabajo?

Amazon MWAA supervisa las tareas en cola y las tareas en ejecución en CloudWatch para determinar si se deben escalar los procesos de trabajo de Apache Airflow en su entorno. Consulte Monitorización y métricas para obtener más información.

¿Puedo crear una métrica personalizada en CloudWatch?

No en la consola de CloudWatch. Sin embargo, puede crear un DAG que escriba métricas personalizadas en CloudWatch. Para obtener más información, consulta the section called “Uso de un DAG para escribir métricas personalizadas”.

DAG, operadores, conexiones y otras preguntas

¿Puedo usar el **`PythonVirtualenvOperator`**?

Amazon MWAA no admite `PythonVirtualenvOperator` de forma explícita, pero puede crear un complemento personalizado que utilice `PythonVirtualenvOperator`. Para ver un ejemplo de código, consulte the section called “Complemento personalizado para parchear PythonVirtualEnvOperator ”.

¿Cuánto tarda Amazon MWAA en reconocer un nuevo archivo DAG?

Los DAG se sincronizan periódicamente desde el bucket de Amazon S3 hasta el entorno. Si añade un nuevo archivo DAG, Amazon MWAA tardará unos 300 segundos en empezar a utilizar el nuevo archivo. Si actualiza un DAG existente, Amazon MWAA tarda unos 30 segundos en reconocer las actualizaciones.

Estos valores, 300 segundos para los DAG nuevos y 30 segundos para las actualizaciones de los DAG existentes, corresponden a las opciones de configuración de Apache Airflow `dag_dir_list_interval` y `min_file_process_interval` respectivamente.

¿Por qué Apache Airflow no recoge mi archivo DAG?

Las siguientes son posibles soluciones para este problema:

1. Compruebe que el rol de ejecución dispone de permisos suficientes para el bucket de Amazon S3. Consulte Rol de ejecución de Amazon MWAA para obtener más información.
2. Compruebe que el bucket de Amazon S3 tenga configurado el bloqueo de acceso público y que el control de versiones esté activado. Consulte Creación de un bucket de Amazon S3 para Amazon MWAA para obtener más información.
3. Compruebe el propio archivo DAG. Por ejemplo, asegúrese de que cada DAG tenga un ID de DAG único.

¿Puedo eliminar un **`plugins.zip`** o **`requirements.txt`** de un entorno?

Actualmente, no hay forma de eliminar un archivo plugins.zip o requirements.txt de un entorno después de haberlo añadido, pero estamos trabajando para solucionar el problema. Mientras tanto, una solución alternativa es apuntar a un archivo de texto o zip vacío, respectivamente. Consulte Eliminación de archivos en Amazon S3 para obtener más información.

¿Por qué no veo mis complementos en el menú de complementos de administrador de Apache Airflow v2.0.2?

Por motivos de seguridad, el servidor web de Apache Airflow de Amazon MWAA tiene una salida de red limitada y no instala complementos ni dependencias de Python directamente en el servidor web de Apache Airflow para los entornos de la versión 2.0.2. El complemento que se muestra permite a Amazon MWAA autenticar a los usuarios de Apache Airflow en AWS Identity and Access Management (IAM).

Para poder instalar complementos y dependencias de Python directamente en el servidor web, recomendamos crear un nuevo entorno con Apache Airflow v2.2 y versiones posteriores. Amazon MWAA instala las dependencias de Python y los complementos personalizados directamente en el servidor web para Apache Airflow v2.2 y versiones posteriores.

¿Puedo utilizar operadores Database Migration Service (DMS) de AWS?

Amazon MWAA es compatible con los operadores de DMS. Sin embargo, este operador no se puede utilizar para realizar acciones en la base de datos de metadatos de Amazon Aurora PostgreSQL asociada a un entorno Amazon MWAA.

Si accedo a la API de REST de Airflow con las credenciales de AWS, ¿puedo aumentar la limitación a más de 10 transacciones por segundo (TPS)?

Sí, puede. Para aumentar la limitación, póngase en contacto con el servicio de atención al cliente de AWS.

¿Dónde se ejecuta el servidor de la API de ejecución de tareas de Airflow en Amazon MWAA?

Amazon MWAA ejecuta el servidor API de ejecución de tareas de Airflow en el componente de servidor web. Las API de ejecución de tareas solo están disponibles en Apache Airflow v3 y versiones posteriores. Para obtener más información sobre la arquitectura de Amazon MWAA, consulte Arquitectura.

Solución de problemas de Amazon Managed Workflows para Apache Airflow

En este capítulo, se describen los problemas y errores más comunes que se puede encontrar al usar Apache Airflow en Amazon Managed Workflows para Apache Airflow y se recomiendan los pasos para resolverlos.

Contenido

- Me sale Executor reports task instance %s finished (%s) although the task says its %s en los registros de procesamiento de DAG
- Me sale Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log. en los registros de tareas

Solución de problemas: DAG, operadores, conexiones y otros problemas

En los temas de esta página, se describen las soluciones a las dependencias de Python de Apache Airflow v2 y v3, los complementos personalizados, los DAG, los operadores, las conexiones, las tareas y los problemas del servidor web que pueden surgir en un entorno de Amazon Managed Workflows para Apache Airflow.

Contenido

- Se produjo un error PermissionError: [Errno 13] Permission denied al utilizar el operador S3Transform

Connections

En el siguiente tema, se describen los errores que se pueden producir al usar una conexión de Apache Airflow o al usar otra base de datos de AWS.

No puedo conectarme a Secrets Manager

Recomendamos los siguientes pasos:

1. Aprenda a crear claves secretas para su conexión y variables de Apache Airflow en the section called “Configuración de Secrets Manager”.
2. Aprenda a usar la clave secreta para una variable de Apache Airflow (`test-variable`) en Uso de una clave secreta en AWS Secrets Manager para una variable de Apache Airflow.
3. Aprenda a usar la clave secreta para una conexión de Apache Airflow (`myconn`) en Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow.

¿Cómo configuro las condiciones del administrador de información secreta de **`secretsmanager:ResourceTag/<tag-key>`** o una restricción de recursos en mi política de funciones de ejecución?

> **Note**
>
> Se aplica a la versión 2.0 y anteriores de Apache Airflow.

Actualmente no puede limitar el acceso a la información secreta de Secrets Manager mediante claves de condición u otras restricciones de recursos en el rol de ejecución de su entorno, debido a un problema conocido en Apache Airflow.

No puedo conectarme a Snowflake

Recomendamos los siguientes pasos:

1. Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

2. Agregue las siguientes entradas al archivo requirements.txt de su entorno.

```
apache-airflow-providers-snowflake==1.3.0
```

3. Añada las siguientes importaciones al DAG:

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

Asegúrese de que el objeto de conexión de Apache Airflow incluya los siguientes pares clave-valor:

1. ID de conexión: snowflake_conn
2. Tipo de conexión: Snowflake
3. Host: <my account> <my region if not us-west-2>.snowflakecomputing.com
4. Esquema: <my schema>
5. Inicio de sesión: <my user name>
6. Contraseña: ********
7. Puerto: <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 ejemplo:

```
>>> import json
    >>> from airflow.models.connection import Connection
    >>> myconn = Connection(
    ...    conn_id='snowflake_conn',
    ...    conn_type='Snowflake',
    ...    host='123456789012.us-east-1.snowflakecomputing.com',
    ...    schema='YOUR_SCHEMA'
    ...    login='YOUR_USERNAME',
    ...    password='YOUR_PASSWORD',
```

```
    ...     port='YOUR_PORT'
    ...     extra=json.dumps(dict(account='123456789012', warehouse='YOUR_WAREHOUSE',
database='YOUR_DB_OPTION', region='us-east-1')),
    ... )
```

No veo mi conexión en la UI de Airflow

Apache Airflow proporciona plantillas de conexión en la interfaz de usuario de Apache Airflow. Lo usa para generar la cadena URI de conexión, independientemente del tipo de conexión. Si no hay ninguna plantilla de conexión disponible en la interfaz de usuario de Apache Airflow, se puede utilizar una plantilla de conexión alternativa para generar una cadena URI de conexión, por ejemplo, mediante la plantilla de conexión HTTP.

Recomendamos los siguientes pasos:

1. Consulte los tipos de conexión que proporciona Amazon MWAA en la UI de Apache Airflow en Paquetes de proveedores de Apache Airflow instalados en entornos Amazon MWAA.
2. Consulte los comandos para crear una conexión de Apache Airflow en la CLI de Referencia de los comandos de la CLI de Apache Airflow.
3. Aprenda a utilizar las plantillas de conexión de la interfaz de usuario de Apache Airflow indistintamente para los tipos de conexión que no están disponibles en la interfaz de usuario de Apache Airflow en Amazon MWAA de Información general sobre los tipos de conexión.

Servidor web

En el siguiente tema, se describen los errores que puede encontrar en su servidor web de Apache Airflow en Amazon MWAA.

Aparece el error 5xx al acceder al servidor web

Recomendamos los siguientes pasos:

1. Compruebe las opciones de configuración de Apache Airflow. Compruebe que los pares clave-valor que especificó como opción de configuración de Apache Airflow, por ejemplo AWS Secrets Manager, se hayan configurado correctamente. Consulte the section called “No puedo conectarme a Secrets Manager” para obtener más información.
2. Compruebe los `requirements.txt`. Compruebe que el paquete “extras” de Airflow y las demás bibliotecas que figuran en su `requirements.txt` sean compatibles con su versión de Apache Airflow.

3. Considere otras formas de especificar las dependencias de Python en un archivo `requirements.txt`, que se detallan en Administración de las dependencias de Python en requirements.txt.

Me da el error **`The scheduler does not seem to be running`**

Si parece que el programador no está funcionando o si la última señal se recibió hace varias horas, es posible que sus DAG no aparezcan en Apache Airflow y no se programen nuevas tareas.

Recomendamos los siguientes pasos:

1. Confirme que su grupo de seguridad de VPC permita el acceso entrante al puerto `5432`. Este puerto es necesario para conectarse a la base de datos de metadatos de PostgreSQL de Amazon Aurora de su entorno. Tras añadir esta regla, dé unos minutos a Amazon MWAA y el error desaparecerá. Consulte the section called “Seguridad en la VPC” para obtener más información.

   > **Note**
   >
   > - La base de metadatos de Aurora PostgreSQL forma parte de la arquitectura de servicios de Amazon MWAA y no es visible en su Cuenta de AWS.
   > - Los errores relacionados con la base de datos suelen ser síntoma de un fallo del programador, no su causa principal.

2. Si el programador no se está ejecutando, podría deberse a varios factores, como errores en la instalación de las dependencias o a una sobrecarga del programador. Confirme que sus DAG, complementos y requisitos funcionan correctamente consultando los grupos de registros correspondientes en CloudWatch. Consulte Monitorización y métricas para obtener más información.

Tareas

En el siguiente tema, se describen los errores que puede encontrar al realizar tareas de Apache Airflow en un entorno.

Veo que mis tareas están bloqueadas o no terminan

Si sus tareas de Apache Airflow están “bloqueadas” o no se completan, le recomendamos que siga los siguientes pasos:

1. Es posible que haya definido una gran cantidad de DAG. Reduzca el número de DAG y actualice el entorno (por ejemplo, cambiando el nivel de registro) para forzar un restablecimiento.

    a. Airflow analiza los DAG independientemente de si están activados o no. Si usa más del 50 % de la capacidad de su entorno, puede que el programador de Apache Airflow empiece a sobrecargarse. Esto se traduce en un tiempo de análisis total elevado en CloudWatch Metrics o en tiempos de procesamiento de DAG prolongados en CloudWatch Logs. Hay otras formas de optimizar las configuraciones de Apache Airflow que no se tratan en esta guía.

    b. Para obtener más información sobre las prácticas recomendadas para ajustar el rendimiento de su entorno, consulte the section called “Ajuste del rendimiento de Apache Airflow”.

2. Es posible que haya una gran cantidad de tareas en la cola. Esto suele aparecer como un número elevado (y creciente) de tareas con el estado `None` o como un número elevado en `Queued Tasks` y `Tasks Pending` en CloudWatch. Esto puede producirse por varias razones:

    a. Si hay más tareas pendientes de ejecución de las que el entorno puede ejecutar o si hay un gran número de tareas en cola antes del escalado automático, tendrá tiempo para detectarlas y desplegar más procesos de trabajo.

    b. Si hay más tareas pendientes de ejecución de las que un entorno puede ejecutar, le recomendamos reducir la cantidad de tareas que sus DAG ejecutan simultáneamente o aumentar el número mínimo de procesos de trabajo de Apache Airflow.

    c. Si hay un gran número de tareas en cola antes de que el escalado automático haya tenido tiempo de detectar y implementar más procesos de trabajo, recomendamos escalonar la implementación de las tareas o aumentar el número mínimo de procesos de trabajo de Apache Airflow.

    d. Use el comando update-environment en la AWS Command Line Interface (AWS CLI) para cambiar la cantidad mínima o máxima de procesos de trabajo que se ejecutan en su entorno.

    ```
    aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-
    workers 10
    ```

e. Para obtener más información sobre las prácticas recomendadas para ajustar el rendimiento de su entorno, consulte the section called “Ajuste del rendimiento de Apache Airflow”.

3. Si sus tareas están bloqueadas en el estado “en ejecución”, también puede borrarlas o marcarlas como ejecutadas o fallidas. Esto permite que el componente de escalado automático de su entorno reduzca verticalmente la cantidad de procesos de trabajo que trabajan en su entorno. En la siguiente imagen, se muestra un ejemplo de tarea pendiente.

   - Elija el círculo para la tarea pendiente y, a continuación, seleccione Borrar como se muestra. Esto permite a Amazon MWAA reducir verticalmente el número de procesos de trabajo; de lo contrario, Amazon MWAA no puede determinar qué DAG están activados o desactivados, ni puede reducirlos verticalmente si todavía hay tareas pendientes.

4. Para más información sobre el ciclo de vida de las tareas de Apache Airflow en Conceptos, consulte la guía de referencia de Apache Airflow.

Recibo errores en las tareas sin registros en Airflow v3

Si sus tareas de Apache Airflow 3 generarn errores sin registros, siga los siguientes pasos:

- Si los registros de procesos de trabajo presentan un error, como `Task handler raised error: WorkerLostError('Worker exited prematurely: exitcode 15 Job: 12.')`, cuando no se completó la tarea, esto indica que es probable que el proceso de trabajo bifurcado asignado a la tarea haya finalizado inesperadamente.

  Para solucionar este problema, considere la posibilidad de configurar celery.worker_autoscale con los mismos valores mínimos y máximos. Por ejemplo:

  ```
  celery.worker_autoscale=5,5  # for mw1.small
  celery.worker_autoscale=10,10 # for mw1.medium
  celery.worker_autoscale=20,20 # for mw1.large
  ```

  Esto garantiza que el tamaño del grupo de procesos de trabajo permanezca fijo, lo que evita la finalización inesperada de procesos de trabajo.

CLI

En el siguiente tema, se describen los errores que puede haber al ejecutar los comandos de la CLI de Airflow en la AWS Command Line Interface.

Me sale el error 503 al activar un DAG en la CLI

La CLI de Airflow se ejecuta en el servidor web de Apache Airflow, que tiene una simultaneidad limitada. Por lo general, se pueden ejecutar un máximo de 4 comandos de CLI simultáneamente.

¿Por qué falla el comando **`dags backfill`** de la CLI de Apache Airflow? ¿Qué solución hay?

> **Note**
>
> Lo siguiente solo se aplica a los entornos Apache Airflow v2.0.2.

El comando `backfill`, como otros comandos de la CLI de Apache Airflow, analiza todos los DAG localmente antes procesar ningún DAG, independientemente del DAG al que se aplique la operación de la CLI. En los entornos de Amazon MWAA que usan Apache Airflow v2.0.2, como los complementos y los requisitos aún no están instalados en el servidor web cuando se ejecuta el comando de la CLI, se produce un error en la operación de análisis y no se invoca la operación `backfill`. Si no tuviera ningún requisito ni complemento en su entorno, la operación `backfill` se realizaría correctamente.

Para poder ejecutar el comando `backfill` de la CLI, recomendamos invocarlo en un operador de bash. En un operador de bash, `backfill` se inicia desde el proceso de trabajo. Esto permite a los DAG analizarlos correctamente, ya que todos los requisitos y complementos necesarios están disponibles e instalados. En el ejemplo siguiente, se muestra cómo crear un DAG con un `BashOperator` para ejecutar `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

En el siguiente tema, se describen los errores que se pueden producir al usar operadores.

Se produjo un error **`PermissionError: [Errno 13] Permission denied`** al utilizar el operador S3Transform

Siga estos pasos si está intentando ejecutar un script de shell con el operador S3Transform y recibe un mensaje de error `PermissionError: [Errno 13] Permission denied`. En los pasos siguientes se supone que ya tiene un archivo plugins.zip. Si va a crear un nuevo plugins.zip, consulte Instalación de complementos personalizados.

1. Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.
2. Cree su script de transformación.

   ```
   #!/bin/bash
   cp $1 $2
   ```

3. (Opcional) Es posible que los usuarios de macOS y Linux tengan que ejecutar el comando siguiente para asegurarse de que el script sea ejecutable.

   ```
   chmod 777 transform_test.sh
   ```

4. Añada el script a su plugins.zip.

   ```
   zip plugins.zip transform_test.sh
   ```

5. Siga los pasos que se indican en Carga del archivo plugins.zip a Amazon S3.
6. Siga los pasos que se indican en Especificación de la versión de plugins.zip en la consola de Amazon MWAA.
7. Utilice los siguientes 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://amzn-s3-demo-bucket/files/input.txt',
    dest_s3_key='s3://amzn-s3-demo-bucket/files/output.txt'
    )
```

8. Siga los pasos que se indican en Carga del código del DAG a Amazon S3.

Solución de problemas: creación y actualización de entornos de Amazon MWAA

En esta página, se describen los errores que puede encontrar al crear y actualizar entornos de Amazon Managed Workflows para Apache Airflow y cómo resolverlos.

Contenido

Actualización de `requirements.txt`

En la siguiente página, se describen los errores que puede encontrar al actualizar su `requirements.txt`.

He especificado una nueva versión de mi `requirements.txt` y la actualización de mi entorno está tardando más de 20 minutos

Si su entorno tarda más de 20 minutos en instalar una nueva versión de un archivo `requirements.txt`, quiere decir que se ha producido un error en la actualización del entorno y Amazon MWAA está volviendo a la última versión estable de la imagen del contenedor.

1. Compruebe las versiones de los paquetes. Recomendamos especificar siempre una versión específica (==) o una versión máxima (<=) para las dependencias de Python en su `requirements.txt`.
2. Compruebe los registros de Apache Airflow. Si ha activado los registros de Apache Airflow, compruebe que los grupos de registros se hayan creado correctamente en la página Grupos de registros de la consola de CloudWatch. Si ve registros en blanco, el motivo más común es la falta permisos en su rol de ejecución para CloudWatch o Amazon S3, donde se escriben los registros. Consulte Rol de ejecución para obtener más información.
3. Compruebe las opciones de configuración de Apache Airflow. Si utiliza Secrets Manager, compruebe que los pares clave-valor que especificó como opción de configuración de Apache Airflow se hayan configurado correctamente. Consulte the section called “Configuración de Secrets Manager” para obtener más información.
4. Compruebe la configuración de la red de VPC. Consulte the section called “Estado atascado” para obtener más información.

5. Compruebe los permisos de los roles de ejecución. Un rol de ejecución es un rol AWS Identity and Access Management (de IAM) con una política de permisos que otorga a Amazon MWAA permiso para invocar los recursos de otros servicios de AWS (como Amazon S3, CloudWatch, Amazon SQS o Amazon ECR) en su nombre. También se debe permitir el acceso a su clave administrada por el cliente o su clave propiedad de AWS. Consulte Rol de ejecución para obtener más información.
6. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.

Complementos

En la siguiente página, se describen los problemas que pueden surgir al configurar o actualizar los complementos de Apache Airflow.

¿Admite Amazon MWAA la implementación de una interfaz de usuario personalizada?

A partir de la versión 2.2.2 de Apache Airflow, Amazon MWAA admite la instalación de complementos en el servidor web de Apache Airflow y la implementación de una UI personalizada. Si su entorno Amazon MWAA ejecuta Apache Airflow v2.0.2 o una versión anterior, no podrá implementar una interfaz de usuario personalizada.

Para obtener más información sobre la administración de versiones y la actualización de sus entornos existentes, consulte Versiones.

Creación de un bucket

En la siguiente página, se describen los errores que pueden generarse al crear un bucket de Amazon S3.

No puedo seleccionar la opción de bloqueo de acceso público en S3

El rol de ejecución de su entorno de Amazon MWAA necesita permiso para realizar la acción de `GetBucketPublicAccessBlock` en el bucket de Amazon S3 para verificar que el bucket ha bloqueado el acceso público. Recomendamos los siguientes pasos:

1. Siga los pasos para adjuntar una política de JSON a su rol de ejecución.
2. Adjunte la siguiente política JSON:

```
{
  "Effect":"Allow",
  "Action":[
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Resource":[
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ]
}
```

Sustituya los marcadores de posición de ejemplo en *amzn-s3-demo-bucket* por el nombre del bucket de Amazon S3.

3. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.

Creación de un entorno

En la siguiente página, se describen los errores que pueden generarse al crear un entorno.

Intenté crear un entorno, pero se atascó en el estado **`Creating`**

Recomendamos los siguientes pasos:

1. Compruebe la red de VPC con enrutamiento público. Si usa una Amazon VPC con acceso a Internet, compruebe lo siguiente:

   - Que su Amazon VPC esté configurada para permitir el tráfico de red entre los distintos recursos de AWS que utiliza su entorno de Amazon MWAA, tal y como se define en the section called “Acerca de las redes”. Por ejemplo, su grupo de seguridad de VPC debe permitir todo el tráfico en una regla de autorreferencia o, si lo desea, especificar el rango de puertos para el rango de puertos HTTPS 443 y un rango de puertos TCP 5432.

2. Compruebe la red de VPC con enrutamiento privado. Si usa una Amazon VPC sin acceso a Internet, compruebe lo siguiente:

- Que su Amazon VPC esté configurada para permitir el tráfico de red entre los distintos recursos de AWS de su entorno Amazon MWAA, tal y como se define en the section called “Acerca de las redes”. Por ejemplo, sus dos subredes privadas no deben tener una tabla de enrutamiento a una puerta de enlace de NAT (o instancia de NAT) ni a una puerta de enlace de Internet.

3. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.

Intenté crear un entorno, pero aparece el estado como **`Create failed`**

Recomendamos los siguientes pasos:

1. Compruebe la configuración de la red de VPC. Consulte the section called “Estado atascado” para obtener más información.
2. Compruebe los permisos de usuario. Amazon MWAA realiza una prueba con las credenciales de un usuario antes de crear un entorno. Es posible que su Cuenta de AWS no tenga permiso en AWS Identity and Access Management (IAM) para crear algunos de los recursos de un entorno. Por ejemplo, si eligió el modo de acceso de red privada para Apache Airflow, su administrador debe haber concedido acceso a su Cuenta de AWS a la política de control de acceso de AmazonMWAAFullConsoleAccess para su entorno, que permite a su cuenta crear puntos de conexión de VPC.
3. Compruebe los permisos de los roles de ejecución. Un rol de ejecución es un rol AWS Identity and Access Management (de IAM) con una política de permisos que otorga a Amazon MWAA permiso para invocar los recursos de otros servicios de AWS (como Amazon S3, CloudWatch, Amazon SQS o Amazon ECR) en su nombre. También se debe permitir el acceso a su clave administrada por el cliente o su clave propiedad de AWS. Consulte Rol de ejecución para obtener más información.
4. Compruebe los registros de Apache Airflow. Si ha activado los registros de Apache Airflow, compruebe que los grupos de registros se hayan creado correctamente en la página Grupos de registros de la consola de CloudWatch. Si ve registros en blanco, el motivo más común es la falta permisos en su rol de ejecución para CloudWatch o Amazon S3, donde se escriben los registros. Consulte Rol de ejecución para obtener más información.

5. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.
6. Si utiliza una Amazon VPC sin acceso a Internet, asegúrese de haber creado un punto de conexión de puerta de enlace de Amazon S3 y de haber concedido los permisos mínimos necesarios a Amazon ECR para acceder a Amazon S3. Para obtener más información sobre cómo crear un punto de conexión de puerta de enlace de Amazon S3, consulte los siguientes recursos:

   - Creación de una red Amazon VPC sin acceso a Internet
   - Creación del punto de conexión de la puerta de enlace de Amazon S3 en la Guía del usuario de Amazon Elastic Container Registry

Intenté seleccionar una VPC y se produjo el error **`Network Failure`**

Recomendamos los siguientes pasos:

- Si aparece el error `Network Failure` al intentar seleccionar una Amazon VPC cuando se crea el entorno, desactive todos los proxies del navegador que se estén ejecutando e inténtelo de nuevo.

He intentado crear un entorno y he recibido un mensaje de error que indica que el servicio, la partición o el recurso “deben pasarse”

Recomendamos los siguientes pasos:

- Es posible que reciba este error porque el URI que especificó para su bucket de Amazon S3 incluye una barra oblicua (“/”) al final del URI. Le recomendamos que elimine la barra inclinada (“/”) de la ruta. El valor debe tener el siguiente formato:

  ```
  s3://amzn-s3-demo-bucket
  ```

Intenté crear un entorno que muestra el estado como **`Available`**, pero cuando intento acceder a la UI de Airflow aparece el mensaje de error **`Empty Reply from Server`** o **`502 Bad Gateway`**.

Recomendamos los siguientes pasos:

1. Compruebe la configuración del grupo de seguridad de VPC. Consulte the section called “Estado atascado” para obtener más información.
2. Confirme que todos los paquetes de Apache Airflow que haya publicado en `requirements.txt` corresponden a la versión de Apache Airflow que utiliza en Amazon MWAA. Consulte Instalación de dependencias de Python para obtener más información.
3. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.

Intenté crear un entorno y mi nombre de usuario es un conjunto de nombres de caracteres aleatorios

- Apache Airflow tiene un máximo de 64 caracteres para los nombres de usuario. Si su rol AWS Identity and Access Management (de IAM) supera esta longitud, se utilizará un algoritmo de hash para reducirla y, para que, al mismo tiempo, siga siendo única.

Actualización de entornos

En la siguiente página, se describen los errores que se pueden producir al actualizar un entorno.

Intenté cambiar la clase del entorno, pero la actualización falló

Si actualiza su entorno a una clase de entorno diferente (por ejemplo, al cambiar de un `mw1.medium` a un `mw1.small`) y la solicitud de actualización no se puede realizar correctamente, el estado del entorno pasará a `UPDATE_FAILED` y el entorno se revertirá a su versión estable anterior y se facturará de acuerdo con ella.

Recomendamos los siguientes pasos:

1. Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

2. Para ejecutar un script de solución de problemas que compruebe la configuración de la red de Amazon VPC para su entorno Amazon MWAA, consulte el script Verify Environment (Verificar entorno) en las herramientas de soporte de AWS en GitHub.

Acceso a entornos

En la siguiente página, se describen los errores que se pueden producir al acceder un entorno.

No puedo acceder a la interfaz de usuario de Apache Airflow

Recomendamos los siguientes pasos:

1. Compruebe los permisos de usuario. Es posible que no se le haya concedido acceso a una política de permisos que le permita ver la UI de Apache Airflow. Consulte the section called "Acceso a un entorno de Amazon MWAA" para obtener más información.
2. Compruebe el acceso a la red. Esto puede deberse a que seleccionó el modo de acceso de red privada. Si la URL de la UI de Apache Airflow tiene el siguiente formato `387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-west-2.airflow.amazonaws.com`, significa que está usando un enrutamiento privado para su servidor web de Apache Airflow. Puede actualizar el modo de acceso de Apache Airflow al modo de acceso de red pública o crear un mecanismo para acceder al punto de conexión de VPC de su servidor web de Apache Airflow. Consulte the section called "Administración del acceso a puntos de conexión de VPC" para obtener más información.

Solución de problemas: errores de CloudWatch Logs y CloudTrail

En esta página, se describen soluciones a errores de registros de Amazon CloudWatch y AWS CloudTrail que puede encontrar en un entorno de Amazon Managed Workflows para Apache Airflow.

Contenido

- Me sale Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory en los registros de Apache Airflow
- Con respecto a psycopg2, me aparece “server closed the connection unexpectedly” (el servidor cerró la conexión de forma inesperada) en mis registros de programador
- Me sale Executor reports task instance %s finished (%s) although the task says its %s en los registros de procesamiento de DAG
- Me sale Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log. en los registros de tareas

Registros

En la siguiente página, se describen los errores que pueden aparecer cuando vea registros de Apache Airflow.

No puedo encontrar mis registros de tareas o recibí el mensaje de error **`Reading remote log from Cloudwatch log_group`**

Amazon MWAA ha configurado Apache Airflow para que lea y escriba registros directamente desde y hacia Registros de Amazon CloudWatch. Si un proceso de trabajo no puede iniciar una tarea o no puede escribir ningún registro, aparecerá el siguiente error:

```
*** 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 los siguientes pasos:

  a. Compruebe que han habilitado los registros de tareas en el nivel `INFO` de su entorno. Para obtener más información, consulta Visualización de registros en Amazon CloudWatch.

  b. Compruebe que el rol de ejecución del entorno tenga las políticas de permisos correctas.

  c. Compruebe que el operador o la tarea funcionen correctamente, que cuenten con recursos suficientes para analizar el DAG y que tengan las bibliotecas de Python adecuadas para cargarse. Para comprobar si tiene las dependencias correctas, intente eliminar las importaciones hasta que encuentre la que está causando el problema. Recomendamos que pruebe sus dependencias de Python con aws-mwaa-docker-images.

Las tareas dan error sin ningún registro

Si las tareas dan error en un flujo de trabajo y no encuentra ningún registro de las tareas que generan errores, compruebe si el parámetro `queue` está configurado en sus argumentos predeterminados, como se muestra a continuación.

```
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 el problema, elimine `queue` del código y vuelva a invocar el DAG.

Me sale el error **`ResourceAlreadyExistsException`** en CloudTrail

```
"errorCode": "ResourceAlreadyExistsException",
    "errorMessage": "The specified log stream already exists",
    "requestParameters": {
        "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
        "logStreamName": "scheduler_cross-account-eks.py.log"
    }
```

Algunos requisitos de Python, como `apache-airflow-backport-providers-amazon`, revierten la biblioteca `watchtower` que Amazon MWAA utiliza para comunicarse con CloudWatch a una versión anterior. Recomendamos los siguientes pasos:

- Añada lo siguiente a su archivo `requirements.txt`

  ```
  watchtower==1.0.6
  ```

Me sale el error **`Invalid request`** en CloudTrail

```
Invalid request provided: Provided role does not have sufficient permissions for s3
 location airflow-xxx-xxx/dags
```

Si va a crear un entorno de Amazon MWAA y un bucket de Amazon S3 con la misma plantilla de CloudFormation, tendrá que añadir una sección `DependsOn` dentro de la plantilla de CloudFormation. Los dos recursos (el entorno de MWAA y la política de ejecución de MWAA) dependen de CloudFormation. Recomendamos los siguientes pasos:

- Añada la siguiente declaración **`DependsOn`** a su plantilla de 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 un ejemplo, consulte Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow.

Me sale **`Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory`** en los registros de Apache Airflow

- Recomendamos los siguientes pasos:
  - Si utiliza Apache Airflow v2, agregue `core.lazy_load_plugins : False` como opción de configuración de Apache Airflow. Para obtener más información, consulte Uso de las opciones de configuración para cargar complementos en la versión 2.

Con respecto a psycopg2, me aparece “server closed the connection unexpectedly” (el servidor cerró la conexión de forma inesperada) en mis registros de programador

Si aparece un error similar al siguiente, es posible que el programador de Apache Airflow se haya quedado sin 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 los siguientes pasos:

- Valore la posibilidad de actualizar a Apache Airflow v2.0.2, donde podrá especificar hasta 5 programadores.

Me sale **`Executor reports task instance %s finished (%s) although the task says its %s`** en los registros de procesamiento de DAG

Si le sale un error similar al siguiente, es posible que sus tareas de larga ejecución hayan alcanzado el límite de tiempo de tareas en Amazon MWAA. Amazon MWAA tiene un límite de 12 horas para cada tarea de Airflow, a fin de evitar que las tareas se queden atascadas en la cola y bloqueen actividades como el escalado automático.

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
 %s) Was the task killed externally
```

Recomendamos los siguientes pasos:

- Valore la posibilidad de dividir la tarea en varias tareas de ejecución más cortas. Airflow suele tener un modelo en el que los operadores son asíncronos. Invoca actividades en sistemas externos y los sensores de Apache Airflow llevan a cabo un sondeo para ver cuándo se han completado. Si un sensor falla, se puede volver a probar de forma segura sin que hacerlo afecte a la funcionalidad del operador.

Me sale **`Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.`** en los registros de tareas

Si aparece un error similar al siguiente, es posible que el rol de ejecución de su entorno no contenga una política de permisos para crear flujos de registro para los registros de tareas.

```
Could not read remote logs from log_group: airflow-*{*environmentName}-Task
 log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

Recomendamos los siguientes pasos:

- Modifique el rol de ejecución de su entorno mediante una de las políticas de ejemplo que se muestran en the section called “Rol de ejecución”.

Es posible que también haya especificado un paquete de proveedores en el archivo `requirements.txt` que no sea compatible con su versión de Apache Airflow. Por ejemplo, si usa Apache Airflow v2.0.2, es posible que haya especificado un paquete, por ejemplo, apache-airflow-providers-databricks que solo es compatible con Airflow 2.1 y versiones posteriores.

Recomendamos los siguientes pasos:

1. Si utiliza Apache Airflow v2.0.2, modifique el archivo `requirements.txt` y añada `apache-airflow[databricks]`. Así se instala la versión correcta del paquete de Databricks compatible con Apache Airflow v2.0.2.
2. Pruebe sus DAG, complementos personalizados y dependencias de Python de forma local con aws-mwaa-docker-images en GitHub.

Historial de revisión de la guía de usuario de Amazon MWAA

En la siguiente tabla se describen los cambios importantes de la documentación de Amazon MWAA, a partir de noviembre de 2020. Para recibir notificaciones sobre las actualizaciones de esta documentación, suscríbase a la fuente RSS.

Cambio	Descripción	Fecha
Se agregó compatibilidad con Apache Airflow v3	Se actualizó la documentación para incluir compatibilidad con Apache Airflow v3. • Ejemplos de código • Versiones	1 de octubre de 2025
Actualizaciones de IPv6	Se agregó información sobre la compatibilidad con IPv6. • the section called “Acerca de las redes”	26 de agosto de 2025
Actualización de entornos	Se agregó una nota sobre el cambio de `workerReplacementStrategy` de `GRACEFUL` a `FORCED` si se realiza una actualización mientras el entorno está en estado `MAINTENANCE` . • the section called “Actualización de un entorno”	6 de agosto de 2025
Información de baja de versiones	Se actualizó el tema sobre la obsolescencia de la versión para incluir los avisos y plazos de caducidad de Apache	24 de junio de 2025

	Airflow v2.4.3, Apache Airflow v2.5.1 y Apache Airflow v2.6.3. • the section called “Versiones obsoletas de Apache Airflow”	
Se agregó una nueva clase de entorno: mw1.micro	Amazon MWAA ahora ofrece una nueva clase de entorno: mw1.micro. • the section called “Configuración de la clase de entorno” • the section called “Ajuste del rendimiento de Apache Airflow”	19 de noviembre de 2024
Compatibilidad con un método más sencillo de acceder a la API de REST de Apache Airflow	Amazon MWAA ahora ofrece un enfoque simplificado para interactuar con la API de REST de Apache Airflow mediante credenciales de AWS. • the section called “Uso de la API de REST de Apache Airflow” • the section called “Acceso a la API de REST de Apache Airflow”	23 de octubre de 2024

Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.10.1. En esta actualización, se incluye información sobre los paquetes de proveedores actualizados y detalles sobre el uso de Apache Airflow v2.10.1 en Amazon MWAA. • Versiones • the section called “Paquetes de proveedores específicos para cada versión”	26 de septiembre de 2024
Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.9.2. En esta actualización, se incluye información sobre los paquetes de proveedores actualizados y detalles sobre el uso de Apache Airflow v2.9.2 en Amazon MWAA. • Versiones • the section called “Paquetes de proveedores específicos para cada versión”	9 de julio de 2024

Amazon MWAA es compatible con la configuración de nombres de dominio de un servidor web personalizado	Amazon MWAA permite configurar nombres de dominio de un servidor web personalizado para entornos privados sin acceso a Internet. Esta actualización incluye el siguiente tema nuevo que describe cómo configurar un nuevo dominio personalizado. • the section called “Configuración de un dominio personalizado”	18 de junio de 2024
Amazon MWAA es compatible con el escalado automático de servidores web y la API de REST de Apache Airflow	Amazon MWAA ahora permite el escalado automático de servidores web y brinda la capacidad de acceder a la API de REST de Apache Airflow y usarla. • the section called “Configuración del escalado automático del servidor web” • the section called “Uso de la API de REST de Apache Airflow”	16 de mayo de 2024

Descripción mejorada del comportamiento de escalado automático	Se actualizó el siguiente tema para reflejar el nuevo comportamiento de escalado automático de Amazon MWAA cuando los procesos de trabajo asumen nuevas tareas a medida que los procesos de trabajo de Fargate se reducen. • the section called “Configuración del escalado automático de los procesos de trabajo”	10 de mayo de 2024
Compatibilidad con tamaños de instancia más grandes	Amazon MWAA ahora es compatible con dos opciones de tamaños de instancia más grandes para cargas de trabajo más grandes: `mw1.xlarge` y `mw1.2xlarge`. • the section called “Capacidades del entorno”	16 de abril de 2024

Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.8.1. En esta actualización, se incluye información sobre los paquetes de proveedor es actualizados y detalles sobre el uso de Apache Airflow v2.8.1 en Amazon MWAA. • Versiones • the section called “Paquetes de proveedores específicos para cada versión”	22 de febrero de 2024
Compatible con Amazon VPC compartido	Amazon MWAA admite la creación de entornos multicuentas para las organizaciones que utilizan Amazon OpenSearch Service para gestionar los recursos de Amazon MWAA mediante una Amazon VPC central compartida en una cuenta de propietario. Como parte de este lanzamiento, Amazon MWAA le permite elegir crear y administrar sus propios puntos de conexión de Amazon VPC. • the section called “Administración de sus propios puntos de conexión de Amazon VPC”	15 de noviembre de 2023

Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.7.2. En esta actualización, se incluye información sobre los paquetes de proveedores actualizados y detalles sobre el uso de Apache Airflow v2.7.2 en Amazon MWAA. • Versiones • the section called “Paquetes de proveedores específicos para cada versión”	6 de noviembre de 2023
Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.6.3. En esta actualización, se incluye información sobre los paquetes de proveedores actualizados y detalles sobre el uso de Apache Airflow v2.6.3 en Amazon MWAA. • Versiones	9 de agosto de 2023
Información de baja de versiones	Se actualizó el tema sobre la obsolescencia de la versión para incluir los avisos y plazos de caducidad de Apache Airflow v2.0.2 y Apache Airflow v2.2.2. • the section called “Versiones obsoletas de Apache Airflow”	31 de julio de 2023

Nuevos temas y casos de uso	Amazon MWAA admite la actualización a versiones menores. Esta actualización incluye el siguiente tema nuevo en el que se describe cómo actualizar el entorno y asegurarse de que los recursos del flujo de trabajo sean compatibles con la versión de Apache Airflow a la que se va a actualizar: • the section called “Cambio de versión”	5 de junio de 2023
Tema actualizado	Se actualizaron las políticas de IAM administradas por el cliente que otorgan al usuario acceso completo a Amazon MWAA desde la consola y la API. En la actualiza ción, se describe por qué debe proporcionar permiso para `iam:PassRole` , de modo que un usuario pueda transferir roles a Amazon MWAA. Amazon MWAA utiliza estos permisos para realizar acciones en nombre de un usuario. • the section called “Acceso a un entorno de Amazon MWAA”	12 de abril de 2023

Nueva guía	Se ha actualizado el tema sobre la configuración AWS Secrets Manager como backend de Amazon MWAA a fin de proporcionar orientación sobre el uso de los patrones de búsqueda. El uso de patrones de búsqueda reduce los secretos que busca Apache Airflow y reduce el número de llamadas a la API que Amazon MWAA realiza a Secrets Manager para recuperar conexione s y variables. Esto reduce los costes asociados al uso de Secrets Manager como backend. • Creación del backend de Secrets Manager como opción de configuración de Apache Airflow	12 de abril de 2023
Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.5.1. En esta actualización, se incluye información sobre los paquetes de proveedores actualizados y detalles sobre el uso de Apache Airflow v2.5.1 en Amazon MWAA. • Versiones	11 de abril de 2023

Nuevos temas y casos de uso	Se ha añadido un tema nuevo sobre el uso de un script de inicio en un entorno Amazon MWAA. En este tema se describe la configuración de un script de inicio para un entorno existente, su uso para instalar tiempos de ejecución de Linux y la configuración de las variables de entorno. • the section called “Uso de un script de inicio”	3 de abril de 2023
Se ha actualizado la sección sobre el acceso a servidores web privados	Se actualizó el siguiente tema sobre el acceso a servidores web privados. En la actualización, se aclara que en entornos con acceso a un servidor web privado, debe usar un archivo wheel de Python (`.whl`) para empaquetar e instalar las dependencias. • Modo de acceso a un servidor web privado	24 de febrero de 2023

Se añadió información sobre las versiones obsoletas de Apache Airflow	Se actualizó el tema de versiones con nueva información sobre cómo Amazon MWAA administró las versiones obsoletas de Apache Airflow. Se eliminó una sección sobre la actualización a una versión más reciente de Apache Airflow y una sección que describía los cambios entre Apache Airflow v1 y Apache Airflow v2. Para obtener más información sobre cómo migrar a una versión más reciente de Apache Airflow, consulte la guía de migración a Amazon MWAA. • the section called “Versiones obsoletas de Apache Airflow” • the section called “Preguntas frecuentes y compatibilidad de Apache Airflow”	17 de febrero de 2023

Correcciones en las métricas de contenedores de Amazon MWAA	Se actualizó el tema de las métricas de contenedores y se eliminó un conjunto de métricas erróneas que no existían en la dimensión `Cluster`. Se añadió una sección adicional en la que se describe cómo se puede evaluar la cantidad de procesos de trabajo adicionales que utiliza un entorno en un momento dado; para ello, se representa gráficamente la métrica `CPUUtilization` o la métrica `MemoryUtilization` para el componente `AdditionalWorker` y se establece el tipo de estadística en `SampleCount`. • the section called “Evaluación de la cantidad de contenedores adicionales para procesos de trabajo y servidores web”	20 de enero de 2023

Nueva versión de Apache Airflow	Amazon MWAA ahora es compatible con Apache Airflow v2.4.3. Esta actualización incluye información sobre los paquetes de proveedores actualizados, detalles sobre el uso de Apache Airflow v2.4.3 en Amazon MWAA e información consolidada sobre las características compatibles con cada versión de Apache Airflow en Amazon MWAA. • Versiones	5 de enero de 2023
Tema actualizado sobre el rol vinculado a servicio	Información actualizada sobre el rol vinculado al servicio que Amazon MWAA utiliza para crear y gestionar recursos de AWS en su nombre, incluida información sobre cómo puede eliminar el rol vinculado al servicio cuando ya no lo necesite. Esto incluye una política actualizada de permisos de roles vinculados a servicios que permite a Amazon MWAA publicar métricas adicionales de CloudWatch en el espacio de nombres `AWS/MWAA`. • the section called “Rol vinculado a servicios”	18 de noviembre de 2022

Nuevo tema sobre las métricas de los servicios	Se ha añadido un tema nuevo, en el que se describen las métricas de servicio emitidas por Amazon MWAA en el espacio de nombres `AWS/MWAA`. Entre ellas, se incluyen los programadores de métricas de clústeres, los procesos de trabajo y los servidores web de Amazon ECS, las métricas de Amazon SQS para las colas que permiten a Amazon MWAA desvincular a los programad ores de los procesos de trabajo, así como las métricas de Amazon RDS para la base de datos de metadatos. • the section called “Métricas de contenedores, colas y bases de datos”	18 de noviembre de 2022
Nuevo tema	Se ha añadido una nueva guía sobre la modificación de un archivo de restricci ones para especificar nuevas versiones de los paquetes de los proveedores para utilizarl as en su entorno de Amazon MWAA. • the section called “Archivo de restricciones”	18 de noviembre de 2022

Entrada de preguntas frecuentes actualizada	Información actualizada relacionada con la aptitud de Amazon MWAA para cumplir con la HIPAA. • the section called “Conformidad con HIPAA”	15 de noviembre de 2022
Nuevo tema	Se ha añadido un nuevo tema sobre el uso de `aws:SourceArn` y las claves de condición de contexto global `aws:SourceAccount` en una política de confianza para los roles de ejecución de Amazon MWAA, con el fin de evitar confundir a los suplentes de un servicio a otro. • the section called “Prevención de la sustitución confusa entre servicios”	21 de octubre de 2022
Nueva muestra de código	Se añadieron instrucciones actualizadas y un código de muestra DAG que escribe métricas personalizadas a nivel de sistema operativo en CloudWatch. • the section called “Uso de un DAG para escribir métricas personalizadas”	13 de septiembre de 2022

Nueva muestra de código	Se añadieron instrucciones actualizadas y un nuevo código de muestra Python AWS Lambda que recupera un token CLI de Apache Airflow y, a continuación, invoca un DAG en un entorno Amazon MWAA específico. • the section called “Invocación de DAG con Lambda”	12 de septiembre de 2022
Nuevos diagramas de arquitectura	Se añadieron nuevos diagramas de arquitectura que muestran un entorno de Amazon MWAA con un servidor web público y privado. • the section called “Modos de acceso de Apache Airflow”	12 de septiembre de 2022
Nueva muestra de código	Se añadieron instrucciones actualizadas y un nuevo código de muestra de DAG que recupera un token CLI de Apache Airflow y, a continuación, invoca otro DAG en un entorno Amazon MWAA diferente. • the section called “Invocación de DAG en diferentes entornos”	16 de agosto de 2022

Nueva muestra de código	Se añadieron instrucciones actualizadas y un nuevo DAG que consulta la informaci ón de metadatos de Aurora PostgreSQL de un entorno, escribe el resultado en archivos CSV y los almacena en Amazon S3. • the section called “Exportac ión de metadatos del entorno a Amazon S3”	12 de agosto de 2022
Nueva muestra de código	Se añadieron instrucciones actualizadas y un nuevo DAG que actualiza un token AWS CodeArtifacten tiempo de ejecución y almacena el resultado en Amazon S3. • the section called “Actualizar un token AWS CodeArtifact en tiempo de ejecución”	3 de agosto de 2022
Nueva muestra de código	Se añadieron instrucciones actualizadas y un código de muestra DAG para el uso de `ECSOperator` en Amazon MWAA. • the section called “Uso de `ECSOperator` ”	26 de julio de 2022

Nueva muestra de código	Se añadieron instrucciones actualizadas y un código de muestra DAG para el uso de `SSHOperator` en Amazon MWAA. • the section called “Uso de `SSHOperator`”	15 de julio de 2022
Nueva muestra de código	Se añadieron nuevas instrucciones y un código de muestra DAG para usar Postgres de dbt con Amazon MWAA. • the section called “Uso de dbt con Amazon MWAA”	17 de junio de 2022
Nuevos temas y casos de uso	Se añadieron nuevas instrucciones y un código de muestra DAG para instalar dependencias mediante archivos de rueda de Python para entornos Amazon MWAA con acceso público y privado. • Gestión de dependencias mediante ruedas de Python	13 de mayo de 2022
Nuevos temas y casos de uso	Se ha añadido una nueva guía sobre la elección de las métricas de Apache Airflow que Amazon MWAA envía a CloudWatch. • Cómo elegir qué métricas de Apache Airflow se van a registrar	19 de abril de 2022

Nuevas guías	Amazon MWAA ofrece una guía de migración para migrar los flujos de trabajo de Apache Airflow desde implement aciones autogestionadas, así como desde los entornos de Amazon MWAA existentes. • Guía de migración a Amazon MWAA	7 de marzo de 2022
Nuevos temas y casos de uso	Se añadieron nuevas prácticas recomendadas de seguridad para trabajar con Apache Airflow, incluida una solución para detectar cambios en los privilegios de usuario de Apache Airflow. • the section called “Prácticas recomendadas de seguridad en Apache Airflow”	18 de febrero de 2022
Nueva muestra de código	Se añadió un nuevo código de muestra para crear DAG con reconocimiento de zonas horarias mediante Pendulum y se aclaró cómo usar un complemento personali zado para cambiar la zona horaria en la que se crean los registros de Apache Airflow. • the section called “Cómo cambiar la zona horaria de un DAG”	11 de febrero de 2022

Lanzamiento de Apache Airflow v	Amazon Managed Workflows para Apache Airflow ahora es compatible con Apache Airflow v2.2.2. A partir de la versión 2.2, Amazon MWAA instalará paquetes de Python y complementos personalizados directamente en el servidor web Apache Airflow, lo que le permitirá administrar sus entornos con mayor flexibilidad. Para obtener más información, consulte los siguientes recursos. • Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow. • Registro de cambios de Apache Airflow v2.2.2 en el sitio web de documentación de Apache Airflow.	27 de enero de 2022

Nuevos tutoriales	Se añadió un nuevo tutorial en el que se muestra cómo crear un nuevo rol personalizado de Apache Airflow y cómo asignar el rol a un usuario de Apache Airflow definido desde IAM para limitar el acceso del usuario a un subconjunto de DAG específicos. • Tutorial: Restricciones de acceso de los usuarios de Amazon MWAA a un subconjunto de DAG	8 de diciembre de 2021

Correcciones	Se ha corregido una de las prácticas recomendadas para establecer el valor de `scheduler.min_file_process_interval` con el fin de optimizar el uso de la CPU. Se añadió un ejemplo de política de IAM que otorga acceso a los recursos de Secrets Manager en el rol de ejecución. Se añadió un tema de solución de problemas sobre el uso de las claves de condición de Secrets Manager. • Ajuste del rendimiento: la forma en que el programador analiza los DAG • Otorgue permiso a Amazon MWAA para acceder a las claves secretas de Secrets Manager • Configuración de claves de condición en el rol de ejecución de Amazon MWAA para Secrets Manager	22 de noviembre de 2021

Nueva muestra de código	Se añadió el siguiente código de muestra nuevo para modificar la zona horaria en la que se procesan los DAG mediante un complemento personalizado y un nuevo tema de solución de problemas para invocar el comando de la CLI de Apache Airflow `dags backfill` desde un operador bash. • the section called “Cómo cambiar la zona horaria de un DAG” • Comando backfill de la CLI mediante un operador de bash	1 de noviembre de 2021
Correcciones	Se corrigieron problemas en el código de muestra de operador de Amazon ECS y se aclararon los permisos adicionales necesarios en el rol de ejecución de Amazon MWAA para permitir que el entorno accediera al grupo de registro de tareas de Amazon ECS en CloudWatch Logs. • Permisos de operador de Amazon ECS.	26 de octubre de 2021

Nueva muestra de código	Se añadió un nuevo código de muestra que consulta la base de datos Aurora PostgreSQL para obtener información relevante sobre las ejecuciones de DAG y escribe los resultados en un archivo CSV almacenado en Amazon S3. • the section called “Exportación de metadatos del entorno a Amazon S3”.	1 de octubre de 2021
Correcciones	Se ha corregido la información sobre cómo Amazon MWAA sincroniza automáticamente los objetos nuevos y modificados del bucket de Amazon S3 de destino con los programadores y los procesos de trabajo. • Cómo funciona la carpeta DAG.	1 de octubre de 2021
Ahora es compatible	Amazon MWAA ahora admite paquetes de proveedores adicionales para Apache Airflow 2.0+. Para obtener más información sobre los paquetes compatibles, consulte los siguientes recursos:	24 de septiembre de 2021

Nuevos comandos y procedimientos	Se añadieron instrucciones y ejemplos de comandos de la AWS CLI adicionales para crear un punto de conexión de puerta de enlace de Amazon S3 cuando se utiliza una Amazon VPC sin acceso a Internet: • Creación de una red Amazon VPC sin acceso a Internet.	24 de septiembre de 2021
Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha añadido un nuevo código de muestra que utiliza un operador de Amazon Elastic Container Service en the section called “Uso de `ECSOperator` ”. • Se añadieron nuevos temas de solución de problemas relacionados con la configuración de los complementos de Apache Airflow en the section called “Complementos”.	19 de septiembre de 2021

Compatibilidad con nueva región	Amazon MWAA ya está disponible en las siguientes regiones: • Asia-Pacífico (Bombay): ap-south-1 • Asia-Pacífico (Seúl): ap-northeast-2 • Europa (Londres): eu-west-2 • Europa (París): eu-west-3 • Canadá (centro): ca-central-1 • América del Sur (São Paulo): sa-east-1 Para obtener más información sobre la disponibilidad en las regiones y los puntos de conexión de servicio, consulte los siguientes recursos: • Puntos de conexión y cuotas de Amazon MWAA en la Referencia general de AWS.	31 de agosto de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se actualizaron las políticas de muestra para permitir que Amazon MWAA recupere la configuración de Amazon S3 a nivel de cuenta (`s3:GetAccountPublicAccessBlock`) en Rol de ejecución de Amazon MWAA.	27 de agosto de 2021
Correcciones	Se añadieron los siguientes cambios: • Se corrigió la plantilla de CloudFormation para usar una regla de entrada con autorreferencia para el grupo de seguridad en Creación de la red de VPC. • Se corrigió la plantilla de CloudFormation para usar una regla de entrada con autorreferencia para el grupo de seguridad en Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow.	27 de agosto de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió el decorador DAG a la lista de elementos compatibles con Apache Airflow v2.0.2 Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow.	20 de agosto de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió un caso de uso `celery.sync_parallelism` a Ajuste del desempeño de Apache Airflow en Amazon MWAA. • Se añadieron puntos de conexión de servicio a la página de cuotas y se cambió el nombre a Puntos de conexión y cuotas de servicio de Amazon Managed Workflows para Apache Airflow. • Se aclararon los requisitos previos de red en función de los comentarios de los usuarios en Introducción a Amazon Managed Workflows para Apache Airflow. • Se movió `dags list-runs` y `dags next-execution` a comandos de la CLI de Airflow no compatibles en Referencia de los comandos de la CLI de Apache Airflow.	13 de agosto de 2021

Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un ejemplo de bash para configura r, obtener o eliminar una variable de Apache Airflow v2.0.2 en Referencia de los comandos de la CLI de Apache Airflow. • Se añadieron las dependenc ias de Apache Airflow v2.0.2 y un ejemplo de conexión de Airflow a Uso de Amazon MWAA con Amazon RDS para Microsoft SQL Server.	13 de agosto de 2021
Correcciones	Se añadieron los siguientes cambios: • Se corrigió el código de muestra de Python basado en los comentarios de los usuarios en Creación de una conexión SSH mediante `SSHOperator` .	13 de agosto de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se trasladó `variables set` a los comandos de la CLI de Airflow compatibles en Referencia de los comandos de la CLI de Apache Airflow. • Se añadió el resumen de los cambios en la versión 2.0.2 de la página de versiones de Airflow para Instalación de dependencias de Python en base a los comentarios de los usuarios. • Se añadió el resumen de los cambios en la versión 2.0.2 de la página de versiones de Airflow para Referencia de los comandos de la CLI de Apache Airflow en base a los comentarios de los usuarios. • Se añadió el resumen de los cambios en la versión 2.0.2 de la página de versiones de Airflow para Información general sobre los tipos de conexión en base a los comentarios de los usuarios. • Se añadió el resumen de los cambios en la versión 2.0.2 de la página de versiones de Airflow para Instalación de complemen	6 de agosto de 2021

	tos personalizados en base a los comentarios de los usuarios. • Se añadió el resumen de los cambios en la versión 2.0.2 de la página de versiones de Airflow para Cómo añadir o actualizar DAG en base a los comentarios de los usuarios.	
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra de Apache Airflow v2.0.2 a Uso de un DAG para importar variables en la CLI. • Se añadió un código de muestra de Apache Airflow v2.0.2 a Invocación de DAG con una función de Lambda.	6 de agosto de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha añadido un tema de solución de problemas para el ítem “No veo mi conexión en la UI de Airflow” en Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se ha añadido una lista de VPC de Amazon compatibles con Amazon MWAA en Acerca de las redes en Amazon MWAA.	29 de julio de 2021

Correcciones	Se añadieron los siguientes cambios: • Se corrigió el código de muestra de Python basado en los comentarios de los usuarios para imprimir el token de inicio de sesión web en Cree un token de acceso al servidor web Apache Airflow. • Se corrigió el problema de conexión con Snowflake , basado en los comentarios de los usuarios, para usar una comilla única para el parámetro de almacén en Solución de problemas de Amazon Managed Workflows para Apache Airflow.	29 de julio de 2021

Temas eliminados o movidos	Se añadieron los siguientes cambios: • Se ha reestructurado la página existente para incluir todas las páginas de documentación sobre monitorización y métricas en Monitorización y métricas de Amazon Managed Workflows para Apache Airflow. • Se trasladó Métricas del entorno de Apache Airflow en CloudWatch al menú de navegación de monitoriz ación y métricas.	23 de julio de 2021
Nuevas guías	Se añadieron los siguientes cambios: • Se creó Paquetes de proveedores de Apache Airflow instalados en entornos Amazon MWAA. • Se creó Información general sobre la monitorización en Amazon MWAA. • Se creó Acceso a los registros de auditoría en AWS CloudTrail. • Se creó Visualización de registros en Amazon CloudWatch.	23 de julio de 2021

Correcciones	Se añadieron los siguientes cambios: • Se corrigió el código de muestra de Python basado en los comentarios de los usuarios para generar una cadena de conexión de Airflow en la secuencia correcta y se añadió el parámetro de puerto en Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager. • Se ha añadido un paso para instalar un paquete de descompresión de forma local en función de los comentarios de los usuarios a Creación de un complemento personalizado con Oracle.	23 de julio de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha añadido un tema para los operadores de DMS de AWS en Preguntas frecuentes sobre Amazon MWAA. • Se añadió un tema de solución de problemas para un error de registro remoto a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se trasladó `variables set` a comandos de la CLI de Airflow no compatibles en Referencia de los comandos de la CLI de Apache Airflow.	16 de julio de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se han añadido pasos secuenciales para crear un archivo requirements.txt en función de los comentarios de los usuarios en Instalación de dependencias de Python. • Se han añadido pasos secuenciales para crear un archivo plugins.zip a partir de los comentarios de los usuarios en Instalación de complementos personalizados. • Se añadieron enlaces de referencia cruzados en la guía del usuario a la guía de referencia de la API de Amazon Managed Workflows para Apache Airflow. • Se ha añadido un tema sobre por qué los complementos no aparecen en el menú “Admin > Plugins” (Administrador > Complementos) de Airflow 2.0 en Preguntas frecuentes sobre Amazon MWAA.	9 de julio de 2021

Nuevas guías	Se añadieron los siguientes cambios: • Se creó Eliminación de archivos en Amazon S3.	9 de julio de 2021
Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una lista de valores admitidos en Uso de claves maestras de cliente para el cifrado. • Se actualizó y aclaró el ejemplo de una URL de repositorio privada en función de los comentarios de los usuarios en Administración de las dependencias de Python en requirements.txt.	2 de julio de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra de Apache Airflow v1.10.12 para usar una clave privada en AWS Secrets Manager para una conexión SSH en Creación de una conexión SSH mediante `SSHOperator`.	2 de julio de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadieron las métricas StartedTaskInstances y FinishedTaskInstances a Métricas del entorno de Apache Airflow en CloudWatch.	25 de junio de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió el código de muestra de Apache Airflow v2.0.2 en Uso de imágenes de Amazon ECR con Amazon EKS.	25 de junio de 2021
Nuevas guías	Se añadieron los siguientes cambios: • Se creó Ajuste del desempeño de Apache Airflow en Amazon MWAA.	25 de junio de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadieron `connections add` y `connections delete` a los comandos de la CLI compatibles de Apache Airflow v2.0.2 en Referencia de los comandos de la CLI de Apache Airflow. • Se añadió que la última versión disponible en CloudFormation es Apache Airflow v2.0.2 a Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow. • Se añadió una pregunta para almacenar datos temporales sobre los procesos de trabajo de Apache Airflow en Preguntas frecuentes sobre Amazon MWAA. • Se ha añadido un tema para el error “Executor reports task instance %s finished” a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se ha añadido un tema para el registro “El servidor ha cerrado la conexión de forma inesperada” a Solución de problemas	18 de junio de 2021

de Amazon Managed Workflows para Apache Airflow.

- Se añadió un ejemplo para ejecutar comandos de la CLI en un túnel SSH a un host bastión. Creación de un token de la CLI de Apache Airflow
- Se añadió un tema para los nombres de usuario generados aleatoriamente a Solución de problemas de Amazon Managed Workflows para Apache Airflow.
- Se añadió un tema para un error 503 al ejecutar un DAG en la CLI para Solución de problemas de Amazon Managed Workflows para Apache Airflow.
- Se añadió un tema para los complementos personalizados en Apache Airflow v2.0.2, que requieren una opción de configuración de Airflow de `core.lazy_load_plugins : False` para cargar los complementos al inicio de cada proceso de Airflow para anular la configuración predeterminada de la versión en Uso de las

opciones de configuración de Apache Airflow en Amazon MWAA.

- Se añadió el paso de opciones de configuración de Airflow para el código de muestra de los complementos de Apache Airflow v2.0.2 en Creación de un complemento personalizado con Apache Hive y Hadoop.
- Se añadió el paso de opciones de configuración para el código de muestra de los complementos de Apache Airflow v2.0.2.
- Se añadió el paso de opciones de configuración de Airflow para el código de muestra de los complementos de Apache Airflow v2.0.2 en Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator.
- Se añadió el paso de opciones de configuración de Airflow para el código de muestra de los complementos de Apache Airflow v2.0.2 en Creación de un complemento personalizado con Oracle.

Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra para una conexión de Apache Airflow Snowflake en Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow Snowflake.	18 de junio de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una guía de cifrado del servidor a Creación de un bucket de Amazon S3 para Amazon MWAA. • Se añadió el backend de secretos de Apache Airflow v2.0.2 a Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager. • Se añadió una pregunta para que los procesos de trabajo de Apache Airflow aumenten las solicitudes de cuotas a Preguntas frecuentes sobre Amazon MWAA. • Se añadió la pregunta sobre qué métricas se usan para determinar si se deben escalar los procesos de trabajo de Apache Airflow a Preguntas frecuentes sobre Amazon MWAA. • Se añadió una pregunta para crear métricas personalizadas en CloudWatch a Preguntas frecuentes sobre Amazon MWAA.	2 de junio de 2021

	• Se añadieron pasos para habilitar las direcciones IP privadas para un punto de conexión de interfaz de VPC de Amazon S3 para una VPC con enrutamie nto privado a Creación de los puntos de conexión del servicio de VPC necesario s en una Amazon VPC con enrutamiento privado. • Se añadió una opción para configurar un túnel SSH utilizando el enrutamiento de puertos local en Tutorial: Configuración del acceso a la red privada mediante un host bastión de Linux.	
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra para un DAG que consulta la base de datos de metadatos de Amazon Aurora PostgreSQ L y publica métricas personalizadas en Amazon CloudWatch a Uso de un DAG para escribir métricas personalizadas en CloudWatch.	2 de junio de 2021

Nuevas guías	Se añadieron los siguientes cambios: • Se creó una guía sobre cómo usar las plantilla s de conexión de forma intercambiable en la interfaz de usuario de Apache Airflow a Información general sobre los tipos de conexión.	2 de junio de 2021
Correcciones	Se añadieron los siguientes cambios: • Se añadieron puntos de conexión de VPC de Apache Airflow a la plantilla de CloudFormation en la opción tres: crear una red de VPC sin acceso a Internet a Creación de la red de VPC.	2 de junio de 2021

Lanzamiento de Apache Airflow v	Lanzamiento de disponibilidad general de Apache Airflow v2.0.2. • Se creó Versiones de Apache Airflow en Amazon Managed Workflows para Apache Airflow. • Se creó Métricas del entorno de Apache Airflow en CloudWatch. • Se añadieron enlaces específicos de la versión para Apache Airflow v2.0.2 a Uso de las opciones de configuración de Apache Airflow en Amazon MWAA. • Se añadió una guía específica para la versión de Apache Airflow v2.0.2 a Instalación de dependencias de Python. • Se añadió una guía específica para la versión de Apache Airflow v2.0.2 a Administración de las dependencias de Python en requirements.txt. • Se añadieron complemen tos de muestra para Apache Airflow v2.0.2 a Instalación de complementos personali zados. • Se añadió un código de muestra de Apache Airflow	26 de mayo de 2021

v2.0.2 a Limpieza de bases de datos de Aurora PostgreSQL en un entorno de Amazon MWAA.

- Se añadió un código de muestra de Apache Airflow v2.0.2 a Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow.
- Se añadió un código de muestra de Apache Airflow v2.0.2 a Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator.
- Se añadieron los comandos Apache Airflow v2.0.2 a Referencia de los comandos de la CLI de Apache Airflow.
- Se añadieron scripts de Apache Airflow v2.0.2 a Creación de un token de la CLI de Apache Airflow.
- Se ha añadido una nota en la que se indica que Amazon MWAA utiliza la versión más reciente de Apache Airflow de forma predeterminada a Creación de entornos de Amazon MWAA.

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha añadido una guía para solucionar problemas de tareas de Airflow que están atascadas o que no se están ejecutando a Solución de problemas de Amazon Managed Workflows para Apache Airflow.	14 de mayo de 2021
Correcciones	Se añadieron los siguientes cambios: • Hemos actualizado el código del plugin de muestra para usarlo en la versión más reciente de Java a Creación de un complemento personalizado con Apache Hive y 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 mayo de 2021

Temas eliminados o movidos	Se añadieron los siguientes cambios: • Se han movido los temas Solución de problemas de Amazon Managed Workflows para Apache Airflow a nuevas páginas por categoría.	10 de mayo de 2021
Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una descripción general del bucket de Amazon S3 a Trabajo con DAG en Amazon MWAA.	10 de mayo de 2021
Temas eliminados o movidos	Se añadieron los siguientes cambios: • Se ha trasladado Acceso a Apache Airflow al nivel superior de navegación y se han añadido páginas para Cree un token de acceso al servidor web Apache Airflow, Creación de un token de la CLI de Apache Airflow y Referencia de los comandos de la CLI de Apache Airflow.	7 de mayo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadieron enlaces específicos de la versión a la guía de referencia de Apache Airflow para todos los comandos de la CLI de Airflow compatibles y no compatibles a Referencia de los comandos de la CLI de Apache Airflow. • Se añadieron enlaces específicos de la versión a la guía de referencia de Apache Airflow para todas las opciones de configuración en Uso de las opciones de configuración de Apache Airflow en Amazon MWAA. • Se añadió la utilidad CLI Amazon MWAA a Administración de las dependencias de Python en requirements.txt.	7 de mayo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se han añadido ejemplos planos y anidados sobre cómo estructurar un archivo plugins.zip en Instalación de complementos personalizados. • Se añadió la utilidad CLI Amazon MWAA a las páginas Cómo añadir o actualizar DAG, Instalación de complementos personalizados y Instalación de dependencias de Python. • Se reestructuró el contenido en una descripción general, cárguelo en Amazon S3 e instálelo en las secciones de Amazon MWAA en función de los comentarios de los usuarios en las páginas Instalación de complementos personalizados y Instalación de dependencias de Python. • Se añadió un ejemplo de caso de uso para crear y asociar los puntos de conexión de VPC necesarios a una Amazon VPC existente sin acceso a Internet en Acerca de las redes en Amazon MWAA.	30 de abril de 2021

Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra que usa una clave secreta en Secrets Manager para una variable de Apache Airflow en Uso de una clave secreta en AWS Secrets Manager para una variable de Apache Airflow.	30 de abril de 2021
Nuevas guías	Se añadieron los siguientes cambios: • Se creó Creación de los puntos de conexión del servicio de VPC necesarios en una Amazon VPC con enrutamiento privado.	30 de abril de 2021
Correcciones	Se añadieron los siguientes cambios: • Hemos actualizado `core.default_ui_timezone` para `webserver.default_ui_timezone` en Uso de las opciones de configuración de Apache Airflow en Amazon MWAA.	30 de abril de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadieron pasos de Windows (PuTTY) para el túnel SSH a Tutorial: Configuración del acceso a la red privada mediante un host bastión de Linux. • Se añadió un tema para `apache-airflow-providers-amazon`, que solo es compatible con Apache Airflow 2.0 para Solución de problemas de Amazon Managed Workflows para Apache Airflow.	23 de abril de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra que usa una clave secreta en Secrets Manager para una conexión de Apache Airflow en Uso de una clave secreta en AWS Secrets Manager para una conexión de Apache Airflow.	23 de abril de 2021

Nuevas guías	Se añadieron los siguientes cambios: • Se creó Acerca de las redes en Amazon MWAA. • Se creó Seguridad en la VPC en Amazon MWAA. • Se creó Administración del acceso a los puntos de conexión de Amazon VPC específicos del servicio en Amazon MWAA.	23 de abril de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha añadido una nueva plantilla de CloudFormation para crear una red de Amazon VPC sin acceso a Internet en Creación de la red de VPC. • Se ha añadido un nuevo tutorial para crear un AWS Client VPN en Tutorial: Configuración del acceso a la red privada mediante una AWS Client VPN. • Se cambió el nombre de la página de acceso a la red por el de modos de acceso a Apache Airflow en función de los comentarios de los usuarios y se simplificó la documentación en Modos de acceso de Apache Airflow. • Se simplificaron los documentos para incluir únicamente información de introducción a Amazon VPC y plantillas basadas en los comentarios de los usuarios en Creación de la red de VPC. • Se añadió una solución alternativa para el operador de BigQuery a Solución de problemas de Amazon	16 de abril de 2021

	Managed Workflows para Apache Airflow. • Se añadió la práctica recomendada de un archivo de restricciones de Apache Airflow v1.10.12 a Instalación de dependencias de Python.	
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra para crear un complemento personalizado con Oracle en Creación de un complemento personalizado con Oracle. • Se añadió un código de muestra para crear un complemento personalizado que genere variables de entorno de ejecución. •	16 de abril de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió un tema sobre el requisito de la regla de autorreferencia en un grupo de seguridad de VPC a Preguntas frecuentes sobre Amazon MWAA. • Se añadió un directorio de complementos personalizados y límites de tamaño a Instalación de complementos personalizados. • Se añadió el directorio de requisitos y los límites de tamaño a Instalación de dependencias de Python. • Se aclararon las opciones de configuración de Apache Airflow para `foo.user` y `foo.pass` en Administración de las dependencias de Python en requirements.txt. • Se añadió una descripción general de las opciones de configuración a Uso de las opciones de configuración de Apache Airflow en Amazon MWAA.	9 de abril de 2021

Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra para crear un complemento personalizado con `PythonVirtualenvOperator` en Creación de un complemento personalizado para Apache Airflow PythonVirtualEnvOperator. • Se añadió un código de muestra para crear un complemento personalizado con Apache Hive y Hadoop a Creación de un complemento personalizado con Apache Hive y Hadoop.	9 de abril de 2021
Correcciones	Se añadieron los siguientes cambios: • Hemos actualizado el formato de un archivo requirements.txt y hemos añadido un ejemplo que es compatible con la versión 1.10.12 de Apache Airflow en Instalación de dependencias de Python.	31 de marzo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una solución alternativa para eliminar un archivo requirements.txt o plugins.zip a Preguntas frecuentes sobre Amazon MWAA. • Se añadió una solución alternativa de bash para SSH en un entorno a Preguntas frecuentes sobre Amazon MWAA. • Se ha añadido un tema para el error ResourceAlreadyExistsException de CloudTrail a Solución de problemas de Amazon Managed Workflows para Apache Airflow.	26 de marzo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una lista de servicios de AWS utilizados a Rol de ejecución de Amazon MWAA. • Se añadió una lista de servicios de AWS utilizados a Roles vinculados a servicios para Amazon MWAA. • Se añadió una pregunta para la versión 3.7 de Python para Amazon MWAA a Preguntas frecuentes sobre Amazon MWAA. • Se agregó una pregunta `PythonVirtualenvOperator` para Preguntas frecuentes sobre Amazon MWAA. • Se añadió el script de solución de problemas como pasos siguientes para todos los temas relacionados con la configuración de la VPC y el entorno en Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se aclaró la documentación de que un bastión de Linux debe estar en la misma región que un entorno en	19 de marzo de 2021

	Tutorial: Configuración del acceso a la red privada mediante un host bastión de Linux.	
Nuevas guías	Se añadieron los siguientes cambios: • Se creó la guía de conexiones de Apache Airflow para AWS Secrets Manager a Configuración de una conexión de Apache Airflow mediante un secreto de AWS Secrets Manager. • Se creó un tutorial de inicio rápido con una plantilla de CloudFormation para crear la infraestructura de Amazon VPC, el bucket de Amazon S3 y el entorno de Amazon MWAA en Tutorial de inicio rápido de Amazon Managed Workflows para Apache Airflow.	19 de marzo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió el tema de solución de problemas Solución de problemas de Amazon Managed Workflows para Apache Airflow sobre la creación de un bucket de Amazon S3. • Se añadieron pasos para crear y adjuntar una política de JSON a Rol de ejecución de Amazon MWAA.	12 de marzo de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadió un código de muestra para añadir una configuración al activar un DAG a Acceso a Apache Airflow.	12 de marzo de 2021
Nuevas guías	Se añadieron los siguientes cambios: • Se creó una guía de mejores prácticas en Administración de las dependencias de Python en requirements.txt.	12 de marzo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió un tema de solución de problemas de Google/GCP/BigQuery a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se añadió un tema de solución de problemas de Cython a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se ha añadido un tema de solución de problemas de MySQL a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se añadió el tema de solución de problemas de errores del servidor web 5xx a Solución de problemas de Amazon Managed Workflows para Apache Airflow.	5 de marzo de 2021

Ahora es compatible	Se añadieron los siguientes cambios: • Anteriormente, `backend_kwargs` no era compatible con AWS Secrets Manager y se necesitaba una solución alternativa para anular la llamada a la función Secrets Manager. Ahora, `backend_kwargs` es compatible. Consulte el tema de solución de problemas de AWS Secrets Manager en Solución de problemas de Amazon Managed Workflows para Apache Airflow.	4 de marzo de 2021
Correcciones	Se añadieron los siguientes cambios: • Hemos actualizado el tamaño de cada clase de entorno para reflejar los GB reales en Configuración de la clase de entorno Amazon MWAA.	4 de marzo de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió acceso a la red privada mediante una política de puntos de conexión de VPC a Modos de acceso de Apache Airflow. • Se añadieron comprobaciones adicionales para el tema de solución de problemas sobre la creación de un entorno a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se agregaron pasos para ver los registros para `requirements.txt` a Instalación de dependencias de Python.	26 de febrero de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió el caso de us de Apache Hive a Instalación de dependencias de Python. • Se aclaró en los documento s que las dependencias requeridas para un paquete de Apache Airflow deben incluirse en el archivo `requirements.txt` en Instalación de dependencias de Python. • Se añadió el tema de solución de problemas sobre la actualización de requirements.txt a Solución de problemas de Amazon Managed Workflows para Apache Airflow.	25 de febrero de 2021
Nuevos tutoriales	Se añadieron los siguientes cambios: • Se añadió un tutorial de red privada a Tutorial: Configura ción del acceso a la red privada mediante un host bastión de Linux.	22 de febrero de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadieron configuraciones de redes públicas y privadas a Modos de acceso de Apache Airflow. • Se añadieron casos de uso y escenarios de usuario para grupos de desarrollo a Rol de ejecución de Amazon MWAA.	22 de febrero de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Se añadieron ejemplos de scripts de Python para el token de inicio de sesión web y el token CLI a Acceso a Apache Airflow. • Se añadió un código de muestra para activar el DAG en otro entorno a Códigos de ejemplo de Amazon Managed Workflows para Apache Airflow. • Se añadió un código de muestra para activar el DAG mediante una función de Lambda a Invocación de DAG con una función de Lambda.	22 de febrero de 2021

Nuevos comandos y procedimientos	Se añadieron los siguientes cambios: • Se añadieron procedimientos paso a paso a todos los scripts en Acceso a Apache Airflow.	22 de febrero de 2021
Nueva muestra de código	Se añadieron los siguientes cambios: • Muestra de curl actualizado para el token de inicio de sesión web en Acceso a Apache Airflow. • Se añadió un código de muestra para conectarse a un Amazon RDS Microsoft SQL Server en Uso de Amazon MWAA con Amazon RDS para Microsoft SQL Server.	17 de febrero de 2021

Nuevos comandos y procedimientos	Se añadieron los siguientes cambios: • Se añadieron comandos de la AWS CLI a las páginas Trabajo con DAG en Amazon MWAA. • Apache Airflow no admite los DAG serializados en los comandos de la CLI. Como la CLI se ejecuta en el servidor web, que no tiene complementos ni requisitos por motivos de seguridad, los entornos MWAA con plugins.zip o requirements.txt no admitirán estos comandos. Se han trasladado los comandos de Apache Airflow `list_dags` y `backfill` a comandos no compatibles en Acceso a Apache Airflow.	17 de febrero de 2021
Lanzamiento de GitHub	Los documentos de la guía del usuario son ahora de código abierto en GitHub. Seleccione Edit this page on GitHub (Editar esta página en GitHub) en cualquier página.	17 de febrero de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió una pregunta para el caso de uso de Step Functions frente a Amazon MWAA a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se añadió una política de acceso a CLI a Acceso a un entorno de Amazon MWAA. • Se aclaró en los documentos que cualquier opción de configuración de Apache Airflow compatible se puede especificar en Uso de las opciones de configuración de Apache Airflow en Amazon MWAA. • Se aclaró en los documentos que si un contenedor de Fargate en una zona de disponibilidad falla, MWAA cambia al otro contenedor en una zona de disponibilidad diferente en Creación de la red de VPC.	12 de febrero de 2021
Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se ha agregado Configuración de la clase de entorno Amazon MWAA.	5 de febrero de 2021

Temas eliminados o movidos	Se añadieron los siguientes cambios: • Se ha eliminado el requisito de que el nombre del bucket de Amazon S3 comience por `airflow-` a Introducción a Amazon Managed Workflows para Apache Airflow. • Se movió Acceso a un entorno de Amazon MWAA y Rol de ejecución de Amazon MWAA a Administración del acceso a un entorno de Amazon MWAA.	4 de febrero de 2021
Amazon MWA CloudFormation	Actualice los parámetros para crear un entorno en Amazon MWAA CloudFormation. • Elimine SubnetList. • Elimine TagList. • Añada NetworkConfiguration. • Añada TagMap. • Añada ejemplos de solicitudes de creación de entornos.	4 de febrero de 2021

Nuevos temas y casos de uso	Se añadieron los siguientes cambios: • Se añadió un ejemplo de configuración de correo electrónico a Uso de las opciones de configuración de Apache Airflow en Amazon MWAA. • Se añadió un tema de solución de problemas de PostgreShook a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se ha añadido un tema sobre solución de problemas AWS Secrets Manager a Solución de problemas de Amazon Managed Workflows para Apache Airflow. • Se añadió un caso de uso de alto rendimiento a Configuración del escalado automático de los procesos de trabajo de Amazon MWAA.	29 de enero de 2021

Lanzamiento de Amazon MWAA	Lanzamiento de disponibilidad general de Amazon Managed Workflows para Apache Airflow. • Documentación de la guía del usuario • CloudFormationDocumentación de	24 de noviembre de 2020