aws

사용자 가이드

Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows for Apache Airflow: 사용자 가이드

Table of Contents

Amazon Managed Workflows for Apache Airflow란 무엇입니까?

Amazon Managed Workflows for Apache Airflow(MWAA)는 Apache Airflow에 대한 관리형 오케스트레이션 서비스로 클라우드에서 데이터 파이프라인을 대규모로 설정하고 운영하는 데 사용할 수 있습니다. Apache Airflow는 워크플로우라고 하는 프로세스 및 작업 시퀀스를 프로그래밍 방식으로 작성, 예약 및 모니터링하는 데 사용되는 오픈 소스 도구입니다. Amazon MWAA를 사용하면 확장성, 가용성 및 보안을 위해 기본 인프라를 관리할 필요 없이 Apache Airflow와 Python을 사용하여 워크플로우를 생성할 수 있습니다. Amazon MWAA는 필요에 맞게 워크플로 실행 용량을 자동으로 확장합니다. Amazon MWAA는 AWS 보안 서비스와 통합되어 데이터에 빠르고 안전하게 액세스할 수 있도록 지원합니다.

내용

기능

- Automatic Airflow 설정 – Amazon MWAA 환경을 생성할 때 Apache Airflow 버전을 선택하여 Apache Airflow를 빠르게 설정할 수 있습니다. Amazon MWAA는 인터넷에서 다운로드할 수 있는 것과 동일한 Apache Airflow 사용자 인터페이스와 오픈 소스 코드를 사용하여 자동으로 Apache Airflow를 설정합니다.
- 자동 규모 조정 – 사용자 환경에서 실행되는 최소 및 최대 작업자 수를 설정하여 Apache Airflow 작업자를 규모를 자동으로 조정합니다. Amazon MWAA는 사용자 환경의 작업자를 모니터링하고 자동 규모 조정 구성 요소를 사용하여 사용자가 정의한 최대 작업자 수에 도달할 때까지 수요에 맞춰 작업자를 추가합니다.
- 내장 인증 — IAM () 에서 액세스 제어 정책을 정의하여 Apache Airflow 웹 서버에 대한 역할 기반 인증 및 권한 부여를 활성화합니다. AWS Identity and Access Management Apache Airflow 작업자는 서비스에 대한 보안 액세스를 위해 이러한 정책을 적용합니다. AWS

- 내장 보안 – Apache Airflow 작업자 및 스케줄러는 Amazon MWAA의 Amazon VPC에서 실행됩니다. 또한 를 사용하여 데이터가 자동으로 AWS Key Management Service 암호화되므로 사용자 환경은 기본적으로 안전합니다.
- 퍼블릭 또는 프라이빗 액세스 모드 – 프라이빗 또는 퍼블릭 액세스 모드를 사용하여 Apache Airflow 웹 서버에 액세스합니다. 퍼블릭 네트워크 액세스 모드는 인터넷을 통해 액세스할 수 있는 Apache Airflow 웹 서버의 VPC 엔드포인트를 사용합니다. 프라이빗 네트워크 액세스 모드는 사용자의 VPC에서 액세스할 수 있는 Apache Airflow 웹 서버의 VPC 엔드포인트를 사용합니다. 두 경우 모두 Apache Airflow 사용자의 액세스는 IAM AWS Identity and Access Management (IAM) 에서 정의한 액세스 제어 정책 및 SSO에 의해 제어됩니다. AWS
- 간소화된 업그레이드 및 패치 – Amazon MWAA는 정기적으로 새 버전의 Apache Airflow를 제공합니다. Amazon MWAA 팀이 해당 버전의 이미지를 업데이트하고 패치합니다.
- 워크플로 모니터링 — CloudWatch Amazon에서 Apache Airflow 로그 및 Apache Airflow 지표를 확인하여 추가 타사 도구 없이도 Apache Airflow 작업 지연이나 워크플로 오류를 식별할 수 있습니다. Amazon MWAA는 환경 지표를 자동으로 전송하며, 활성화된 경우 Apache Airflow 로그를 에 전송합니다. CloudWatch
- AWS 통합 — Amazon MWAA는 Amazon Athena, Amazon, Amazon DynamoDB, Amazon AWS DataSync EMR AWS Batch, Amazon CloudWatch EKS, Amazon Data Firehose,,, AWS Fargate Amazon AWS Lambda Redshift, Amazon SQS, Amazon SNS, AWS Glue Amazon S3와의 오픈 소스 통합은 물론 수백 개의 내장 및 커뮤니티 기능을 지원합니다. 오퍼레이터와 센서를 만들었습니다. SageMaker
- 작업자 플릿 — Amazon MWAA는 컨테이너를 사용하여 온디맨드로 작업자 플릿의 규모를 조정하고 AWS Fargate의 Amazon ECS를 사용하여 스케줄러 중단을 줄일 수 있도록 지원합니다. Amazon ECS 컨테이너에서 작업을 호출하는 연산자와 Kubernetes 클러스터에서 파드를 생성하고 실행하는 Kubernetes 연산자가 지원됩니다.

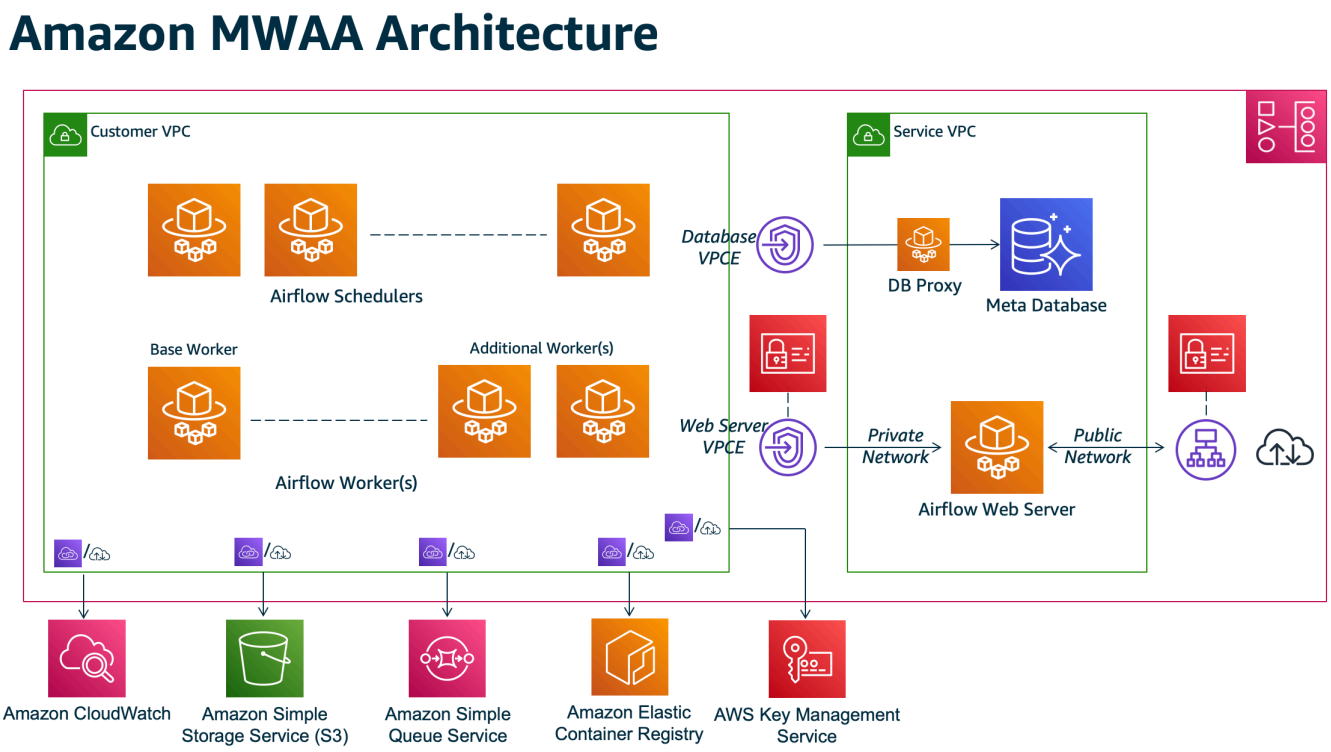
아키텍처

외부 상자(아래 이미지)에 포함된 모든 구성 요소는 계정에서 단일 Amazon MWAA 환경으로 나타납니다. Apache 에어플로우 스케줄러와 워커는 AWS Fargate (Fargate) 사용자 환경의 Amazon VPC에 있는 프라이빗 서브넷에 연결하는 컨테이너입니다. 각 환경에는 개인 보안이 설정된 VPC 엔드포인트를 통해 스케줄러 및 Workers Fargate 컨테이너에 액세스할 수 AWS 있는 자체 Apache Airflow 메타데이터베이스가 관리됩니다.

Amazon CloudWatch, Amazon S3, Amazon SQS, Amazon AWS KMS ECR은 Amazon MWAA와 별개이므로 아파치 에어플로우 스케줄러 및 Fargate 컨테이너의 작업자를 통해 액세스할 수 있어야 합니다.

퍼블릭 네트워크 Apache Airflow 액세스 모드를 선택하여 인터넷을 통해 Apache Airflow 웹 서버에 액세스하거나 사용자의 VPC 내에서 프라이빗 네트워크 Apache Airflow 액세스 모드를 선택하여 액세스할 수 있습니다. 두 경우 모두 Apache Airflow 사용자의 액세스는 사용자가 정의한 액세스 제어 정책 (IAM) 에 의해 제어됩니다. AWS Identity and Access Management

> **Note**
>
> 다중 Apache Airflow 스케줄러는 Apache Airflow v2 이상에서만 사용할 수 있습니다. Apache Airflow 참조 가이드의 개념에서 Apache Airflow 작업 수명 주기에 대해 자세히 알아봅니다.



통합

활발하게 성장하고 있는 Apache Airflow 오픈 소스 커뮤니티는 Apache Airflow를 서비스와 통합할 수 있는 운영자 (서비스 연결을 단순화하는 플러그인) 를 제공합니다. AWS 여기에는 Amazon S3, Amazon Redshift, Amazon EMR SageMaker, AWS Batch Amazon 등의 서비스와 다른 클라우드 플랫폼의 서비스가 포함됩니다.

Amazon MWAA와 함께 Apache Airflow를 사용하면 AWS 서비스 및 Apache 하둡, Presto, Hive, Spark와 같은 인기 있는 타사 도구와의 통합이 완벽하게 지원되어 데이터 처리 작업을 수행할 수 있습니다.

Amazon MWAA는 Amazon MWAA API와의 호환성을 유지하기 위해 최선을 다하고 있으며, Amazon MWAA는 AWS 서비스에 대한 안정적인 통합을 제공하고 커뮤니티에서 사용할 수 있도록 하며 커뮤니티 기능 개발에 참여하고자 합니다.

샘플 코드에 대한 내용은 Amazon Managed Workflows for Apache Airflow용 코드 예제 단원을 참조하십시오.

지원되는 버전

Amazon MWAA는 여러 버전의 Apache Airflow를 지원합니다. 지원하는 Apache Airflow 버전과 각 버전에 포함된 Apache Airflow 구성 요소에 대한 자세한 내용은 Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전 단원을 참조하십시오.

다음 단계

- 에어플로우 DAG 및 지원 파일을 위한 Amazon S3 버킷, 퍼블릭 라우팅을 지원하는 Amazon VPC, Amazon MWAA 환경을 생성하는 단일 AWS CloudFormation 템플릿으로 시작해 보십시오. Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼
- Airflow DAG와 지원 파일을 위한 Amazon S3 버킷을 만들고, 세 가지 Amazon VPC 네트워킹 옵션 중 하나를 선택하고, Amazon Managed Workflows for Apache Airflow 시작하기에서 Amazon MWAA 환경을 만들어 단계적으로 시작해 보십시오.

Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼

이 퀵 스타트 자습서에서는 Amazon VPC 인프라, dags 폴더가 있는 Amazon S3 버킷, Apache Airflow용 Amazon 관리형 워크플로를 동시에 생성하는 AWS CloudFormation 템플릿을 사용합니다.

주제

이 튜토리얼에서는

이 자습서에서는 Amazon S3에 DAG를 업로드하고, Apache Airflow에서 DAG를 실행하고, 로그인을 확인하는 세 가지 AWS Command Line Interface (AWS CLI) 명령을 안내합니다. CloudWatch 마지막으로 Apache Airflow 개발 팀을 위한 IAM 정책을 생성하는 단계를 안내합니다.

> **Note**
>
> 이 페이지의 AWS CloudFormation 템플릿은 에서 사용 가능한 최신 버전의 Apache Airflow를 위한 Amazon Apache Airflow용 관리형 워크플로 환경을 생성합니다. AWS CloudFormation사용 가능한 최신 버전은 아파치 에어플로우 v2.8.1입니다.

이 페이지의 AWS CloudFormation 템플릿은 다음을 생성합니다.

- VPC 인프라. 이 템플릿은 인터넷을 통한 퍼블릭 라우팅을(를) 사용합니다. `WebserverAccessMode: PUBLIC_ONLY`의 Apache Airflow 웹 서버에 퍼블릭 네트워크 액세스 모드을(를) 사용합니다.

- Amazon S3 버킷. 템플릿은 dags 폴더가 있는 Amazon S3 버킷을 생성합니다. Amazon MWAA용 Amazon S3 버킷 생성에 정의된 대로 버킷 버저닝을 활성화하여 모든 퍼블릭 액세스를 차단하도록 구성되어 있습니다.
- Amazon MWAA 환경. 템플릿은 Amazon S3 버킷의 dags 폴더와 연결된 Amazon MWAA 환경, Amazon MWAA에서 사용하는 AWS 서비스에 대한 권한을 가진 실행 역할, 그리고 에서 정의한 대로 AWS 소유 키를 사용한 암호화의 기본 환경을 생성합니다. Amazon MWAA 환경 생성
- CloudWatch 로그. 템플릿을 사용하면 에 정의된 CloudWatch 대로 Apache Airflow를 Airflow 스케줄러 로그 그룹, Airflow 웹 서버 로그 그룹, Airflow 작업자 로그 그룹, Airflow DAG 처리 로그 그룹 및 Airflow 작업 로그 그룹에 대해 “정보” 수준 이상으로 로그인할 수 있습니다. Amazon에서 에어플로우 로그 보기 CloudWatch

이 튜토리얼에서는 다음 작업을 수행할 수 있습니다.

- DAG를 업로드하고 실행. Amazon MWAA가 지원하는 최신 Apache Airflow 버전에 대한 Apache Airflow 튜토리얼 DAG를 Amazon S3에 업로드한 다음 DAG 추가 또는 업데이트에 정의된 대로 Apache Airflow UI에서 실행할 수 있습니다.
- 로그 확인. 에 정의된 대로 로그에서 Airflow 웹 서버 로그 그룹을 확인하십시오. CloudWatch Amazon에서 에어플로우 로그 보기 CloudWatch
- 액세스 제어 정책 생성. Amazon MWAA 환경 액세스에 정의된 대로 Apache Airflow 개발 팀을 위한 액세스 제어 정책을 IAM에서 생성합니다.

필수 조건

AWS Command Line Interface (AWS CLI) 는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI — 버전 2를 설치합니다.
- AWS CLI — 빠른 구성 `aws configure`.

1단계: AWS CloudFormation 템플릿을 로컬에 저장

- 다음 템플릿의 내용을 복사하고 로컬에 `mwaa-public-network.yml`로 저장합니다. 템플릿을 다운로드할 수도 있습니다.

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

```
Parameters:

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

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

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

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

  PrivateSubnet1CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.21.0/24
  MaxWorkerNodes:
    Description: The maximum number of workers that can run in the environment
    Type: Number
    Default: 2
  DagProcessingLogs:
    Description: Log level for DagProcessing
    Type: String
    Default: INFO
  SchedulerLogsLevel:
```

```
    Description: Log level for SchedulerLogs
    Type: String
    Default: INFO
  TaskLogsLevel:
    Description: Log level for TaskLogs
    Type: String
    Default: INFO
  WorkerLogsLevel:
    Description: Log level for WorkerLogs
    Type: String
    Default: INFO
  WebserverLogsLevel:
    Description: Log level for WebserverLogs
    Type: String
    Default: INFO

Resources:

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

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

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

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: MWAAEnvironment

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
```

```
      VpcId: !Ref VPC

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

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

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

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

```
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

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

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

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

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

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

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

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
```

```
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

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


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

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

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

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

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
```

```
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

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

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

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

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


##################################################################################################
  # CREATE MWAA

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

  MwaaEnvironment:
    Type: AWS::MWAA::Environment
    DependsOn: MwaaExecutionPolicy
    Properties:
```

```
      Name: !Sub "${AWS::StackName}-MwaaEnvironment"
      SourceBucketArn: !GetAtt EnvironmentBucket.Arn
      ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
      DagS3Path: dags
      NetworkConfiguration:
        SecurityGroupIds:
          - !GetAtt SecurityGroup.GroupId
        SubnetIds:
          - !Ref PrivateSubnet1
          - !Ref PrivateSubnet2
      WebserverAccessMode: PUBLIC_ONLY
      MaxWorkers: !Ref MaxWorkerNodes
      LoggingConfiguration:
        DagProcessingLogs:
          LogLevel: !Ref DagProcessingLogs
          Enabled: true
        SchedulerLogs:
          LogLevel: !Ref SchedulerLogsLevel
          Enabled: true
        TaskLogs:
          LogLevel: !Ref TaskLogsLevel
          Enabled: true
        WorkerLogs:
          LogLevel: !Ref WorkerLogsLevel
          Enabled: true
        WebserverLogs:
          LogLevel: !Ref WebserverLogsLevel
          Enabled: true
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId: !Ref VPC
      GroupDescription: !Sub "Security Group for Amazon MWAA Environment
${AWS::StackName}-MwaaEnvironment"
      GroupName: !Sub "airflow-security-group-${AWS::StackName}-MwaaEnvironment"

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

  SecurityGroupEgress:
```

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

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

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

          - Effect: Allow
            Action:
              - "s3:GetObject*"
```

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

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

```
              StringLike:
                "kms:ViaService":
                  - !Sub "sqs.${AWS::Region}.amazonaws.com"
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

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

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

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

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

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

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

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

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

2단계: 를 사용하여 스택 생성 AWS CLI

1. 명령 프롬프트에서 `mwaa-public-network.yml`이 저장된 디렉터리로 이동합니다. 예:

```
cd mwaaproject
```

2. AWS CLI을(를) 사용하여 스택을 생성하려면 `aws cloudformation create-stack` 명령을 사용합니다.

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

> **Note**
>
> Amazon VPC 인프라, Amazon S3 버킷, Amazon MWAA 환경을 생성하는 데 30분 이상이 소요됩니다.

3단계: Amazon S3에 DAG를 업로드하고 Apache Airflow UI에서 실행

1. 지원되는 최신 Apache Airflow 버전의 `tutorial.py` 파일 내용을 복사하고 로컬에 `tutorial.py`(으)로 저장합니다.
2. 명령 프롬프트에서 `tutorial.py`이 저장된 디렉터리로 이동합니다. 예:

```
cd mwaaproject
```

3. 다음 명령을 사용하여 Amazon S3 버킷을 모두 나열합니다.

```
aws s3 ls
```

4. 다음 명령을 사용하여 사용자 환경의 Amazon S3 버킷에 있는 파일과 폴더를 나열합니다.

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

5. 다음 스크립트를 사용하면 `tutorial.py` 파일을 `dags` 폴더에 업로드할 수 있습니다. *YOUR_S3_BUCKET_NAME*의 샘플 값을 대체합니다.

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

6. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
7. 환경을 선택합니다.
8. Airflow UI 열기를 선택합니다.
9. Apache Airflow UI의 사용 가능한 DAG 목록에서 튜토리얼 DAG를 선택합니다.
10. DAG 세부 정보 페이지에서 DAG 이름 옆에 있는 DAG 일시 중지/일시 중지 해제 토글을 선택하여 DAG 일시 중지를 해제합니다.
11. 트리거 DAG를 선택합니다.

4단계: 로그에서 CloudWatch 로그 보기

CloudWatch 콘솔에서 스택에 의해 활성화된 모든 Apache Airflow 로그의 Apache Airflow 로그를 볼 수 있습니다. AWS CloudFormation 다음 섹션에서는 Airflow 웹 서버 로그 그룹의 로그를 보는 방법을 확인할 수 있습니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 모니터링 창에서 Airflow 웹 서버 로그 그룹을 선택합니다.
4. 로그 스트림의 `webserver_console_ip` 로그를 선택합니다.

다음 단계

- DAG를 업로드하고, `requirements.txt`에서 Python 종속성을 지정하고, Amazon MWAA에서 DAG 작업의 `plugins.zip`에서 사용자 지정 플러그인을 지정하는 방법에 대해 자세히 알아봅니다.
- Amazon MWAA의 Apache Airflow 성능 튜닝에서 환경 성능 조정을 위해 권장하는 모범 사례에 대해 자세히 알아봅니다.
- Amazon MWAA의 대시보드 및 알람 모니터링에서 사용자 환경을 위한 모니터링 대시보드를 생성합니다.
- Amazon Managed Workflows for Apache Airflow용 코드 예제에서 일부 DAG 코드 샘플을 실행합니다.

Amazon Managed Workflows for Apache Airflow 시작하기

Amazon Managed Workflows for Apache Airflow는 Amazon S3 스토리지 버킷의 Amazon VPC, DAG 코드 및 지원 파일을 사용하여 환경을 생성합니다. 이 가이드에서는 Amazon MWAA를 시작하는 데 필요한 사전 요구 사항과 필수 AWS 리소스를 설명합니다.

주제

필수 조건

Amazon MWAA 환경을 생성하려면 생성해야 할 AWS 리소스에 대한 권한이 있는지 확인하는 추가 단계를 수행해야 합니다.

- AWS계정 — Amazon MWAA와 사용자 환경에서 사용하는 AWS 서비스 및 리소스를 사용할 권한이 있는 AWS 계정입니다.

이 가이드 정보

이 섹션에서는 이 가이드에서 생성할 AWS 인프라 및 리소스에 대해 설명합니다.

- Amazon VPC — Amazon MWAA 환경에 필요한 Amazon VPC 네트워킹 구성 요소입니다. Amazon MWAA에서의 네트워킹에 대해에 표시된 대로 이러한 요구 사항(고급)을 충족하는 기존 VPC를 구성하거나 the section called “VPC 네트워크 생성”에서 정의한 대로 VPC 및 네트워킹 구성 요소를 생성할 수 있습니다.

- Amazon S3 버킷 — DAG와 관련 파일(예: `plugins.zip` 및 `requirements.txt`)을 저장하는 Amazon S3 버킷입니다. Amazon MWAA용 Amazon S3 버킷 생성에 정의된 대로 버킷 버전 관리를 활성화하고 모든 퍼블릭 액세스를 차단하도록 Amazon S3 버킷을 구성해야 합니다.
- Amazon MWAA 환경 — Amazon MWAA 환경 생성에 정의된 대로 Amazon S3 버킷의 위치, DAG 코드 경로, 사용자 지정 플러그인 또는 Python 종속성, Amazon VPC 및 해당 보안 그룹으로 구성된 Amazon MWAA 환경입니다.

시작하기 전에

Amazon MWAA 환경을 생성하려면 환경을 생성하기 전에 추가 단계를 수행하여 다른 AWS 리소스를 생성하고 구성하는 것이 좋습니다.

환경을 생성하려면 다음이 필요합니다.

- AWS KMS키 - 사용자 환경의 데이터 암호화를 위한 AWS KMS 키입니다. Amazon MWAA 콘솔에서 기본 옵션을 선택하여 환경을 생성할 때 AWS소유 키를 생성하거나 구성된 환경(고급)에서 사용하는 다른 AWS 서비스에 대한 권한을 가진 기존 고객 관리형 키를 지정할 수 있습니다. 자세한 내용은 암호화를 위한 고객 관리형 키 사용 단원을 참조하십시오.
- 실행 역할 — Amazon MWAA가 사용자 환경의 AWS 리소스에 액세스할 수 있도록 하는 실행 역할입니다. Amazon MWAA 콘솔에서 기본 옵션을 선택하여 환경을 생성할 때 실행 역할을 생성할 수 있습니다. 자세한 내용은 Amazon MWAA 실행 역할 단원을 참조하십시오.
- VPC 보안 그룹 — Amazon MWAA가 VPC 네트워크의 다른 AWS 리소스에 액세스할 수 있도록 허용하는 VPC 보안 그룹입니다. Amazon MWAA 콘솔에서 기본 옵션을 선택하여 환경을 생성할 때 보안 그룹을 생성하거나 보안 그룹에 적절한 인바운드 및 아웃바운드 규칙(고급)을 제공할 수 있습니다. 자세한 내용은 Amazon MWAA에서 VPC 보안 단원을 참조하십시오.

사용 가능한 리전

이제 다음 AWS 리전에서 Amazon MWAA를 사용할 수 있습니다.

- 유럽(스톡홀름) - eu-north-1
- 유럽(프랑크푸르트) eu-central-1
- 유럽(아일랜드) - eu-west-1
- 유럽(런던) - eu-west-2
- 유럽(파리) - eu-west-3

- 아시아 태평양(뭄바이) - ap-south-1
- 아시아 태평양(싱가포르) - ap-southeast-1
- 아시아 태평양(시드니) - ap-southeast-2
- 아시아 태평양(도쿄) - ap-northeast-1
- 아시아 태평양(서울) - ap-northeast-2
- 미국 동부(버지니아 북부) - us-east-1
- 미국 동부(오하이오) - us-east-2
- 미국 서부(오레곤) - us-west-2
- 캐나다(중부) - ca-central-1
- 남아메리카(상파울루) - sa-east-1

Amazon MWAA용 Amazon S3 버킷 생성

이 가이드에서는 Apache Airflow DAG(Directed Acyclic Graph), 사용자 지정 플러그인을 `plugins.zip` 파일에, Python 종속성을 `requirements.txt` 파일에 저장하기 위해 Amazon S3 버킷을 생성하는 단계를 설명합니다.

목차

시작하기 전에

- 버킷을 생성한 후에는 Amazon S3 버킷 이름을 변경할 수 없습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 이름 지정 규칙을 참조하십시오.
- Amazon MWAA 환경에 사용되는 Amazon S3 버킷은 버킷 버전 관리를 활성화한 상태에서 모든 퍼블릭 액세스를 차단하도록 구성해야 합니다.
- Amazon MWAA 환경에 사용되는 Amazon S3 버킷은 Amazon MWAA 환경과 동일한 AWS 리전에 위치해야 합니다. Amazon MWAA에 대한 AWS 리전 목록을 보려면 AWS 일반 참조에서 Amazon MWAA 엔드포인트 및 할당량을 참조하십시오.

버킷 생성

이 섹션에서는 환경에 맞는 Amazon S3 버킷을 생성하는 단계를 설명합니다.

버킷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 생성을 선택합니다.
3. 버킷 이름에 버킷의 DNS 호환 이름을 입력합니다.

   버킷 이름은 다음과 같아야 합니다.

   - 모든 Amazon S3에서 고유해야 합니다.
   - 3~63자 이내여야 합니다.
   - 대문자가 없어야 합니다.
   - 소문자 또는 숫자로 시작해야 합니다.

   > **Important**
   > 버킷 이름에 계정 번호와 같은 중요한 정보를 포함하지 마십시오. 버킷 이름은 버킷의 객체를 가리키는 URL에 표시됩니다.

4. 리전에서 AWS 리전을 선택합니다. 이 AWS 리전은 Amazon MWAA 환경과 동일해야 합니다.

   - 지연 시간과 비용을 최소화하고 규제 요건을 해결하기 위해 가까운 리전을 선택하는 것이 좋습니다.
5. 모든 퍼블릭 액세스 차단을 선택합니다.
6. 버킷 버전 관리에서 활성화를 선택합니다.
7. 선택 사항 - 태그. 키-값 태그 쌍을 추가하여 태그에서 Amazon S3 버킷을 식별할 수 있습니다. 예: `Bucket : Staging`.
8. 선택 사항 - 서버 측 암호화. Amazon S3 버킷에서 다음 암호화 옵션 중 하나를 선택적으로 활성화할 수 있습니다.

   a. 서버 측 암호화에서 Amazon S3 키(SSE-S3)를 선택하여 버킷에 대해 서버 측 암호화를 활성화합니다.

b. Amazon S3 버킷 암호화에 사용할 AWS KMS 키를 사용하려면 AWS Key Management Service 키(SSE-KMS)를 선택합니다.

i. AWS관리형 키(aws/s3) - 이 옵션을 선택하면 Amazon MWAA에서 관리하는 AWS 소유 키를 사용하거나 Amazon MWAA 환경의 암호화를 위한 고객 관리형 키를 지정할 수 있습니다.

ii. AWS KMS 키에서 선택 또는 AWS KMS 키 ARN 입력 - 이 단계에서 고객 관리형 키를 지정하려면 AWS KMS 키 ID 또는 ARN을 지정해야 합니다. AWS KMS 별칭 및 다중 리전 키는 Amazon MWAA에서 지원되지 않습니다. 지정한 AWS KMS 키는 Amazon MWAA 환경의 암호화에도 사용해야 합니다.

9. 선택 사항 - 고급 설정. Amazon S3 객체 잠금을 활성화하려면 다음을 수행합니다.

a. 고급 설정을 선택하고 활성화합니다.

> **Important**
>
> 객체 잠금을 활성화하면 이 버킷의 객체를 영구적으로 잠글 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 객체 잠금을 사용하여 객체 잠금을 참조하십시오.

b. 승인을 선택합니다.

10. 버킷 생성을 선택합니다.

다음 단계

- VPC 네트워크 생성에서 환경에 필요한 Amazon VPC 네트워크를 생성하는 방법을 알아봅니다.
- ACL 버킷 권한을 설정하려면 어떻게 해야 합니까? 에서 액세스 권한을 관리하는 방법을 알아봅니다.
- S3 버킷을 삭제하려면 어떻게 해야 합니까?에서 스토리지 버킷을 삭제하는 방법을 알아봅니다.

VPC 네트워크 생성

Amazon Managed Workflows for Apache Airflow에는 환경을 지원하기 위해 Amazon VPC 및 특정 네트워킹 구성 요소가 필요합니다. 이 가이드에서는 Amazon Managed Workflows for Apache Airflow 환경에 대한 Amazon VPC 네트워크를 생성하는 다양한 옵션에 대해 설명합니다.

> **Note**
>
> Apache Airflow는 지연 시간이 짧은 네트워크 환경에서 가장 잘 작동합니다. 트래픽을 다른 지역이나 온프레미스 환경으로 라우팅하는 기존 Amazon VPC를 사용하는 경우 Amazon SQS, CloudWatch, Amazon S3, AWS KMS, 및 Amazon ECR에 AWS PrivateLink 엔드포인트를 추가하는 것이 좋습니다. Amazon MWAA에 대해 AWS PrivateLink를 구성하는 방법에 대한 자세한 내용은 인터넷 액세스 없이 Amazon VPC 네트워크 생성을 참조하십시오.

목차

필수 조건

AWS Command Line Interface(AWS CLI)는 명령줄 쉘의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI - 버전 2 설치
- AWS CLI - `aws configure`을 통한 빠른 구성.

시작하기 전에

- 환경에 지정된 VPC 네트워크는 환경 생성 후에는 변경할 수 없습니다.
- Amazon VPC 및 Apache Airflow 웹 서버에 프라이빗 또는 퍼블릭 라우팅을 사용할 수 있습니다. 옵션 목록을 보려면 the section called “Amazon VPC 및 Apache Airflow 액세스 모드의 사용 사례 예시” 단원을 참조하십시오.

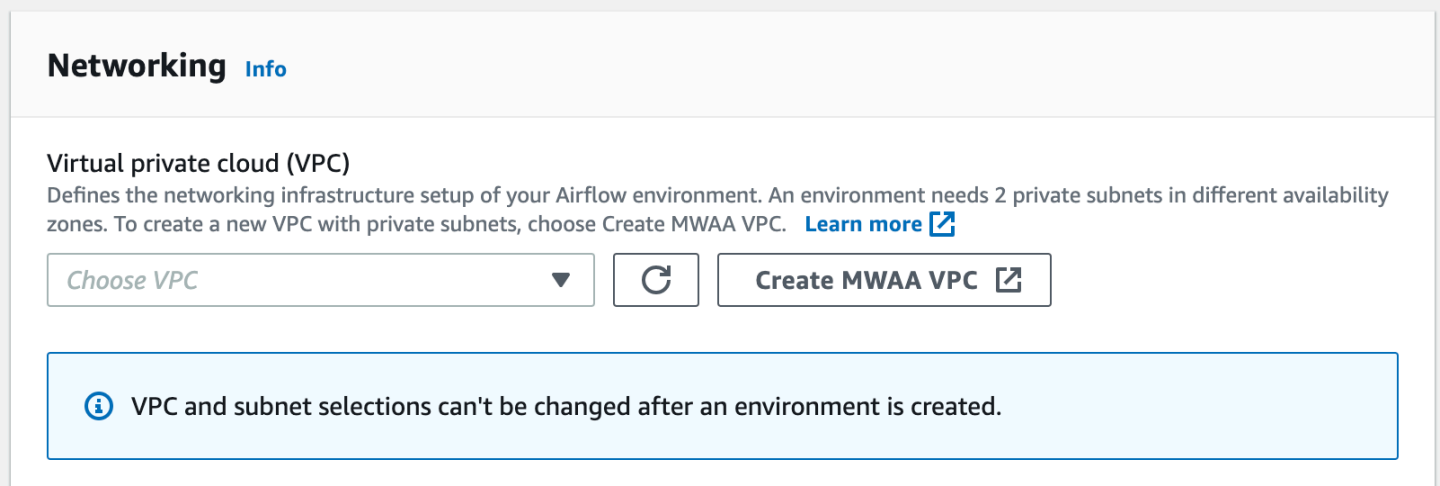
Amazon VPC 네트워크 생성 옵션

다음 섹션에서는 환경에 대한 Amazon VPC 네트워크를 생성하는 데 사용할 수 있는 옵션을 설명합니다.

옵션 1: Amazon MWAA 콘솔에서 VPC 네트워크 생성

다음 섹션에서는 Amazon MWAA 콘솔에서 Amazon VPC 네트워크를 생성하는 방법을 보여줍니다. 이 옵션은 인터넷을 통한 퍼블릭 라우팅를 사용합니다. 프라이빗 네트워크 또는 퍼블릭 네트워크 액세스 모드의 Apache Airflow 웹 서버에 사용할 수 있습니다.

다음 이미지는 Amazon MWAA 콘솔에서 MWAA VPC 생성 버튼을 찾을 수 있는 위치를 보여줍니다.



옵션 2: 인터넷에 액세스할 수 있는 Amazon VPC 네트워크 생성

다음 AWS CloudFormation 템플릿은 기본 AWS 리전에 인터넷 액세스가 가능한 Amazon VPC 네트워크를 생성합니다. 이 옵션은 인터넷을 통한 퍼블릭 라우팅를 사용합니다. 이 템플릿은 프라이빗 네트워크 또는 퍼블릭 네트워크 액세스 모드의 Apache Airflow 웹 서버에 사용할 수 있습니다.

1. 다음 템플릿의 내용을 복사하고 로컬에 `cfn-vpc-public-private.yaml`로 저장합니다. 템플릿을 다운로드할 수도 있습니다.

```
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. 명령 프롬프트에서 `cfn-vpc-public-private.yaml`이 저장된 디렉터리로 이동합니다. 예:

```
cd mwaaproject
```

3. AWS CLI를 사용하여 스택을 생성하려면 `aws cloudformation create-stack` 명령을 사용합니다.

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

> **Note**
>
> Amazon VPC 인프라를 생성하는 데 약 30분 정도 걸립니다.

옵션 3: 인터넷에 액세스할 수 없는 Amazon VPC 네트워크 생성

다음 AWS CloudFormation 템플릿은 기본 AWS 리전에서 인터넷 액세스 없이 Amazon VPC 네트워크를 생성합니다.

Important

인터넷 액세스 없이 Amazon VPC를 사용하는 경우 Amazon ECR에 게이트웨이 엔드포인트를 사용하여 Amazon S3에 액세스할 수 있는 권한을 부여해야 합니다. 다음을 수행하여 게이트웨이 엔드포인트를 생성할 수 있습니다.

1. 다음 JSON IAM 정책을 복사하여 로컬에 `s3-gw-endpoint-policy.json`로 저장합니다. 이 정책은 Amazon ECR이 Amazon S3 리소스에 액세스하는 데 필요한 최소 권한을 부여합니다.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

2. 다음 AWS CLI 명령을 사용하여 엔드포인트를 생성합니다. `--vpc-id` 및 `--route-table-ids`에 대한 값을 Amazon VPC의 정보로 바꿉니다. `--service-name`를 리전에 맞는 이름으로 바꿉니다.

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

Amazon ECR에 대한 Amazon S3 게이트웨이 엔드포인트 생성에 대한 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 Amazon S3 게이트웨이 엔드포인트 생성을 참조하십시오.

이 옵션은 인터넷 접속이 필요 없는 프라이빗 라우팅를 사용합니다. 이 템플릿은 프라이빗 네트워크 액세스 모드가 있는 Apache Airflow 웹 서버에만 사용할 수 있습니다. 이는 환경에서 사용하는 AWS 서비스에 필요한 VPC 엔드포인트를 생성합니다.

1. 다음 템플릿의 내용을 복사하고 로컬에 `cfn-vpc-private.yaml`로 저장합니다. 템플릿을 다운로드할 수도 있습니다.

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

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

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

   PrivateSubnet2CIDR:
     Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
     Type: String
     Default: 10.192.11.0/24

Resources:
   VPC:
     Type: AWS::EC2::VPC
     Properties:
       CidrBlock: !Ref VpcCIDR
       EnableDnsSupport: true
       EnableDnsHostnames: true
       Tags:
```

```
      - Key: Name
        Value: !Ref AWS::StackName

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
        - !Ref SecurityGroup

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

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

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

Outputs:
  VPC:
    Description: A reference to the created VPC
```

```
    Value: !Ref VPC

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

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

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

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

2. 명령 프롬프트에서 `cfn-vpc-private.yml`이 저장된 디렉터리로 이동합니다. 예:

```
cd mwaaproject
```

3. AWS CLI를 사용하여 스택을 생성하려면 `aws cloudformation create-stack` 명령을 사용합니다.

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

> **Note**
>
> Amazon VPC 인프라를 생성하는 데 약 30분 정도 걸립니다.

4. 컴퓨터에서 이러한 VPC 엔드포인트에 액세스하기 위한 메커니즘을 생성해야 합니다. 자세한 내용은 아마존 MWAA의 서비스별 아마존 VPC 엔드포인트에 대한 액세스 관리 단원을 참조하십시오.

> **Note**
>
> Amazon MWAA 보안 그룹의 CIDR에서 아웃바운드 액세스를 추가로 제한할 수 있습니다. 예를 들어 자체 참조 아웃바운드 규칙, Amazon S3의 접두사 목록, Amazon VPC의 CIDR을 추가하여 자체적으로 제한할 수 있습니다.

다음 단계

- Amazon MWAA 환경 생성에서 Amazon MWAA 환경을 만드는 방법을 알아봅니다.
- 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성에서 프라이빗 라우팅을 사용하여 컴퓨터에서 Amazon VPC로 VPN 터널을 생성하는 방법을 알아봅니다.

Amazon MWAA 환경 생성

Apache Airflow용 Amazon Managed Workflows는 Apache에서 제공하는 것과 동일한 오픈 소스 Apache Airflow와 사용자 인터페이스를 사용하여 선택한 버전의 환경에 Apache Airflow를 설정합니다. 이 가이드에서는 Amazon MWAA 환경을 생성하는 단계를 설명합니다.

목차

시작하기 전 준비 사항

- 환경에 지정한 VPC 네트워크는 환경을 만든 후에는 수정할 수 없습니다.
- 버킷 버전 관리가 활성화된 상태에서 모든 퍼블릭 액세스가 차단되도록 구성된 Amazon S3 버킷이 필요합니다.
- Amazon MWAA를 사용할 수 있는 권한과 IAM 역할을 생성하려면 권한 AWS Identity and Access Management (IAM) 이 있는 AWS 계정이 필요합니다. Amazon VPC 내에서 Apache Airflow 액세스

를 제한하는 Apache Airflow 웹 서버의 사설 네트워크 액세스 모드를 선택하는 경우 Amazon VPC 엔드포인트를 생성할 수 있는 IAM의 권한이 필요합니다.

Apache Airflow 버전

다음 Apache Airflow 버전은 Amazon Managed Workflows for Apache Airflow에서 지원됩니다.

> **Note**
>
> - Apache Airflow v2.2.2부터 Amazon MWAA는 Python 요구 사항, 공급자 패키지 및 사용자 지정 플러그인을 Apache Airflow 웹 서버에 직접 설치할 수 있도록 지원합니다.
> - Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.
>
> 요구 사항 파일에서 제약 조건을 설정하는 방법에 대한 자세한 내용은 Python 종속성 설치를 참조하십시오.

Apache Airflow 버전	Apache Airflow 가이드	Apache Airflow 제약 조건	Python 버전
v2.8.1	아파치 에어플로우 v2.8.1 참조 가이드	아파치 에어플로우 v2.8.1 제약 파일	Python 3.11
v2.7.2	Apache Airflow v2.7.2 참조 가이드	Apache Airflow v2.7.2 제약 조건 파일	Python 3.11
v2.6.3	Apache Airflow v2.6.3 참조 가이드	Apache Airflow v2.6.3 제약 조건 파일	Python 3.10
v2.5.1	Apache Airflow v2.5.1 참조 가이드	Apache Airflow v2.5.1 제약 조건 파일	Python 3.10
v2.4.3	Apache Airflow v2.4.3 참조 가이드	Apache Airflow v2.4.3 제약 조건 파일	Python 3.10

Apache Airflow 버전	Apache Airflow 가이드	Apache Airflow 제약 조건	Python 버전
v2.2.2	Apache Airflow v2.2.2 참조 가이드	Apache Airflow v2.2.2 제약 조건 파일	Python 3.7
v2.0.2	Apache Airflow v2.0.2 참조 가이드	Apache Airflow v2.0.2 제약 조건 파일	Python 3.7

메타데이터 데이터베이스 백업 지침을 포함하여 자체 관리형 Apache Airflow 배포 마이그레이션 또는 기존 Amazon MWAA 환경 마이그레이션에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드를 참조하십시오.

환경 생성

다음 섹션에서는 Amazon MWAA 환경을 생성하는 단계를 설명합니다.

1단계: 세부 정보 지정

환경에 대한 세부 정보를 지정하려면

1. Amazon MWAA 콘솔을 엽니다.
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.
3. 환경 생성을 선택합니다.
4. 세부 정보 지정 페이지의 환경 세부 정보에서:
   a. 이름에 환경의 고유 이름을 입력합니다.
   b. Airflow 버전에서 Apache Airflow 버전을 선택합니다.

> **Note**
> 값을 지정하지 않으면 기본적으로 최신 Airflow 버전이 지정됩니다. 사용 가능한 최신 버전은 아파치 에어플로우 v2.8.1입니다.

5. Amazon S3의 DAG 코드에서 다음을 지정합니다.
   a. S3 버킷. S3 찾아보기를 선택하고 Amazon S3 버킷을 선택하거나 Amazon S3 URI를 입력합니다.

b. DAG 폴더. S3 찾아보기를 선택하고 Amazon S3 버킷에서 `dags` 폴더를 선택하거나 Amazon S3 URI를 입력합니다.

c. 플러그인 파일 - 선택 사항. S3 찾아보기를 선택하고 Amazon S3 버킷에서 `plugins.zip` 파일을 선택하거나 Amazon S3 URI를 입력합니다.

d. 요구 사항 파일 - 선택 사항. S3 찾아보기를 선택하고 Amazon S3 버킷에서 `requirements.txt` 파일을 선택하거나 Amazon S3 URI를 입력합니다.

e. 시작 스크립트 파일 - 선택 사항, 찾아보기S3를 선택하고 Amazon S3 버킷에서 스크립트 파일을 선택하거나 Amazon S3 URI를 입력합니다.

6. 다음을 선택합니다.

2단계: 고급 설정 구성

고급 설정을 구성하려면

1. 고급 설정 구성 페이지의 네트워킹에서:

   - Amazon VPC를 선택합니다.

     이 단계에서는 Amazon VPC에 두 개의 프라이빗 서브넷을 채웁니다.

2. 웹 서버 액세스에서 원하는 Apache Airflow 액세스 모드를 선택합니다.

   a. 프라이빗 네트워크. 이렇게 하면 사용자 환경의 IAM 정책에 대한 액세스 권한이 부여된 Amazon VPC 내 사용자만 Apache Airflow UI에 액세스하도록 제한됩니다. 이 단계를 수행하려면 Amazon VPC 엔드포인트를 생성할 권한이 필요합니다.

   > **Note**
   >
   > Apache Airflow UI가 기업 네트워크 내에서만 액세스되고 웹 서버 요구 사항 설치를 위해 퍼블릭 리포지토리에 대한 액세스가 필요하지 않은 경우 프라이빗 네트워크 옵션을 선택합니다. 이 액세스 모드 옵션을 선택하는 경우 Amazon VPC의 Apache Airflow 웹 서버에 액세스하기 위한 메커니즘을 생성해야 합니다. 자세한 내용은 Apache Airflow 웹 서버의 VPC 엔드포인트 액세스(프라이빗 네트워크 액세스) 단원을 참조하십시오.

   b. 퍼블릭 네트워크. 이렇게 하면 사용자 환경의 IAM 정책에 대한 액세스 권한이 부여된 사용자가 인터넷을 통해 Apache Airflow UI에 액세스할 수 있습니다.

3. 보안 그룹에서 Amazon VPC를 보호하는 데 사용할 보안 그룹을 선택합니다.

a. 기본적으로 Amazon MWAA는 새 보안 그룹 생성의 특정 인바운드 및 아웃바운드 규칙을 사용하여 Amazon VPC에 보안 그룹을 생성합니다.

b. 선택 사항. 새 보안 그룹 생성의 확인란을 선택 취소하여 최대 5개의 보안 그룹을 선택합니다.

> **Note**
>
> 네트워크 트래픽을 허용하려면 기존 Amazon VPC 보안 그룹을 특정 인바운드 및 아웃바운드 규칙으로 구성해야 합니다. 자세한 내용은 Amazon MWAA에서 VPC 보안 단원을 참조하십시오.

4. 환경 클래스에서 환경 클래스를 선택합니다.

   워크로드를 지원하는 데 필요한 가장 작은 크기를 선택하는 것이 좋습니다. 환경 클래스는 언제든지 변경할 수 있습니다.

5. 최대 작업자 수에는 환경에서 실행할 최대 Apache Airflow 작업자 수를 지정합니다.

   자세한 정보는 고성능 사용 사례 예시을 참조하세요.

6. 최대 웹 서버 수 및 최소 웹 서버 수를 지정하여 Amazon MWAA가 사용자 환경에서 Apache Airflow 웹 서버를 확장하는 방법을 구성합니다.

   웹 서버 자동 확장에 대한 자세한 내용은 을 참조하십시오. the section called “웹 서버 자동 스케일링 구성”

7. 암호화에서 데이터 암호화 옵션을 선택합니다.

   a. 기본적으로 Amazon MWAA는 AWS 소유 키를 사용하여 데이터를 암호화합니다.

   b. 선택 사항. 암호화 설정 사용자 지정 (고급) 을 선택하여 다른 키를 선택합니다. AWS KMS 이 단계에서 고객 관리 키를 지정하려면 키 ID 또는 ARN을 AWS KMS 지정해야 합니다. AWS KMS Amazon MWAA에서는 별칭 및 다중 지역 키를 지원하지 않습니다. Amazon S3 버킷에서 서버 측 암호화를 위해 Amazon S3 키를 지정한 경우 Amazon MWAA 환경에도 동일한 키를 지정해야 합니다.

   > **Note**
   >
   > Amazon MWAA 콘솔에서 키를 선택하려면 해당 키에 대한 권한이 있어야 합니다. 또한 키 정책 연결에 설명된 정책을 첨부하여 Amazon MWAA에 키를 사용할 수 있는 권한을 부여해야 합니다.

8. 권장. 모니터링에서 Apache Airflow 로그를 로그로 전송할 Airflow 로깅 구성의 로그 범주를 하나 이상 선택합니다. CloudWatch

    a. Airflow 작업 로그. 로그 인 로그 레벨로 전송할 Apache Airflow 작업 로그의 유형을 선택합니다. CloudWatch

    b. Airflow 웹 서버 로그. 로그인 로그 수준으로 보낼 Apache Airflow 웹 서버 로그의 유형을 선택합니다. CloudWatch

    c. Airflow 스케줄러 로그. 로그인 로그 수준으로 보낼 Apache Airflow 스케줄러 로그의 유형을 선택합니다. CloudWatch

    d. Airflow 작업자 로그. 로그 인 로그 수준으로 보낼 Apache Airflow 작업자 로그의 유형을 선택합니다. CloudWatch

    e. Airflow DAG 처리 로그. 로그 인 로그 수준으로 전송할 Apache Airflow DAG 처리 로그의 유형을 선택합니다. CloudWatch

9. 선택 사항. Airflow 구성 옵션에서는 사용자 지정 구성 옵션 추가를 선택합니다.

    Apache Airflow 버전에 대해서는 Apache Airflow 구성 옵션의 제안 드롭다운 목록에서 선택하거나 사용자 지정 구성 옵션을 지정할 수 있습니다. 예: `core.default_task_retries : 3`.

10. 선택 사항. 태그에서 새 태그 추가를 선택하여 태그를 환경에 연결합니다. 예: `Environment: Staging`.

11. 사용 권한에서 실행 역할을 선택합니다.

    a. 기본적으로 Amazon MWAA는 새 역할 생성에서 실행 역할을 생성합니다. I이 옵션을 사용하려면 IAM 역할을 생성할 권한이 있어야 합니다.

    b. 선택 사항. 역할 ARN 입력을 선택하여 기존 실행 역할의 Amazon 리소스 이름(ARN)을 입력합니다.

12. 다음을 선택합니다.

3단계: 검토 및 생성

환경 요약을 검토하려면

- 환경 요약을 검토하고 환경 생성을 선택합니다.

> **Note**
>
> 환경을 생성하는 데 약 20~30분이 소요됩니다.

다음 단계

- Amazon MWAA용 Amazon S3 버킷 생성에서 Amazon S3 버킷을 생성하는 방법을 알아봅니다.

Amazon MWAA 환경에 대한 액세스 관리

Apache Airflow용 Amazon 관리형 워크플로는 환경에서 사용되는 다른 AWS 서비스 및 리소스를 사용할 수 있도록 허용되어야 합니다. 또한 Amazon MWAA 환경 및 IAM (아파치 에어플로우 UI) 에 AWS Identity and Access Management 액세스할 수 있는 권한을 부여받아야 합니다. 이 섹션에서는 사용자 환경의 AWS 리소스에 대한 액세스 권한을 부여하는 데 사용되는 실행 역할, 권한을 추가하는 방법, Amazon MWAA 환경 및 Apache Airflow UI에 액세스하는 데 필요한 AWS 계정 권한을 설명합니다.

주제

Amazon MWAA 환경 액세스

Amazon Managed Workflows for Apache Airflow를 사용하려면 필요한 권한이 있는 계정과 IAM 엔티티를 사용해야 합니다. 이 페이지에서는 Amazon Managed Workflows for Apache Airflow 환경을 위해 Apache Airflow 개발 팀과 Apache Airflow 사용자에게 적용할 수 있는 액세스 정책에 대해 설명합니다.

Amazon MWAA 리소스에 액세스하려면 임시 보안 인증 정보를 사용하고 그룹 및 역할을 사용하여 페더레이션 ID를 구성하는 것이 좋습니다. 정책을 IAM 사용자에게 직접 연결하지 말고 대신 그룹이나 역할을 정의하여 리소스에 대한 임시 액세스를 제공하는 것이 좋습니다. AWS

IAM 역할은 계정에 생성할 수 있는, 특정 권한을 지닌 IAM 자격 증명입니다. IAM 역할은 자격 증명이 수행할 수 있는 작업과 수행할 수 없는 작업을 결정하는 권한 정책을 가진 AWS ID라는 점에서 IAM 사용자와 유사합니다. AWS그러나 역할은 한 사람하고만 연관되지 않고 해당 역할이 필요한 사람이라면 누구든지 맡을 수 있어야 합니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준 장기 자격 증명이 없습니다. 대신에 역할을 맡은 사람에게는 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다.

페더레이션 아이덴티티에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의해주면 됩니다. 페더레이션 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요.

IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.

계정의 IAM 역할을 사용하여 계정 리소스에 액세스할 수 있는 다른 AWS 계정 권한을 부여할 수 있습니다. 예를 들어 IAM 사용 설명서의 자습서: IAM 역할 AWS 계정 사용에 대한 액세스 위임을 참조하십시오.

Sections

작동 방식

Amazon MWAA 환경에서 사용되는 리소스 및 서비스는 모든 AWS Identity and Access Management (IAM) 개체가 액세스할 수 있는 것은 아닙니다. Apache Airflow 사용자에게 이러한 리소스에 액세스할 수 있는 권한을 부여하는 정책을 생성해야 합니다. 예를 들면, Apache Airflow 개발 팀에 액세스 권한을 부여해 주어야 합니다.

Amazon MWAA는 이러한 정책을 사용하여 사용자가 AWS 콘솔에서 또는 환경에서 사용하는 API를 통해 작업을 수행하는 데 필요한 권한을 가지고 있는지 확인합니다.

이 단원의 JSON 정책을 사용하여 IAM에서 Apache Airflow 사용자를 위한 정책을 생성한 다음, 해당 정책을 IAM의 사용자, 그룹 또는 역할에 연결할 수 있습니다.

- Amazon MWAA FullConsole 액세스 — 이 정책을 사용하여 Amazon MWAA 콘솔에서 환경을 구성할 수 있는 권한을 부여합니다.
- Amazon MWAA FullApi 액세스 — 이 정책을 사용하면 환경을 관리하는 데 사용되는 모든 Amazon MWAA API에 대한 액세스 권한을 부여할 수 있습니다.

- Amazon MWAA ReadOnly 액세스 — 이 정책을 사용하면 Amazon MWAA 콘솔 환경에서 사용하는 리소스를 볼 수 있는 액세스 권한을 부여할 수 있습니다.
- Amazon MWAA WebServer 액세스 — 이 정책을 사용하여 Apache Airflow 웹 서버에 대한 액세스 권한을 부여할 수 있습니다.
- AmazonMWAA AirflowCli 액세스 — 이 정책을 사용하면 Apache Airflow CLI 명령을 실행할 수 있는 액세스 권한을 부여할 수 있습니다.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 다음 AWS IAM Identity Center분야의 사용자 및 그룹:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 제공자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

전체 콘솔 액세스 정책: 아마존 FullConsole MWAA 액세스

Amazon MWAA 콘솔에서 환경을 구성해야 하는 사용자는 `AmazonMWAAFullConsoleAccess` 권한 정책에 액세스해야 할 수 있습니다.

> **Note**
>
> 전체 콘솔 액세스 정책에는 `iam:PassRole`을 수행할 수 있는 권한이 포함되어야 합니다. 이를 통해 사용자는 서비스 연결 역할 및 실행 역할을 Amazon MWAA에 전달할 수 있습니다. Amazon MWAA는 사용자를 대신하여 다른 AWS 서비스를 호출하기 위해 각 역할을 맡습니다. 다음 예제는 `iam:PassedToService` 조건 키를 사용하여 Amazon MWAA 서비스 주체(`airflow.amazonaws.com`)를 역할 전달이 가능한 서비스로 지정합니다.

에 대한 *iam:PassRole* 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 역할을 전달할 수 있는 사용자 권한 부여를 참조하십시오.

저장 중 암호화를 위해 AWS 소유 키를 사용하여 Amazon MWAA 환경을 만들고 관리하려면 다음 정책을 사용합니다.

사용 AWS 소유 키

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreatePolicy"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:policy/service-role/MWAA-Execution-
Policy*"
```

```
      },
            {
         "Effect":"Allow",
         "Action":[
            "iam:AttachRolePolicy",
            "iam:CreateRole"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:role/service-role/AmazonMWAA*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:ListBucketVersions"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutObject",
            "s3:GetEncryptionConfiguration"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
         ],
```

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

저장 중 암호화를 위한 고객 관리형 키를 사용하여 Amazon MWAA 환경을 생성하고 관리하려면 다음 정책을 사용합니다. 고객 관리형 키를 사용하려면 IAM 보안 주체가 계정에 저장된 키를 사용하여 AWS KMS 리소스에 액세스할 수 있는 권한이 있어야 합니다.

고객 관리형 키 사용

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreatePolicy"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:policy/service-role/MWAA-Execution-
Policy*"
      },
            {
         "Effect":"Allow",
         "Action":[
            "iam:AttachRolePolicy",
            "iam:CreateRole"
         ],
         "Resource":"arn:aws:iam::YOUR_ACCOUNT_ID:role/service-role/AmazonMWAA*"
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:ListBucketVersions"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutObject",
            "s3:GetEncryptionConfiguration"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:CreateSecurityGroup"
         ],
```

```
            "Resource":"arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "kms:ListAliases"
            ],
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "kms:DescribeKey",
                "kms:ListGrants",
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource":"arn:aws:kms:*:YOUR_ACCOUNT_ID:key/YOUR_KMS_ID"
        },
        {
            "Effect":"Allow",
            "Action":"ec2:CreateVpcEndpoint",
            "Resource":[
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect":"Allow",
            "Action":[
                "ec2:CreateNetworkInterface"
            ],
            "Resource":[
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
```

```
}
```

전체 API 및 콘솔 액세스 정책: FullApi AmazonMWAA 액세스

사용자가 환경을 관리하는 데 사용되는 모든 Amazon MWAA API에 액세스해야 하는 경우 `AmazonMWAAFullApiAccess` 권한 정책에 액세스해야 할 수 있습니다. 이것이 Apache Airflow UI에 대한 액세스 권한을 부여해 주지는 않습니다.

> **Note**
>
> 전체 API 액세스 정책에는 `iam:PassRole`을 수행할 권한이 포함되어야 합니다. 이를 통해 사용자는 서비스 연결 역할 및 실행 역할을 Amazon MWAA에 전달할 수 있습니다. Amazon MWAA는 사용자를 대신하여 다른 AWS 서비스를 호출하기 위해 각 역할을 맡습니다. 다음 예제는 `iam:PassedToService` 조건 키를 사용하여 Amazon MWAA 서비스 주체(`airflow.amazonaws.com`)를 역할 전달이 가능한 서비스로 지정합니다.
> 에 대한 *`iam:PassRole`* 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 역할을 전달할 수 있는 사용자 권한 부여를 참조하십시오.

저장 중 암호화를 사용하여 Amazon MWAA 환경을 생성하고 관리하려면 다음 정책을 사용하십시오. AWS 소유 키

사용 AWS 소유 키

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

저장 중 암호화를 위한 고객 관리형 키를 사용하여 Amazon MWAA 환경을 생성하고 관리하려면 다음 정책을 사용합니다. 고객 관리형 키를 사용하려면 IAM 보안 주체가 계정에 저장된 키를 사용하여 AWS KMS 리소스에 액세스할 수 있는 권한이 있어야 합니다.

고객 관리형 키 사용

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

```
            "Action":[
               "ec2:DescribeSecurityGroups",
               "ec2:DescribeSubnets",
               "ec2:DescribeVpcs",
               "ec2:DescribeRouteTables"
            ],
            "Resource":"*"
         },
         {
            "Effect":"Allow",
            "Action":[
               "kms:DescribeKey",
               "kms:ListGrants",
               "kms:CreateGrant",
               "kms:RevokeGrant",
               "kms:Decrypt",
               "kms:Encrypt",
               "kms:GenerateDataKey*",
               "kms:ReEncrypt*"
            ],
            "Resource":"arn:aws:kms:*:YOUR_ACCOUNT_ID:key/YOUR_KMS_ID"
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:GetEncryptionConfiguration"
            ],
            "Resource":"arn:aws:s3:::*"
         },
         {
            "Effect":"Allow",
            "Action":"ec2:CreateVpcEndpoint",
            "Resource":[
               "arn:aws:ec2:*:*:vpc-endpoint/*",
               "arn:aws:ec2:*:*:vpc/*",
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:security-group/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "ec2:CreateNetworkInterface"
            ],
```

```
         "Resource":[
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:network-interface/*"
         ]
      }
   ]
}
```

읽기 전용 콘솔 액세스 정책: AmazonMWAA 액세스 ReadOnly

사용자는 Amazon MWAA 콘솔 환경 세부 정보 페이지에서 환경이 사용하는 리소스를 확인해야 하는 경우, `AmazonMWAAReadOnlyAccess` 권한 정책에 대한 액세스 권한이 필요할 수 있습니다. 이에 의해 사용자가 새 환경을 만들거나 기존 환경을 편집하거나 Apache Airflow UI를 볼 수 있도록 허용되는 것은 아닙니다.

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

아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer

사용자가 Apache Airflow UI에 액세스해야 하는 경우, `AmazonMWAAWebServerAccess` 권한 정책에 대한 액세스 권한이 필요할 수 있습니다. 이에 의해 사용자가 Amazon MWAA 콘솔에서 환경을 보거나 Amazon MWAA API를 사용하여 어떤 작업을 수행할 수 있도록 허용되는 것은 아닙니다. `{airflow-role}`에서 `Admin`, `Op`, `User`, `Viewer` 또는 `Public` 역할을 지정하여 웹 토큰 사용자의 액세스 수준을 사용자 지정합니다. 자세한 내용은 Apache Airflow 참조 가이드의 기본 역할을 참조하십시오.

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

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

> **Note**
>
> Amazon MWAA는 다섯 가지 기본 Apache Airflow 역할 기반 액세스 제어(RBAC) 역할과의 IAM 통합 기능을 제공합니다. 사용자 지정 Apache Airflow 역할을 사용하는 방법에 대한 자세한 내용은 the section called “튜토리얼: 사용자를 DAG의 하위 집합으로 제한” 단원을 참조하십시오.

아파치 에어플로우 CLI 정책: 아마존MWAA 액세스 AirflowCli

사용자는 Apache Airflow CLI 명령(예: `trigger_dag`)을 실행해야 하는 경우, `AmazonMWAAAirflowCliAccess` 권한 정책에 대한 액세스 권한이 필요할 수 있습니다. 이에 의해 사용자가 Amazon MWAA 콘솔에서 환경을 보거나 Amazon MWAA API를 사용하여 어떤 작업을 수행할 수 있도록 허용되는 것은 아닙니다.

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

JSON 정책 생성

JSON 정책을 생성하여 이 정책을 IAM 콘솔의 사용자, 역할 또는 그룹에 연결할 수 있습니다. 다음 단계에서는 IAM에서 JSON 정책을 생성하는 방법을 설명합니다.

JSON 정책 생성

1. IAM 콘솔에서 정책 페이지를 엽니다.
2. 정책 생성을 선택합니다.
3. JSON 탭을 선택합니다.
4. JSON 정책을 추가합니다.
5. 정책 검토를 선택합니다.
6. 이름 및 설명 텍스트 필드에 값을 입력합니다(선택 사항).

   예를 들면, 정책 `AmazonMWAAReadOnlyAccess`의 이름을 지정할 수 있습니다.
7. 정책 생성(Create policy)을 선택합니다.

개발자 그룹에 정책을 연결하는 사용 사례 예시

Apache Airflow 개발 팀의 모든 개발자에게 권한을 부여하기 위해 IAM에서 이름이 `AirflowDevelopmentGroup`으로 지정된 그룹을 사용하고 있다고 가정해 보겠습니다. 이러한 사용자는`AmazonMWAAFullConsoleAccess`, `AmazonMWAAAirflowCliAccess`, 및 `AmazonMWAAWebServerAccess` 권한 정책에 액세스할 수 있어야 합니다. 이 섹션에서는 IAM에서 그룹을 생성하고, 정책을 생성 및 연결하며, 이 그룹을 IAM 사용자와 연결하는 방법을 설명합니다. 이 단계에서는 AWS 소유 키를 사용한다고 가정합니다.

아마존 MWAA 정책을 FullConsoleAccess 만들려면

1. 아마존 MWAA 액세스 정책을 다운로드하십시오. FullConsoleAccess
2. IAM 콘솔에서 정책 페이지를 엽니다.
3. 정책 생성을 선택합니다.
4. JSON 탭을 선택합니다.
5. `AmazonMWAAFullConsoleAccess`에 대한 JSON 정책을 붙여넣습니다.
6. 다음 값을 대체합니다.

   a. *`{your-account-id} – ## ID (#`*:) AWS `0123456789`

b. *{your-kms-id}* - 고객 관리형 키에 대한 고유 식별자로, 저장 중 암호화에 고객 관리형 키를 사용하는 경우에만 적용됩니다.

7. 정책 검토를 선택합니다.
8. 이름에 AmazonMWAAFullConsoleAccess를 입력합니다.
9. 정책 생성(Create policy)을 선택합니다.

아마존 MWAA 정책을 만들려면 WebServerAccess

1. 아마존 MWAA 액세스 정책을 다운로드하십시오. WebServerAccess
2. IAM 콘솔에서 정책 페이지를 엽니다.
3. 정책 생성을 선택합니다.
4. JSON 탭을 선택합니다.
5. AmazonMWAAWebServerAccess에 대한 JSON 정책을 붙여넣습니다.
6. 다음 값을 대체합니다.

   a. *{your-region}* - Amazon MWAA 환경의 리전(예: us-east-1)

   b. *{your-account-id} – ## ID (#*:) AWS 0123456789

   c. *{your-environment-name}* - 사용자의 Amazon MWAA 환경 이름(예: MyAirflowEnvironment)

   d. *{airflow-role}* - Admin Apache Airflow의 기본 역할

7. 정책 검토를 선택합니다.
8. 이름에 AmazonMWAAWebServerAccess를 입력합니다.
9. 정책 생성(Create policy)을 선택합니다.

아마존 MWAA 정책을 생성하려면 AirflowCliAccess

1. 아마존 MWAA 액세스 정책을 다운로드하십시오. AirflowCliAccess
2. IAM 콘솔에서 정책 페이지를 엽니다.
3. 정책 생성을 선택합니다.
4. JSON 탭을 선택합니다.
5. AmazonMWAAAirflowCliAccess에 대한 JSON 정책을 붙여넣습니다.
6. 정책 검토를 선택합니다.
7. 이름에 AmazonMWAAAirflowCliAccess를 입력합니다.

8. 정책 생성(Create policy)을 선택합니다.

그룹을 생성하려면

1. IAM 콘솔의 그룹 페이지를 엽니다.
2. `AirflowDevelopmentGroup` 이름을 입력합니다.
3. 다음 단계를 선택합니다.
4. 필터에서 결과를 필터링하려면 `AmazonMWAA`를 입력합니다.
5. 생성한 세 가지 정책을 선택합니다.
6. 다음 단계를 선택합니다.
7. 그룹 생성을 선택합니다.

사용자와 연결하려면

1. IAM 콘솔에서 사용자 페이지 엽니다.
2. 사용자를 선택합니다.
3. 그룹을 선택합니다.
4. 그룹에 사용자 추가를 선택합니다.
5. 그룹을 선택합니다. AirflowDevelopment
6. 그룹에 추가를 선택합니다.

다음 단계

- 아파치 에어플로우 액세스에서 Apache Airflow UI에 액세스하기 위한 토큰을 생성하는 방법을 알아봅니다.
- IAM 정책 생성에서 IAM 정책 생성에 대한 자세한 내용을 확인합니다.

Amazon MWAA의 서비스 연결 역할

Apache Airflow용 Amazon 관리형 워크플로는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon MWAA에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon MWAA에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로, 서비스 연결 역할은 Amazon MWAA를 더 쉽게 설정할 수 있습니다. Amazon MWAA에서 서비스 연결 역할의 권한을 정의하므로 달리 정의되지 않은 한, Amazon MWAA만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon MWAA 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 표시된 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon MWAA에 대한 서비스 연결 역할 권한

Amazon MWAA는 이름이 지정된 서비스 연결 역할을 사용합니다. 계정에서 생성된 서비스 연결 역할은 Amazon `AWSServiceRoleForAmazonMWAA` MWAA에 다음 서비스에 대한 액세스 권한을 부여합니다. AWS

- Amazon CloudWatch 로그 (CloudWatch 로그) — 아파치 에어플로우 로그의 로그 그룹을 생성합니다.
- Amazon CloudWatch (CloudWatch) — 환경 및 기본 구성 요소와 관련된 지표를 계정에 게시합니다.
- Amazon Elastic Compute Cloud(Amazon EC2) - 다음과 같은 리소스를 생성하려면
  - 아파치 에어플로우 스케줄러 및 작업자가 사용할 관리형 Amazon AWS Aurora PostgreSQL 데이터베이스 클러스터를 위한 VPC의 Amazon VPC 엔드포인트입니다.
  - Apache Airflow 웹 서버에 대해 프라이빗 네트워크 옵션을 선택한 경우, 웹 서버에 대한 네트워크 액세스를 지원하는 추가 Amazon VPC 엔드포인트입니다.
  - Amazon VPC의 탄력적 네트워크 인터페이스 (ENI) 를 사용하면 Amazon VPC에 호스팅된 AWS 리소스에 네트워크로 액세스할 수 있습니다.

다음 신뢰 정책을 이용하면 서비스 주체가 서비스 연결 역할을 맡을 수 있습니다. Amazon MWAA의 서비스 주체는 정책에 명시된 `airflow.amazonaws.com` 바와 같습니다.

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

```
                "Service": "airflow.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

이름이 AmazonMWAAServiceRolePolicy인 역할 권한 정책에 따라 Amazon MWAA에서 지정된 리소스에서 다음 작업을 완료할 수 있습니다.

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

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

Amazon MWAA에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console AWS CLI, 또는 AWS API를 사용하여 새 Amazon MWAA 환경을 생성하면 Amazon MWAA가 사용자를 대신하여 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 다른 환경을 생성할 때는 Amazon MWAA에서 서비스 연결 역할을 자동으로 다시 생성합니다.

Amazon ECS에 대한 서비스 연결 역할 편집

Amazon MWAA에서는 AWSServiceRoleForAmazonMWAA 서비스 연결 역할을 편집할 수 없습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

Amazon ECS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다.

Amazon MWAA 환경을 삭제하면 Amazon MWAA에서 서비스의 일부로 사용하는 모든 관련 리소스를 삭제합니다. 그러나, 서비스 연결 역할을 삭제하기 위해서는 Amazon MWAA에서 환경 삭제를 완료할 때까지 기다려야 합니다. Amazon MWAA에서 환경을 삭제하기 전에 서비스 연결 역할을 삭제하면 Amazon MWAA에서 해당 환경의 모든 관련 리소스를 삭제하지 못할 수 있습니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 서비스 연결 역할을 삭제합니다. AWSServiceRoleForAmazonMWAA 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon MWAA 서비스 연결 역할을 위해 지원 받는 리전

Amazon MWAA는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 Amazon Managed Workflows for Apache Airflow의 엔드포인트 및 할당량을 참조하십시오.

정책 업데이트

변경 사항	설명	날짜
Amazon MWAA의 서비스 연결 역할 권한 정책 업데이트	AmazonMWAAServiceRolePolicy — Amazon MWAA에서는 서비스 연결 역할에 대한 권한 정책을 업데이트하여 서비스의 기본 리소스에 관련된 추가 지표를 고객 계정에 게시할 수 있는 권한을 Amazon MWAA에 부여합니다. 이러한 새 지표는 AWS/MWAA 아래에 게시됩니다.	2022년 11월 18일
Amazon MWAA의 변경 사항 추적 시작	Amazon MWAA는 AWS 관리형 서비스 연결 역할 권한 정책에 대한 변경 사항을 추적하기 시작했습니다.	2022년 11월 18일

Amazon MWAA 실행 역할

실행 역할은 Apache Airflow용 Amazon Managed Workflows에 사용자 대신 다른 AWS 서비스의 리소스를 호출할 수 있는 권한을 부여하는 권한 정책이 있는 AWS Identity and Access Management (IAM) 역할입니다. 여기에는 Amazon S3 버킷, AWS 소유 키 및 CloudWatch 로그와 같은 리소스가 포함될 수 있습니다. Amazon MWAA 환경에는 환경당 하나의 실행 역할이 필요합니다. 이 페이지에서는 Amazon MWAA가 사용자 환경에서 사용하는 다른 AWS 리소스에 액세스할 수 있도록 사용자 환경의 실행 역할을 사용하고 구성하는 방법을 설명합니다.

목차

실행 역할 개요

Amazon MWAA가 사용자 환경에서 사용하는 다른 AWS 서비스를 사용할 수 있는 권한은 실행 역할에서 얻습니다. Amazon MWAA 실행 역할에는 환경에서 사용하는 다음 AWS 서비스에 대한 권한이 필요합니다.

- Amazon CloudWatch (CloudWatch) — 아파치 에어플로우 지표 및 로그를 전송합니다.
- Amazon Simple Storage Service(S3) - 환경의 DAG 코드 및 지원 파일(예: `requirements.txt`)을 파싱합니다.

- Amazon Simple Queue Service(Amazon SQS) - 환경의 Apache Airflow 작업을 Amazon MWAA에서 소유하고 있는 Amazon SQS 대기열에 저장합니다.
- AWS Key Management Service (AWS KMS) — 사용자 환경의 데이터 암호화용 (AWS 소유 키 또는 고객 관리 키 사용).

> **Note**
>
> Amazon MWAA에서 AWS 관리형 KMS 키를 사용하여 데이터를 암호화하도록 선택한 경우 Amazon MWAA 실행 역할에 첨부된 정책에서 Amazon SQS를 통해 계정 외부에 저장된 임의의 KMS 키에 대한 액세스 권한을 부여하는 권한을 정의해야 합니다. 환경의 실행 역할이 임의의 KMS 키에 액세스하게 하려면 다음 두 가지 조건이 필요합니다.
>
> - 타사 계정의 KMS 키는 자체의 리소스 정책을 통해 이러한 크로스 계정 액세스를 허용해야 합니다.
> - DAG 코드는 타사 계정에서 `airflow-celery-`로 시작해야 하며 암호화에 동일한 KMS 키를 사용하는 Amazon SQS 대기열에 액세스해야 합니다.
>
> 리소스에 대한 크로스 계정 액세스와 관련된 위험을 줄이려면, DAG에 배치된 코드를 검토하여 사용자의 워크플로가 사용자 계정 외부 임의의 Amazon SQS 대기열에 액세스하지 않도록 하는 것이 좋습니다. 또한, 사용자 계정에 저장된 고객 관리형 KMS 키를 사용하여 Amazon MWAA에서 암호화를 관리할 수 있습니다. 이로 인해 사용자 환경의 실행 역할이 사용자 계정의 KMS 키에만 액세스하는 것으로 제한됩니다.
>
> 암호화 옵션을 선택한 후에는 기존 환경에 대한 선택사항을 변경할 수 없다는 점에 유의하십시오.

실행 역할에는 다음과 같은 IAM 작업에 대한 권한도 필요합니다.

- `airflow:PublishMetrics`— Amazon MWAA에서 환경의 상태를 모니터링할 수 있도록 합니다.

권한은 기본적으로 연결됩니다.

Amazon MWAA 콘솔의 기본 옵션을 사용하여 실행 역할과 AWS 소유 키를 생성한 다음 이 페이지의 단계를 이용하여 사용자의 실행 역할에 권한 정책을 추가할 수 있습니다.

- 콘솔에서 새 역할 생성 옵션을 선택하면 Amazon MWAA에서 환경에 필요한 최소 권한을 사용자의 실행 역할에 부여합니다.
- 경우에 따라, Amazon MWAA에서 최대 권한을 부여하기도 합니다. 예를 들면, 환경을 만들 때는 Amazon MWAA 콘솔에서 실행 역할을 생성하는 옵션을 선택하는 것이 좋습니다. Amazon MWAA는

실행 역할의 정규식 패턴을 사용하여 모든 CloudWatch 로그 그룹에 대한 권한 정책을 자동으로 추가합니다. `"arn:aws:logs:your-region:your-account-id:log-group:airflow-your-environment-name-*"`

다른 서비스를 사용할 권한을 추가하는 방법 AWS

Amazon MWAA는 환경이 만들어진 후에는 기존 실행 역할에 권한 정책을 추가하거나 편집할 수 없습니다. 사용자는 환경에 필요한 추가 권한 정책으로 실행 역할을 업데이트해야 합니다. 예를 들어 DAG에 AWS Glue대한 액세스가 필요한 경우 Amazon MWAA는 환경에 필요한 권한을 자동으로 감지하거나 실행 역할에 권한을 추가할 수 없습니다.

다음 두 가지 방법으로 실행 역할에 권한을 추가할 수 있습니다.

- 실행 역할에 필요한 JSON 정책을 인라인으로 수정하는 방법 이 페이지의 샘플 JSON 정책 문서를 사용하여 IAM 콘솔에서 실행 역할의 JSON 정책을 추가하거나 바꿀 수 있습니다.
- AWS 서비스에 대한 JSON 정책을 생성하여 실행 역할에 추가함으로써 가능합니다. 이 페이지의 단계를 사용하여 AWS 서비스의 새 JSON 정책 문서를 IAM 콘솔의 실행 역할에 연결할 수 있습니다.

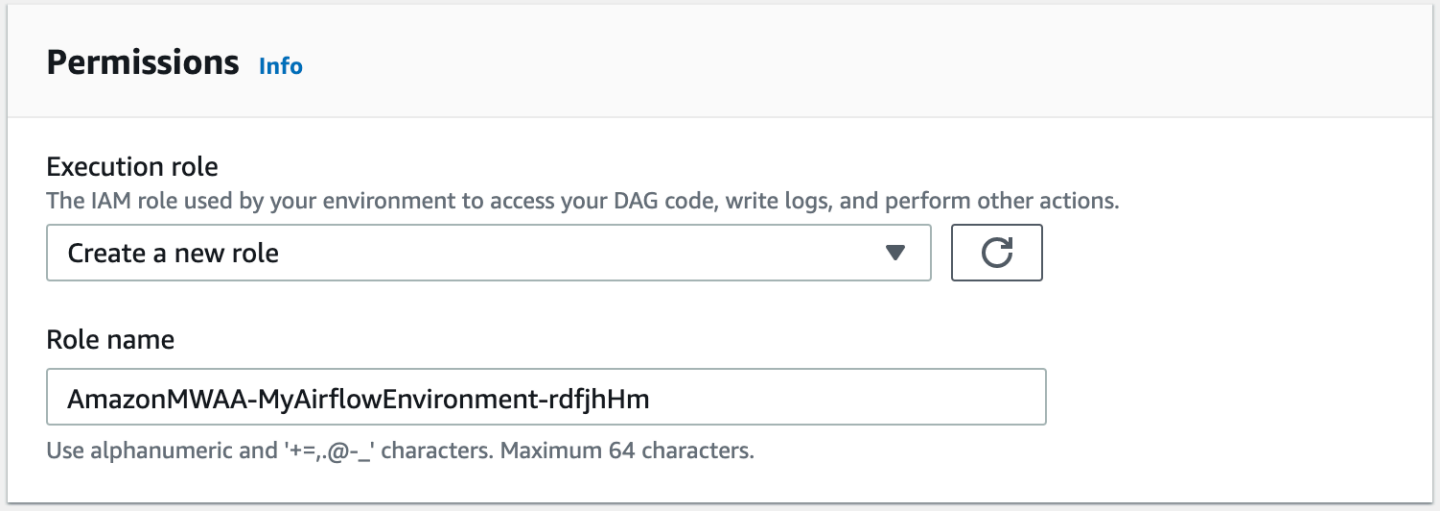
실행 역할이 이미 환경에 연결되어 있다고 가정하면, Amazon MWAA는 추가된 권한 정책을 즉시 사용할 수 있습니다. 즉, 실행 역할에서 필요한 권한을 제거하면 DAG가 실패할 수 있음을 의미하는 것이기도 합니다.

새 실행 역할을 연결하는 방법

언제든지 사용자 환경의 실행 역할을 변경할 수 있습니다. 새 실행 역할이 아직 사용자의 환경에 연결되지 않은 경우, 이 페이지의 단계를 사용하여 새 실행 역할 정책을 만들고 해당 역할을 환경에 연결합니다.

새 역할 생성

기본 설정으로, Amazon MWAA는 사용자를 대신하여 데이터 암호화를 위한 AWS 소유 키 및 실행 역할을 생성합니다. 환경을 생성할 때는 Amazon MWAA 콘솔에서 기본 옵션을 선택할 수 있습니다. 다음 이미지는 환경에 대한 실행 역할을 생성하기 위한 기본 옵션을 보여줍니다.



실행 역할 정책 보기 및 업데이트

Amazon MWAA 콘솔에서 사용자 환경의 실행 역할을 확인하고, IAM 콘솔에서 그 역할에 대한 JSON 정책을 업데이트할 수 있습니다.

실행 역할 정책 업데이트

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. IAM에서 권한 페이지를 열려면 권한 창에서 실행 역할을 선택합니다.
4. 실행 역할 이름을 선택하여 권한 정책을 엽니다.
5. 정책 편집을 선택합니다.
6. JSON 탭을 선택합니다.
7. JSON 정책을 업데이트합니다.
8. 정책 검토를 선택합니다.
9. 변경 사항 저장을 선택합니다.

JSON 정책을 연결하여 다른 서비스를 사용할 수 있습니다. AWS

AWS 서비스에 대한 JSON 정책을 생성하여 실행 역할에 추가할 수 있습니다. 예를 들면, 다음 JSON 정책을 연결하여 AWS Secrets Manager에 있는 모든 리소스에 읽기 전용 액세스 권한을 부여할 수 있습니다.

```
{
```

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

실행 역할에 대한 정책 연결

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 권한 창에서 실행 역할을 선택합니다.
4. 정책 연결을 선택합니다.
5. 정책 생성(Create policy)을 선택합니다.
6. JSON을 선택합니다.
7. JSON 정책을 붙여넣습니다.
8. 다음: 태그를 선택한 후 다음: 검토를 선택합니다.
9. 정책의 이름(예: `SecretsManagerReadPolicy`)과 설명을 입력합니다.
10. 정책 생성(Create policy)을 선택합니다.

Amazon S3 버킷에 대한 액세스 권한을 계정 수준의 퍼블릭 액세스 차단으로 부여합니다.

`PutPublicAccessBlock`Amazon S3 작업을 사용하여 계정의 모든 버킷에 대한 액세스를 차단하는 것이 좋을 수 있습니다. 계정의 모든 버킷에 대한 액세스를 차단할 때는 사용자의 환경 실행 역할은 권한 정책에 `s3:GetAccountPublicAccessBlock` 작업이 포함되게 해야 합니다.

다음 예는 계정의 모든 Amazon S3 버킷에 대한 액세스를 차단할 때 사용자의 실행 역할에 연결해야 하는 정책을 보여줍니다.

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

Amazon S3 버킷에 대한 액세스를 제한하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 Storage에 대한 퍼블릭 액세스 차단을 참조하십시오.

Apache Airflow 연결 사용

또한 Apache Airflow 연결을 만들고 Apache Airflow 연결 객체에 실행 역할과 해당 ARN을 지정할 수 있습니다. 자세한 내용은 Apache Airflow에 대한 연결 관리 단원을 참조하십시오.

실행 역할에 대한 샘플 JSON 정책

이 섹션의 샘플 권한 정책은 기존 실행 역할에 사용된 권한 정책을 대체하거나 새 실행 역할을 생성하여 사용자 환경에 사용할 수 있는 두 가지 정책을 보여줍니다. 이러한 정책에는 Apache Airflow 로그 그룹, Amazon S3 버킷 및 Amazon MWAA 환경에 대한 리소스 ARN 자리 표시자가 포함되어 있습니다.

예제 정책을 복사하여 샘플 ARN 또는 자리 표시자를 교체한 다음, JSON 정책을 사용하여 실행 역할을 생성하거나 업데이트하는 것이 좋습니다. 예를 들면, `{your-region}`을 `us-east-1`로 바꾸는 경우가 있습니다.

고객 관리형 키에 대한 샘플 정책

다음 예는 고객 관리형 키에 사용할 수 있는 실행 역할 정책을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
```

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

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

다음으로, Amazon MWAA에서 이 역할을 맡도록 허용하여 사용자를 대신해서 작업을 수행할 수 있게 해야 합니다. 이를 위해서는 IAM 콘솔을 사용하여 이 실행 역할의 신뢰할 수 있는 객체 목록에 "airflow.amazonaws.com" 및 "airflow-env.amazonaws.com" 서비스 주체를 추가하거나, AWS CLI를 사용하여 IAM Create-Role 명령을 통해 이 실행 역할에 대한 역할 수임 정책 문서에 이러한 서비스 주체를 배치하면 됩니다. 역할 가정 정책 문서의 예는 다음과 같습니다.

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

이어서, 다음 JSON 정책을 고객 관리형 키에 연결합니다. 이 정책은 kms:EncryptionContext조건 키 접두사를 사용하여 Logs의 Apache Airflow 로그 그룹에 대한 액세스를 허용합니다. CloudWatch

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

```
}
```

소유 키에 대한 샘플 정책 AWS

다음 예시는 AWS 소유 키에 사용할 수 있는 실행 역할 정책을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:{your-region}:{your-account-id}:environment/
{your-environment-name}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{your-s3-bucket-name}",
                "arn:aws:s3:::{your-s3-bucket-name}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
```

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

```
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:{your-account-id}:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "sqs.{your-region}.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

다음 단계

- Amazon MWAA 환경 액세스에서 사용자 및 Apache Airflow 사용자가 환경에 액세스하는 데 필요한 필수 권한에 대해 알아봅니다.
- 암호화를 위한 고객 관리형 키 사용에 대해 알아봅니다.
- 더 많은 고객 관리형 정책 예를 살펴봅니다.

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS크로스 서비스 사칭은 대리인 문제로 혼동될 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

Amazon MWAA가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 환경의 실행 역할에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을(를) 사용합니다. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount을 사용하세요.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 arn:aws:airflow:*:*123456789012*:environment/*입니다.

aws:SourceArn의 값은 실행 역할을 생성하려는 Amazon MWAA 환경 ARN이어야 합니다.

다음 예는 사용자 환경의 실행 역할 신뢰 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다. 새 실행 역할을 생성할 때는 다음 신뢰 정책을 사용할 수 있습니다.

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

Apache Airflow 액세스 모드

Amazon Managed Workflows for Apache Airflow 콘솔에는 사용자 환경의 Apache Airflow 웹 서버에 대한 프라이빗 또는 퍼블릭 라우팅을 구성할 수 있는 내장 옵션이 포함되어 있습니다. 이 가이드에서는 Amazon Managed Workflows for Apache Airflow 환경에서 Apache Airflow 웹 서버에 사용할 수 있는 액세스 모드와, 프라이빗 네트워크 옵션을 선택할 경우 Amazon VPC에서 구성해야 하는 추가 리소스에 대해 설명합니다.

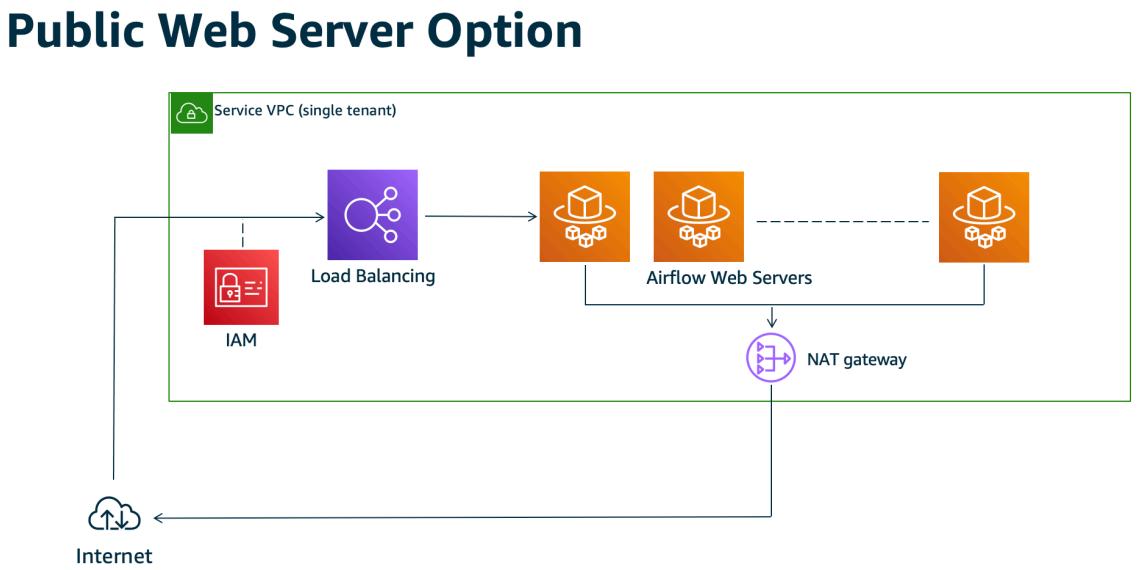
목차

Apache Airflow 액세스 모드

Apache Airflow 웹 서버의 프라이빗 또는 퍼블릭 라우팅을 선택할 수 있습니다. 프라이빗 라우팅을 활성화하려면 프라이빗 네트워크를 선택합니다. 이렇게 하면 Amazon VPC 내에서 Apache Airflow 웹 서버에 대한 사용자 액세스가 제한됩니다. 퍼블릭 라우팅을 활성화하려면 퍼블릭 네트워크를 선택합니다. 이렇게 함으로써 사용자는 인터넷을 통해 Apache Airflow 웹 서버에 액세스할 수 있습니다.

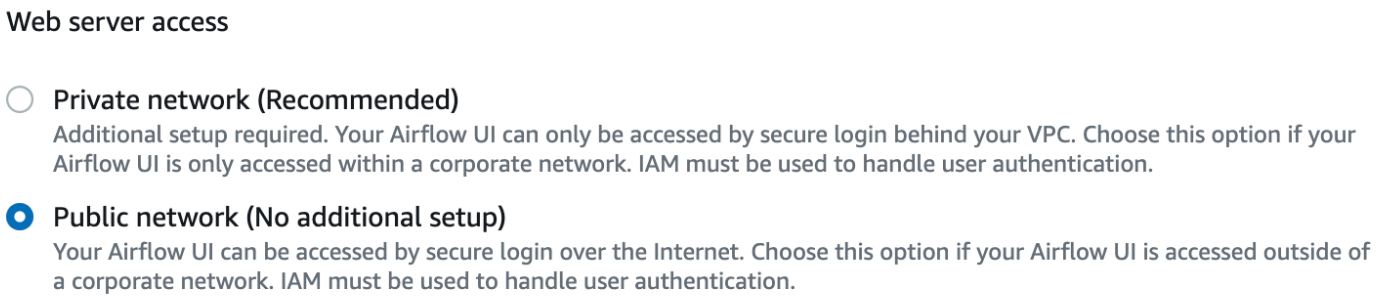
퍼블릭 네트워크

다음 아키텍처 다이어그램은 퍼블릭 웹 서버가 있는 Amazon MWAA 환경을 보여줍니다.



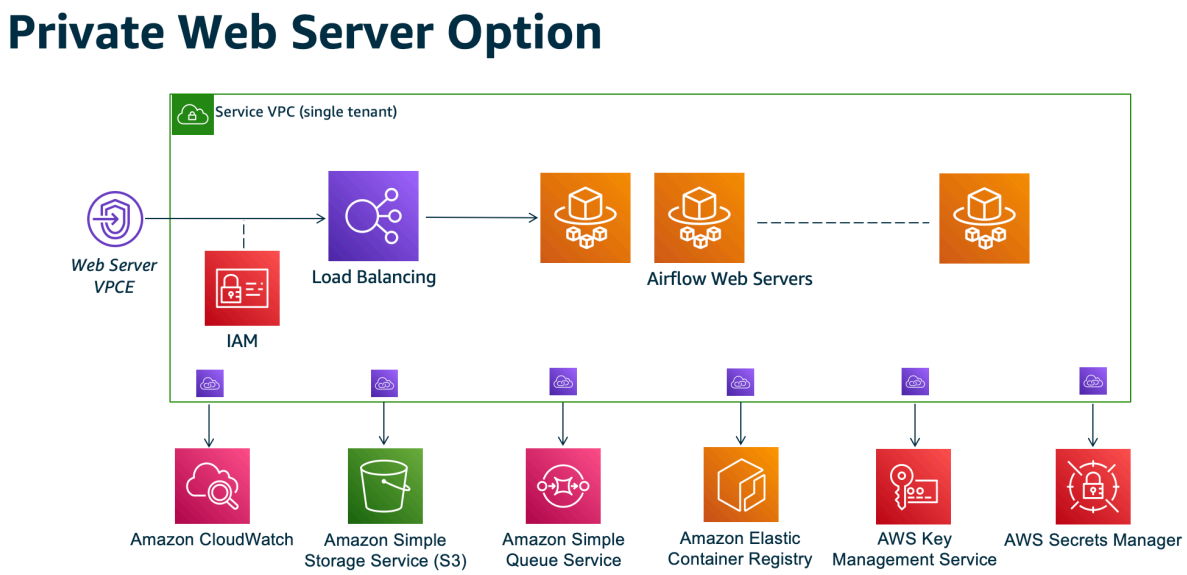
퍼블릭 네트워크 액세스 모드를 사용하면 사용자 환경의 IAM 정책에 대한 액세스 권한이 부여된 사용자가 인터넷을 통해 Apache Airflow UI에 액세스할 수 있습니다.

다음 이미지는 Amazon MWAA 콘솔에서 퍼블릭 네트워크 옵션을 찾을 수 있는 위치를 보여줍니다.



프라이빗 네트워크

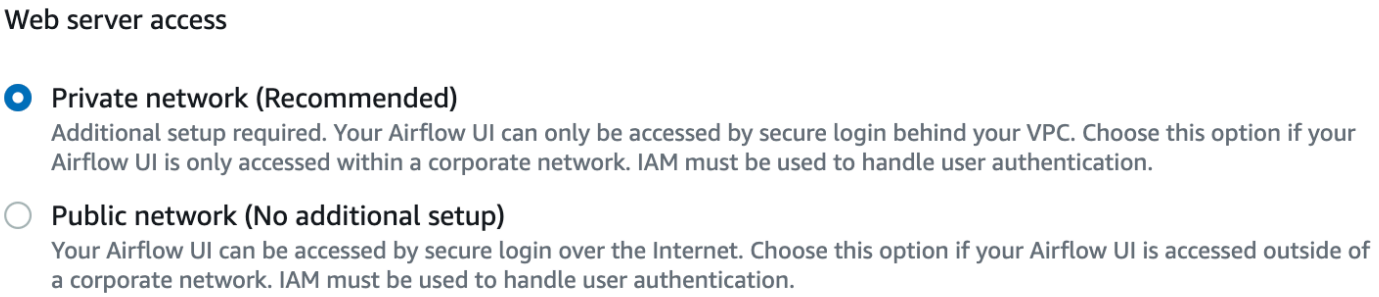
다음 아키텍처 다이어그램은 프라이빗 웹 서버가 있는 Amazon MWAA 환경을 보여줍니다.



프라이빗 네트워크 액세스 모드를 사용하면 Amazon VPC 내에서 사용자 환경의 IAM 정책에 대한 액세스 권한을 부여 받은 사용자만 Apache Airflow UI에 액세스할 수 있도록 제한됩니다.

프라이빗 웹 서버 액세스가 가능한 환경을 만들 때는 모든 종속성을 Python 휠 아카이브(`.whl`)에서 패키징한 다음 `requirements.txt`에서 `.whl`을 참조해야 합니다. 휠을 사용하여 종속성을 패키징하고 설치하는 방법에 대한 지침은 Python 휠을 이용한 종속성 관리를 참조하십시오.

다음 이미지는 Amazon MWAA 콘솔에서 프라이빗 네트워크 옵션을 찾을 수 있는 위치를 보여줍니다.



액세스 모드 개요

이 섹션에서는 퍼블릭 네트워크 또는 프라이빗 네트워크 액세스 모드를 선택할 때 Amazon VPC에 생성되는 VPC 엔드포인트(AWS PrivateLink)에 대해 설명합니다.

퍼블릭 네트워크 액세스 모드

Apache Airflow 웹 서버의 퍼블릭 네트워크 액세스 모드를 선택한 경우, 네트워크 트래픽은 인터넷을 통해 공개적으로 라우팅됩니다.

- Amazon MWAA는 Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 위한 VPC 인터페이스 엔드포인트를 생성합니다. 엔드포인트는 프라이빗 서브넷에 매핑된 가용 영역에 생성되며 다른 계정과는 독립적입니다. AWS
- 그러면 Amazon MWAA에서 프라이빗 서브넷의 IP 주소를 인터페이스 엔드포인트에 바인딩합니다. 이는 Amazon VPC의 각 가용 영역에서 단일 IP를 바인딩하는 모범 사례를 지원하도록 설계되었습니다.

프라이빗 네트워크 액세스 모드

Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택한 경우, 네트워크 트래픽은 Amazon VPC 내에서 프라이빗로 라우팅됩니다.

- Amazon MWAA는 Apache Airflow 웹 서버를 위한 VPC 인터페이스 엔드포인트와, Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 위한 인터페이스 엔드포인트를 생성합니다. 엔드포인트는 프라이빗 서브넷에 매핑된 가용 영역에서 생성되며 다른 계정과는 독립적입니다. AWS
- 그러면 Amazon MWAA에서 프라이빗 서브넷의 IP 주소를 인터페이스 엔드포인트에 바인딩합니다. 이는 Amazon VPC의 각 가용 영역에서 단일 IP를 바인딩하는 모범 사례를 지원하도록 설계되었습니다.

자세한 내용은 the section called “Amazon VPC 및 Apache Airflow 액세스 모드의 사용 사례 예시” 단원을 참조하십시오.

프라이빗 및 퍼블릭 액세스 모드 설정

다음 섹션에서는 환경에 맞게 선택한 Apache Airflow 액세스 모드에 따라 필요한 추가 설정 및 구성을 설명합니다.

퍼블릭 네트워크 설정

Apache Airflow 웹 서버의 퍼블릭 네트워크 옵션을 선택하면, 환경을 생성한 후 Apache Airflow UI 사용을 시작할 수 있습니다.

다음 단계를 수행하여 사용자에 대한 액세스와 사용자 환경에서 다른 서비스를 사용할 수 있는 권한을 구성해야 합니다. AWS

1. 권한 추가 Amazon MWAA에서 다른 AWS 서비스를 사용하려면 권한이 필요합니다. 환경을 만들면 Amazon MWAA는 Amazon Elastic Container 레지스트리 (Amazon ECR), 로그 및 Amazon EC2에 대한 특정 IAM 작업을 사용할 수 있도록 하는 서비스 연결 역할을 생성합니다. CloudWatch

   실행 역할에 권한을 추가하여 이러한 서비스에 추가 작업을 사용하거나 다른 AWS 서비스를 사용할 수 있는 권한을 추가할 수 있습니다. 자세한 내용은 Amazon MWAA 실행 역할 단원을 참조하십시오.
2. 사용자 정책 생성. 사용자가 환경 및 Apache Airflow UI에 대한 액세스를 구성하려면 여러 IAM 정책을 생성해야 할 수 있습니다. 자세한 내용은 Amazon MWAA 환경 액세스 단원을 참조하십시오.

프라이빗 네트워크 설정

Apache Airflow 웹 서버의 사설망 옵션을 선택하는 경우 사용자에 대한 액세스 권한, 사용자 환경에서 다른 AWS 서비스를 사용할 수 있도록 권한을 구성하고, 컴퓨터에서 Amazon VPC의 리소스에 액세스하는 메커니즘을 생성해야 합니다.

1. 권한 추가 Amazon MWAA에서 다른 AWS 서비스를 사용하려면 권한이 필요합니다. 환경을 만들면 Amazon MWAA는 Amazon Elastic Container 레지스트리 (Amazon ECR), 로그 및 Amazon EC2에 대한 특정 IAM 작업을 사용할 수 있도록 하는 서비스 연결 역할을 생성합니다. CloudWatch

실행 역할에 권한을 추가하여 이러한 서비스에 추가 작업을 사용하거나 다른 AWS 서비스를 사용할 수 있는 권한을 추가할 수 있습니다. 자세한 내용은 Amazon MWAA 실행 역할 단원을 참조하십시오.

2. 사용자 정책 생성. 사용자가 환경 및 Apache Airflow UI에 대한 액세스를 구성하려면 여러 IAM 정책을 생성해야 할 수 있습니다. 자세한 내용은 Amazon MWAA 환경 액세스 단원을 참조하십시오.
3. 네트워크 액세스 활성화 Apache Airflow 웹 서버의 VPC 엔드포인트(AWS PrivateLink)에 연결하려면 Amazon VPC에서 메커니즘을 생성해야 합니다. 예를 들면, AWS Client VPN를 사용하여 컴퓨터에서 VPN 터널을 만들 수 있습니다.

Apache Airflow 웹 서버의 VPC 엔드포인트 액세스(프라이빗 네트워크 액세스)

프라이빗 네트워크 옵션을 선택한 경우, Amazon VPC에서 Apache Airflow 웹 서버의 VPC 엔드포인트(AWS PrivateLink)에 액세스할 수 있는 메커니즘을 생성해야 합니다. 이러한 리소스에는 Amazon MWAA 환경과 동일한 Amazon VPC, VPC 보안 그룹 및 프라이빗 서브넷을 사용하는 것이 좋습니다.

자세한 내용은 VPC 엔드포인트의 액세스 관리를 참조하십시오.

아파치 에어플로우 액세스

Amazon MWAA에서는 아파치 에어플로우 사용자 인터페이스 (UI) 콘솔, 아파치 에어플로우 CLI, 아파치 에어플로우 REST API 등 다양한 방법을 사용하여 아파치 에어플로우 환경에 액세스할 수 있습니다. Amazon MWAA 콘솔을 사용하여 Apache Airflow UI에서 DAG를 보고 호출하거나, Amazon MWAA API를 사용하여 토큰을 얻고 DAG를 호출할 수 있습니다. 이 섹션에서는 Apache Airflow UI에 액세스하는 데 필요한 권한, 명령 쉘에서 직접 Amazon MWAA API를 호출하기 위한 토큰을 생성하는 방법, Apache Airflow CLI에서 지원되는 명령에 대해 설명합니다.

주제

필수 조건

다음 섹션에서는 이 섹션의 명령과 스크립트를 사용하는 데 필요한 예비 단계를 설명합니다.

액세스

- AWS 에서 Amazon MWAA 권한 정책에 대한 계정 액세스 AWS Identity and Access Management (IAM) 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer
- AWS Amazon MWAA 권한 정책에 대한 AWS Identity and Access Management (IAM) 계정 액세스 전체 API 및 콘솔 액세스 정책: FullApi AmazonMWAA 액세스

AWS CLI

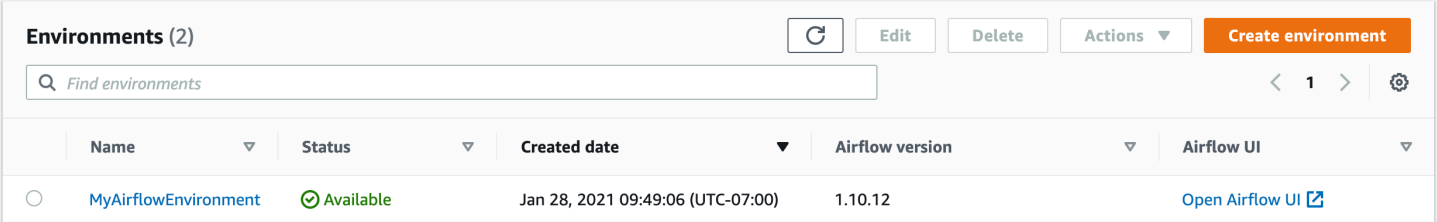
AWS Command Line Interface (AWS CLI) 는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있게 해주는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI — 버전 2를 설치합니다.

- AWS CLI — 빠른 구성 `aws configure`.

Open Airflow UI

다음 이미지는 Amazon MWAA 콘솔의 Apache Airflow UI에 대한 링크를 보여줍니다.



Apache Airflow에 로그인

Apache Airflow UI를 보려면 AWS Identity and Access Management (IAM) 의 AWS 계정에 대한 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer 권한이 필요합니다.

Apache Airflow UI에 액세스하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. Airflow UI 열기를 선택합니다.

아파치 에어플로우 웹 서버 액세스 토큰 생성

이 페이지의 명령을 사용하여 웹 서버 액세스 토큰을 생성할 수 있습니다. 액세스 토큰을 사용하면 Amazon MWAA 환경에 액세스할 수 있습니다. 예를 들어, 토큰을 얻은 다음 Amazon MWAA API를 사용하여 프로그래밍 방식으로 DAG를 배포할 수 있습니다. 다음 섹션에는 bash 스크립트 AWS CLI, POST API 요청 또는 Python 스크립트를 사용하여 Apache Airflow 웹 로그인 토큰을 생성하는 단계가 포함되어 있습니다. 응답에 반환되는 토큰은 60초 동안 유효합니다.

목차

필수 조건

다음 섹션에서는 이 페이지의 명령과 스크립트를 사용하는 데 필요한 예비 단계를 설명합니다.

액세스

- AWS 에서 Amazon MWAA 권한 정책에 대한 계정 액세스 AWS Identity and Access Management (IAM). 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer
- AWS Amazon MWAA 권한 정책에 대한 AWS Identity and Access Management (IAM) 계정 액세스 전체 API 및 콘솔 액세스 정책: FullApi AmazonMWAA 액세스

AWS CLI

AWS Command Line Interface (AWS CLI) 는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있게 해주는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI — 버전 2를 설치합니다.
- AWS CLI — 빠른 구성 `aws configure`.

사용: AWS CLI

다음 예제에서는 의 create-web-login-token명령을 사용하여 Apache Airflow 웹 로그인 토큰을 생성합니다. AWS CLI

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

bash 스크립트 사용

다음 예제에서는 bash 스크립트를 사용하여 에서 create-web-login-token명령을 호출하여 Apache Airflow 웹 로그인 AWS CLI 토큰을 생성합니다.

1. 다음 코드 샘플의 내용을 복사하여 로컬에 `get-web-token.sh`로 저장합니다.

```
#!/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. YOUR_HOST_NAME 및 YOUR_ENVIRONMENT_NAME에 대해 ###으로 표시된 자리 표시자를 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다(https:// 없음.

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

3. (선택 사항) macOS 및 Linux 사용자는 다음 명령을 실행하여 스크립트가 실행 가능한지 확인해야 할 수 있습니다.

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

4. 다음 스크립트를 실행하여 웹 로그인 토큰을 얻습니다.

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

5. 명령 프롬프트에 다음이 표시되어야 합니다.

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

POST API 요청 사용

다음 예시에서는 POST API 요청을 사용하여 Apache Airflow 웹 로그인 토큰을 생성합니다.

1. 다음 URL을 복사하여 REST API 클라이언트의 URL 필드에 붙여넣습니다.

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

2. YOUR_HOST_NAME에 대해 ###으로 표시된 자리 표시자를 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다(https:// 없음.

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

3. 다음 JSON을 복사하여 REST API 클라이언트의 본문 필드에 붙여넣습니다.

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

4. YOUR_ENVIRONMENT_NAME을 ###으로 표시된 자리 표시자로 대체합니다.
5. 권한 필드에 키-값 쌍을 추가합니다. 예를 들어 Postman을 사용하는 경우 AWS 서명을 선택한 후 다음을 입력합니다.
   - AccessKey의 AWS_ACCESS_KEY_ID
   - SecretKey의 AWS_SECRET_ACCESS_KEY
6. 다음과 같은 응답이 표시되어야 합니다.

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

Python 스크립트 사용

다음 예제에서는 Python 스크립트의 boto3 create_web_login_token 메서드를 사용하여 Apache Airflow 웹 로그인 토큰을 생성합니다. 이 스크립트는 Amazon MWAA 외부에서 실행할 수 있습니다. boto3 라이브러리를 설치하기만 하면 됩니다. 가상 환경을 만들어 라이브러리를 설치할 수도 있습니다. 계정에 대한 AWS 인증 자격 증명을 구성했다고 가정합니다.

1. 다음 코드 샘플의 내용을 복사하여 로컬에 create-web-login-token.py로 저장합니다.

```
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. YOUR_ENVIRONMENT_NAME을 ###으로 표시된 자리 표시자로 대체합니다.
3. 다음 스크립트를 실행하여 웹 로그인 토큰을 얻습니다.

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

다음 단계

- 에서 웹 로그인 토큰을 생성하는 데 사용되는 Amazon MWAA API 작업을 살펴보십시오. CreateWebLoginToken

Apache Airflow CLI 토큰 생성

이 페이지의 명령을 사용하여 CLI 토큰을 생성한 명령 셸에서 직접 Amazon Managed Workflows for Apache Airflow API를 호출할 수 있습니다. 예를 들어, 토큰을 얻은 다음 Amazon MWAA API를 사용하여 프로그래밍 방식으로 DAG를 배포할 수 있습니다. 다음 섹션에는 AWS CLI, curl 스크립트, Python 스크립트 또는 bash 스크립트를 사용하여 Apache Airflow CLI 토큰을 생성하는 단계가 포함되어 있습니다. 응답에 반환되는 토큰은 60초 동안 유효합니다.

> **Note**
>
> AWS CLI 토큰은 비동기 API 명령이 아닌 동기 셸 작업을 대체하기 위한 것입니다. 따라서 사용 가능한 동시성은 제한됩니다. 웹 서버가 사용자에게 응답 상태를 유지하려면 이전 요청이 성공적으로 완료될 때까지 새 AWS CLI 요청을 열지 않는 것이 좋습니다.

목차

- 다음 단계

필수 조건

다음 섹션에서는 이 페이지의 명령과 스크립트를 사용하는 데 필요한 예비 단계를 설명합니다.

액세스

- 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer의 Amazon MWAA 권한 정책에 대한 AWS Identity and Access Management(IAM)의 AWS 계정 액세스
- 전체 API 및 콘솔 액세스 정책: FullApi AmazonMWAA 액세스의 Amazon MWAA 권한 정책에 대한 AWS Identity and Access Management(IAM)의 AWS 계정 액세스

AWS CLI

AWS Command Line Interface(AWS CLI)는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI - 버전 2 설치
- AWS CLI - `aws configure`을 통한 빠른 구성.

AWS CLI 사용

다음 예제에서는 AWS CLI에서 create-cli-token 명령을 사용하여 Apache Airflow CLI 토큰을 생성합니다.

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

curl 스크립트 사용

다음 예제에서는 curl 스크립트를 사용하여 AWS CLI에서 create-web-login-token 명령을 호출하여 Apache Airflow 웹 서버의 엔드포인트를 통해 Apache Airflow CLI를 호출합니다.

Apache Airflow v2

1. 텍스트 파일에서 curl 문을 복사하여 명령 셸에 붙여넣습니다.

> **Note**
>
> 이를 클립보드에 복사한 후에는 셸 메뉴에서 편집 > 붙여넣기를 사용해야 할 수 있습니다.

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

2. YOUR_REGION에 대한 자리 표시자를 사용자 환경, YOUR_DAG_NAME 및 YOUR_ENVIRONMENT_NAME에 대한 AWS 리전으로 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다(https://) 없음.

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

3. 명령 프롬프트에 다음이 표시되어야 합니다.

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

Apache Airflow v1

1. 텍스트 파일에서 cURL 문을 복사하여 명령 셸에 붙여넣습니다.

> **Note**
>
> 이를 클립보드에 복사한 후에는 셸 메뉴에서 편집 > 붙여넣기를 사용해야 할 수 있습니다.

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

2. YOUR_REGION에 대한 자리 표시자를 사용자 환경, YOUR_DAG_NAME 및 YOUR_HOST_NAME에 대한 AWS 리전으로 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다(https:// 없음.

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

3. 명령 프롬프트에 다음이 표시되어야 합니다.

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

4. YOUR_ENVIRONMENT_NAME 및 YOUR_DAG_NAME에 대한 자리 표시자를 대체합니다.

bash 스크립트 사용

다음 예제에서는 bash 스크립트를 사용하여 AWS CLI에서 create-cli-token 명령을 호출하여 Apache Airflow CLI 토큰을 생성합니다.

Apache Airflow v2

1. 다음 코드 샘플의 내용을 복사하고 로컬에서 `get-cli-token.sh`로 저장합니다.

```
# 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. ###에서 YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME 및 YOUR_DAG_NAME에 대한 자리 표시자를 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다 (https:// 없음.

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

3. (선택 사항) macOS 및 Linux 사용자는 스크립트가 실행 가능한지 확인하기 위해 다음 명령을 실행해야 할 수도 있습니다.

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

4. 다음 스크립트를 실행하여 Apache Airflow CLI 토큰을 생성합니다.

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

Apache Airflow v1

1. 다음 코드 샘플의 내용을 복사하고 로컬에서 `get-cli-token.sh`로 저장합니다.

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

2. ###에서 YOUR_ENVIRONMENT_NAME, YOUR_HOST_NAME 및 YOUR_DAG_NAME에 대한 자리 표시자를 대체합니다. 예를 들어, 퍼블릭 네트워크의 호스트 이름은 이와 같을 수 있습니다 (https://) 없음.

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

3. (선택 사항) macOS 및 Linux 사용자는 스크립트가 실행 가능한지 확인하기 위해 다음 명령을 실행해야 할 수도 있습니다.

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

4. 다음 스크립트를 실행하여 Apache Airflow CLI 토큰을 생성합니다.

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

Python 스크립트 사용

다음 예제에서는 Python 스크립트의 boto3 create_cli_token 메서드를 사용하여 Apache Airflow CLI 토큰을 생성하고 DAG를 트리거합니다. 이 스크립트는 Amazon MWAA 외부에서 실행할 수 있습니다. boto3 라이브러리를 설치하기만 하면 됩니다. 가상 환경을 만들어 라이브러리를 설치할 수도 있습니다. 계정에 대한 AWS 인증 보안 인증을 구성했다고 가정합니다.

Apache Airflow v2

1. 다음 코드 샘플의 내용을 복사하고 로컬에서 create-cli-token.py로 저장합니다.

```
"""
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. YOUR_ENVIRONMENT_NAME 및 YOUR_DAG_NAME에 대한 자리 표시자를 대체합니다.
3. 다음 스크립트를 실행하여 Apache Airflow CLI 토큰을 생성합니다.

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

Apache Airflow v1

1. 다음 코드 샘플의 내용을 복사하고 로컬에서 `create-cli-token.py`로 저장합니다.

```
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'trigger_dag'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

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

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

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

print(mwaa_response.status_code)
```

```
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. YOUR_ENVIRONMENT_NAME 및 YOUR_DAG_NAME에 대한 자리 표시자를 대체합니다.
3. 다음 스크립트를 실행하여 Apache Airflow CLI 토큰을 생성합니다.

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

다음 단계

- CreateClitoken에서 CLI 토큰을 생성하는 데 사용되는 Amazon MWAA API 작업을 살펴보십시오.

아파치 에어플로우 REST API 사용

아파치 에어플로우용 아마존 관리형 워크플로 (Amazon MWAA) 는 아파치 에어플로우 v2.4.3 이상을 실행하는 환경에서 아파치 에어플로우 REST API를 사용하여 아파치 에어플로우 환경과 직접 상호 작용할 수 있도록 지원합니다. 이를 통해 프로그래밍 방식으로 Amazon MWAA 환경에 액세스하고 관리할 수 있으며, 데이터 오케스트레이션 워크플로를 호출하고, DAG를 관리하고, 메타데이터 데이터베이스, 트리거, 스케줄러와 같은 다양한 Apache Airflow 구성 요소의 상태를 모니터링하는 표준화된 방법을 제공합니다.

Apache Airflow REST API를 직접 사용할 수 있도록 지원하기 위해 Amazon MWAA는 REST API 요청, 명령줄 인터페이스 (CLI) 사용 또는 더 많은 동시 Apache Airflow 사용자 인터페이스 (UI) 사용자 등 증가된 수요를 처리하기 위해 웹 서버 용량을 수평적으로 확장할 수 있는 옵션을 제공합니다. Amazon MWAA가 웹 서버를 확장하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “웹 서버 자동 스케일링 구성”

Apache Airflow REST API를 사용하여 환경에 다음과 같은 사용 사례를 구현할 수 있습니다.

- 프로그래밍 방식 액세스 — 이제 Apache Airflow UI 또는 CLI를 사용하지 않고도 Apache Airflow DAG 실행을 시작하고, 데이터 세트를 관리하고, 메타데이터 데이터베이스, 트리거 및 스케줄러와 같은 다양한 구성 요소의 상태를 검색할 수 있습니다.
- 외부 애플리케이션 및 마이크로서비스와의 통합 — REST API 지원을 통해 Amazon MWAA 환경을 다른 시스템과 통합하는 사용자 지정 솔루션을 구축할 수 있습니다. 예를 들어, 완료된 데이터베이스 작업이나 신규 사용자 등록과 같은 외부 시스템의 이벤트에 대한 응답으로 워크플로를 시작할 수 있습니다.

- 중앙 모니터링 — 여러 Amazon MWAA 환경에서 DAG의 상태를 집계하는 모니터링 대시보드를 구축하여 중앙 집중식 모니터링 및 관리를 가능하게 할 수 있습니다.

다음 주제에서는 웹 서버 액세스 토큰을 획득한 다음 해당 토큰을 사용하여 Apache Airflow REST API에 API를 호출하는 방법을 보여줍니다. 다음 예제에서는 API를 호출하여 새 DAG 실행을 시작합니다.

아파치 에어플로우 REST API에 대한 자세한 내용은 아파치 에어플로우 REST API 레퍼런스를 참조하십시오.

주제

웹 서버 세션 토큰 생성

웹 서버 액세스 토큰을 만들려면 다음 Python 함수를 사용합니다. 이 함수는 먼저 Amazon MWAA API를 호출하여 웹 로그인 토큰을 얻습니다. 60초 후에 만료되는 웹 로그인 토큰은 웹 세션 토큰으로 교환되며, 이를 통해 웹 서버에 액세스하여 Apache Airflow REST API를 사용할 수 있습니다.

> **Note**
>
> 세션 토큰은 12시간 후에 만료됩니다.

```
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
```

아파치 에어플로우 REST API를 호출하십시오.

인증이 완료되면 API 엔드포인트로 요청을 보내기 시작할 수 있는 자격 증명이 생깁니다. 아래 예시에서는 엔드포인트를 /dags/*dag_id*/dag 사용합니다.

```
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)
```

Apache Airflow CLI 명령 참조

이 페이지에서는 Amazon Managed Workflows for Apache Airflow에서 지원되는 Apache Airflow CLI 명령과 지원되지 않는 Apache Airflow CLI 명령을 설명합니다.

목차

필수 조건

다음 섹션에서는 이 페이지의 명령과 스크립트를 사용하는 데 필요한 예비 단계를 설명합니다.

액세스

- AWS 에서 Amazon MWAA 권한 정책에 대한 계정 액세스 AWS Identity and Access Management (IAM) 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer

- AWS Amazon MWAA 권한 정책에 대한 AWS Identity and Access Management (IAM) 계정 액세스 전체 API 및 콘솔 액세스 정책: FullApi AmazonMWAA 액세스

AWS CLI

AWS Command Line Interface (AWS CLI) 는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있게 해주는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI — 버전 2를 설치합니다.
- AWS CLI — 빠른 구성 `aws configure`.

v2에서 변경된 사항

- 신규: Airflow CLI 명령 구조. Apache Airflow v2 CLI는 관련 명령이 하위 명령으로 그룹화되도록 구성되어 있습니다. 이는 Apache Airflow v2로 업그레이드하려면 Apache Airflow v1 스크립트를 업데이트해야 한다는 의미입니다. 예를 들어, Apache Airflow v1에서 `unpause`이 이제 Apache Airflow v2에 있습니다. 자세한 내용을 알아보려면 Apache Airflow 참조 가이드에서 섹션 2의 Airflow CLI 변경 사항을 참조하십시오.

지원되는 CLI 명령

다음 섹션에는 Amazon MWAA에서 사용할 수 있는 Apache Airflow CLI 명령이 나와 있습니다.

지원되는 명령

Apache Airflow v2

마이너 버전	Command
v2.0+	cheat-sheet
v2.0+	connections add
v2.0+	connections delete
v2.2+(참고)	dags backfill
v2.0+	dags delete

마이너 버전	Command
v2.2+(참고)	dags list
v2.0+	dags list-jobs
v2.6+	5 일 list-import-errors
v2.2+(참고)	dags list-runs
v2.2+(참고)	dags next-execution
v2.0+	dags pause
v2.0+	dags report
v2.4+	dags reserialize
v2.0+	dags show
v2.0+	dags state
v2.0+	dags test
v2.0+	dags trigger
v2.0+	dags unpause
v2.4+	db clean
v2.0+	providers behaviours
v2.0+	providers get
v2.0+	providers hooks
v2.0+	providers links
v2.0+	providers list
v2.8+	제공업체 알림

마이너 버전	Command
v2.6+	providers secrets
v2.7+	providers triggerer
v2.0+	providers widgets
v2.6+	roles add-perms
v2.6+	roles del-perms
v2.6+	역할 생성
v2.0+	roles list
v2.0+	tasks clear
v2.0+	tasks failed-deps
v2.0+	tasks list
v2.0+	tasks render
v2.0+	tasks state
v2.0+	태스크 states-for-dag-run
v2.0+	tasks test
v2.0+	variables delete
v2.0+	variables get
v2.0+	variables set
v2.0+	variables list
v2.0+	version

DAG를 구문 분석하는 명령 사용

환경에서 Apache Airflow v1.10.12 또는 v2.0.2를 실행하는 경우 DAG가 다음을 통해 설치된 패키지에 의존하는 플러그인을 사용하는 경우 DAG를 구문 분석하는 CLI 명령이 실패합니다. `requirements.txt`

Apache Airflow v2.0.2

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

DAG가 `requirements.txt`를 통해 설치된 패키지에 의존하는 플러그인을 사용하지 않으면 이러한 CLI 명령을 사용할 수 있습니다.

샘플 코드

다음 섹션에는 Apache Airflow CLI를 사용하는 여러 방법의 예제가 나와 있습니다.

Apache Airflow v2 변수 설정, 가져오기 또는 삭제

다음 샘플 코드를 사용하여 `<script> <mwaa env name> get | set | delete <variable> <variable value> </variable> </variable>`의 형식으로 변수를 설정, 가져오기 또는 삭제할 수 있습니다.

```
[ $# -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
```

DAG를 트리거할 때 구성 추가

Apache Airflow v1 및 Apache Airflow v2에서 다음 샘플 코드를 사용하여 DAG를 트리거할 때 `airflow trigger_dag 'dag_name' –conf '{"key":"value"}'`과 같이 구성을 추가할 수 있습니다.

```
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)
```

Bastion Host에 대한 SSH 터널에서 CLI 명령 실행

다음 예제는 Linux Bastion Host에 대한 SSH 터널 프록시를 사용하여 Airflow CLI 명령을 실행하는 방법을 보여줍니다.

curl 사용

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
   ```

GitHub 샘플 AWS 및 튜토리얼

- Amazon Managed Workflows for Apache Airflow에서 Apache Airflow v2.0.2 파라미터 및 변수 작업
- 명령줄을 통해 Amazon MWAA에서 Apache Airflow v1.10.12와 상호 작용
- Amazon MWAA에서 아파치 에어플로우 v1.10.12를 사용하고 Bash Operator를 사용하는 대화형 명령 GitHub

Apache Airflow에 대한 연결 관리

이 섹션에서는 Amazon Managed Workflows for Apache Airflow 환경을 위해 Apache Airflow 연결을 구성하는 다양한 방법을 설명합니다.

주제

- Apache Airflow 변수 및 연결 개요
- Amazon MWAA 환경에 설치된 Apache Airflow 공급자 패키지
- 연결 유형 개요
- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager

Apache Airflow 변수 및 연결 개요

경우에 따라 AWS 프로필과 같은 환경에 대한 추가 연결 또는 변수를 지정하거나 Apache Airflow 메타스토어의 연결 객체에 실행 역할을 추가한 다음 DAG 내에서 연결을 참조할 수 있습니다.

- 자체 관리형 Apache Airflow 자체 관리형 Apache Airflow 설치에서는 `airflow.cfg`에서 Apache Airflow 구성 옵션을 설정합니다.

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

- Amazon MWAA의 Apache Airflow. Amazon MWAA에서는 이러한 구성 설정을 Amazon MWAA 콘솔에 Apache Airflow 구성 옵션으로 추가해야 합니다. Apache Airflow 구성 옵션은 사용자 환경의 환경 변수로 작성되며 동일한 설정에 대한 다른 모든 기존 구성을 재정의합니다.

Amazon MWAA 환경에 설치된 Apache Airflow 공급자 패키지

Amazon MWAA는 새 환경을 만들 때 Apache Airflow v2 이상의 연결 유형에 대한 추가 공급자를 설치합니다. 공급자 패키지를 설치하면 Apache Airflow UI에서 연결 유형을 볼 수 있습니다. 또한 `requirements.txt` 파일에서 이러한 패키지를 Python 종속성으로 지정할 필요가 없습니다. 이 페이지에는 모든 Apache Airflow v2 환경에 대해 Amazon MWAA에서 설치한 Apache Airflow 공급자 패키지가 나열되어 있습니다.

> **Note**
>
> Apache Airflow v2 이상의 경우 Amazon MWAA는 실행 후 Watchtower 버전 2.0.1을 설치하여 다른 Python 라이브러리 CloudWatch 설치가 로깅과의 호환성을 재정의하지 않도록 합니다. `pip3 install -r requirements.txt`

목차

아파치 에어플로우 v2.8.1 연결용 공급자 패키지

아파치 에어플로우 v2.8.1에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 아파치 에어플로우 연결에 사용되는 다음과 같은 공급자 패키지를 설치합니다.

> **Note**
>
> 지원되는 최신 버전을 `apache-airflow-providers-amazon`을(를) 지정하여 이 공급자를 업그레이드할 수 있습니다. AMI 버전 지정에 대한 자세한 내용은 the section called “새 제공자 패키지 지정” 단원을 참조하십시오.

연결 유형	패키지
AWS 연결	apache-airflow-providers-amazon[아이오보토코어] ==8.16.0

연결 유형	패키지
Postgres 연결	apache-airflow-providers-postgres==5.10.0
FTP 연결	apache-airflow-providers-ftp==3.7.0
Celery 연결	apache-airflow-providers-celery==3.5.1
HTTP 연결	apache-airflow-providers-http==4.8.0
IMAP 연결	apache-airflow-providers-imap==3.5.0
공통 SQL	apache-airflow-providers-common-sql==1.10.0
SQLite 연결	apache-airflow-providers-sqlite==3.7.0

Apache Airflow v2.7.2 연결용 공급자 패키지

Apache Airflow v2.7.2에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

> **Note**
>
> 지원되는 최신 버전을 `apache-airflow-providers-amazon`을(를) 지정하여 이 공급자를 업그레이드할 수 있습니다. AMI 버전 지정에 대한 자세한 내용은 the section called “새 제공자 패키지 지정” 단원을 참조하십시오.

연결 유형	패키지
AWS 커넥션	apache-airflow-providers-amazon[아이오보토코어] ==8.7.1
Postgres 연결	apache-airflow-providers-postgres==5.6.1
FTP 연결	apache-airflow-providers-ftp==3.5.2
Celery 연결	apache-airflow-providers-celery==3.3.4

연결 유형	패키지
HTTP 연결	apache-airflow-providers-http==4.5.2
IMAP 연결	apache-airflow-providers-imap==3.3.2
공통 SQL	apache-airflow-providers-common-sql==1.7.2
SQLite 연결	apache-airflow-providers-sqlite==3.4.3

Apache Airflow v2.6.3 연결을 위한 공급자 패키지

Apache Airflow v2.6.3에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

> **Note**
>
> 지원되는 최신 버전을 `apache-airflow-providers-amazon`을(를) 지정하여 이 공급자를 업그레이드할 수 있습니다. AMI 버전 지정에 대한 자세한 내용은 the section called “새 제공자 패키지 지정” 단원을 참조하십시오.

연결 유형	패키지
AWS 커넥션	apache-airflow-providers-amazon[아이오보토코어] ==8.2.0
Postgres 연결	apache-airflow-providers-postgres==5.5.1
FTP 연결	apache-airflow-providers-ftp==3.4.2
Celery 연결	apache-airflow-providers-celery==3.2.1
HTTP 연결	apache-airflow-providers-http==4.4.2
IMAP 연결	apache-airflow-providers-imap==3.2.2
공통 SQL	apache-airflow-providers-common-sql==1.5.2

연결 유형	패키지
SQLite 연결	apache-airflow-providers-sqlite==3.4.2

Apache Airflow v2.5.1 연결을 위한 공급자 패키지

Apache Airflow v2.5.1에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

> **Note**
>
> 지원되는 최신 버전을 `apache-airflow-providers-amazon`을(를) 지정하여 이 공급자를 업그레이드할 수 있습니다. AMI 버전 지정에 대한 자세한 내용은 the section called “새 제공자 패키지 지정” 단원을 참조하십시오.

연결 유형	패키지
AWS 커넥션	apache-airflow-providers-amazon==7.1.0
Postgres 연결	apache-airflow-providers-postgres==5.4.0
FTP 연결	apache-airflow-providers-ftp==3.3.0
Celery 연결	apache-airflow-providers-celery==3.1.0
HTTP 연결	apache-airflow-providers-http==4.1.1
IMAP 연결	apache-airflow-providers-imap==3.1.1
공통 SQL	apache-airflow-providers-common-sql==1.3.3
SQLite 연결	apache-airflow-providers-sqlite==3.3.1

Apache Airflow v2.4.3 연결을 위한 공급자 패키지

Apache Airflow v2.4.3에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

연결 유형	패키지
AWS 커넥션	apache-airflow-providers-amazon==6.0.0
Postgres 연결	apache-airflow-providers-postgres==5.2.2
FTP 연결	apache-airflow-providers-ftp==3.1.0
Celery 연결	apache-airflow-providers-celery==3.0.0
HTTP 연결	apache-airflow-providers-http==4.0.0
IMAP 연결	apache-airflow-providers-imap==3.0.0
공통 SQL	apache-airflow-providers-common-sql==1.2.0
SQLite 연결	apache-airflow-providers-sqlite==3.2.1

Apache Airflow v2.2.2 연결을 위한 공급자 패키지

Apache Airflow v2.2.2에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

연결 유형	패키지
AWS 커넥션	apache-airflow-providers-amazon==2.4.0
Postgres 연결	apache-airflow-providers-postgres==2.3.0
FTP 연결	apache-airflow-providers-ftp==2.0.1
Celery 연결	apache-airflow-providers-celery==2.1.0
HTTP 연결	apache-airflow-providers-http==2.0.1
IMAP 연결	apache-airflow-providers-imap==2.0.1
SQLite 연결	apache-airflow-providers-sqlite==2.0.1

Apache Airflow v2.0.2 연결용 공급자 패키지

Apache Airflow v2.0.2에서 Amazon MWAA 환경을 생성할 때 Amazon MWAA는 Apache Airflow 연결에 사용되는 공급자 패키지를 다음과 같이 설치합니다.

연결 유형	패키지
Tableau 연결	apache-airflow-providers-tableau==1.0.0
Databricks 연결	apache-airflow-providers-databricks==1.0.1
SSH 연결	apache-airflow-providers-ssh==1.3.0
Postgres 연결	apache-airflow-providers-postgres==1.0.2
Docker 연결	apache-airflow-providers-docker==1.2.0
Oracle 연결	apache-airflow-providers-oracle==1.1.0
Presto 연결	apache-airflow-providers-presto==1.0.2
SFTP 연결	apache-airflow-providers-sftp==1.2.0

새 제공자 패키지 지정

Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.

Apache Airflow 제약 조건 파일은 Apache Airflow 릴리스 당시 사용할 수 있는 공급자 버전을 지정합니다. 그러나 대부분의 경우 최신 공급자가 해당 버전의 Apache Airflow와 호환됩니다. 제약 조건을 사용해야 하므로 새 버전의 공급자 패키지를 지정하려면 특정 공급자 버전에 대한 제약 조건 파일을 수정할 수 있습니다.

1. https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt에서 버전별 제약 조건 파일을 다운로드합니다.
2. 제약 조건 파일의 `apache-airflow-providers-amazon` 버전을 사용하려는 버전으로 수정합니다.

3. 수정된 제약 조건 파일을 Amazon MWAA 환경의 Amazon S3 dags 폴더에 `constraints-3.11-updated.txt`와(과) 같이 저장합니다.
4. 다음과 같이 요구 사항을 지정하시기 바랍니다.

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

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

> **Note**
>
> 프라이빗 웹 서버를 사용하는 경우 Amazon MWAA 로컬 러너를 사용하여 필수 라이브러리를 WHL 파일로 패키징하는 것이 좋습니다.

연결 유형 개요

Apache Airflow는 연결을 연결 URI 문자열로 저장합니다. 연결 유형에 관계없이 연결 URI 문자열을 생성할 수 있는 연결 템플릿을 Apache Airflow UI에 제공합니다. Apache Airflow UI에서 연결 템플릿을 사용할 수 없는 경우 대체 연결 템플릿을 사용하여 이 연결 URI 문자열을 생성할 수 있습니다(예: HTTP 연결 템플릿 사용). 가장 큰 차이점은 URI 접두사(예: `my-conn-type://`)이며, Apache Airflow 제공자는 일반적으로 연결 시 무시합니다. 이 페이지에서는 여러 연결 유형에 대해 Apache Airflow UI의 연결 템플릿을 서로 바꿔서 사용하는 방법을 설명합니다.

> **Warning**
>
> Amazon MWAA에서 `aws_default` 연결을 덮어쓰지 마십시오. Amazon MWAA는 이 연결을 사용하여 작업 로그 수집과 같은 다양한 중요 작업을 수행합니다. 이 연결을 덮어쓰면 데이터가 손실되고 환경 가용성이 중단될 수 있습니다.

주제

연결 URI 문자열 예제

다음 예제에서는 MySQL 연결 유형에 대한 연결 URI 문자열을 보여줍니다.

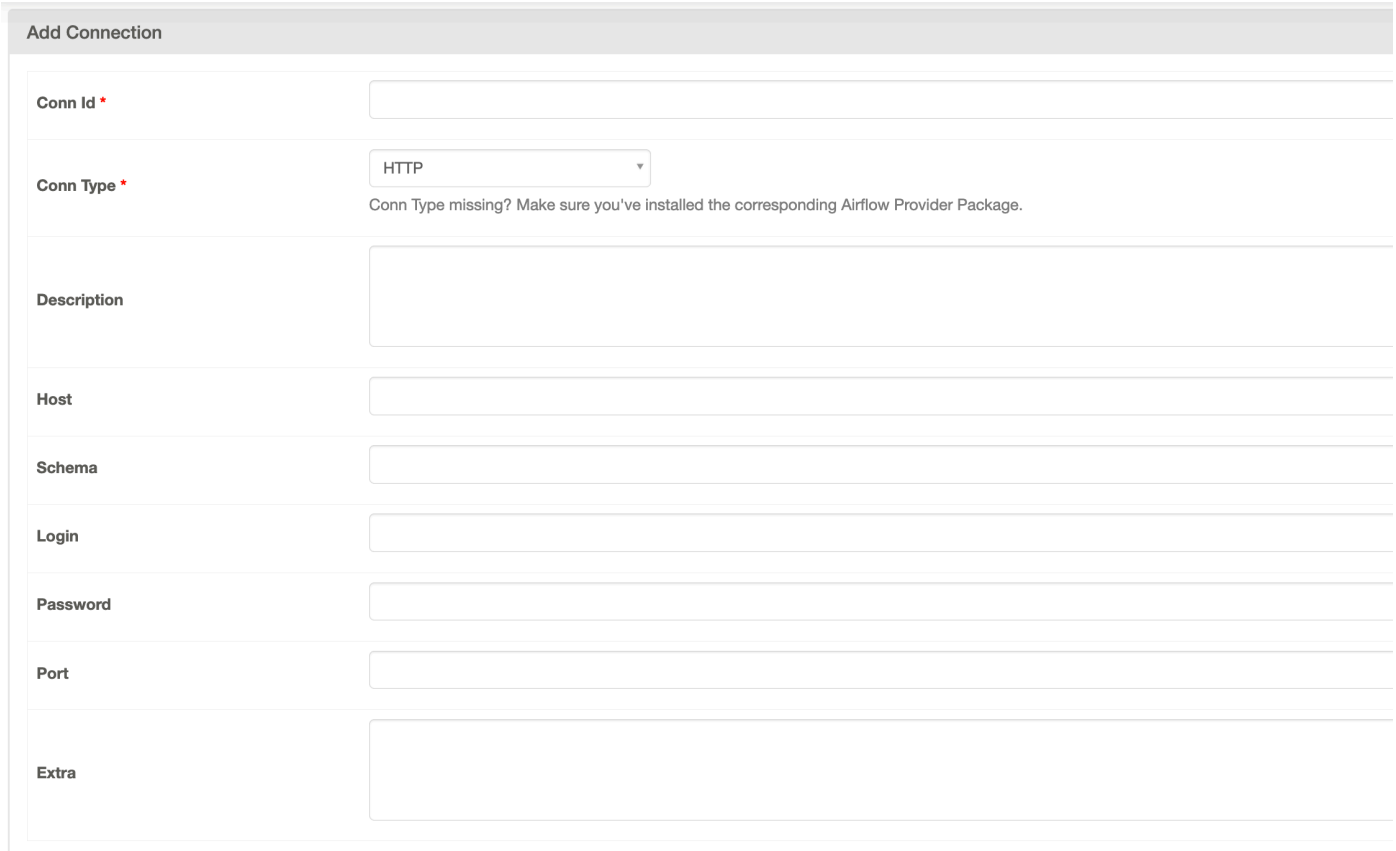
```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role%2FAmazonMWAA-
MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

연결 템플릿 예제

다음 예제에서는 Apache Airflow UI에 있는 HTTP 연결 템플릿을 보여줍니다.

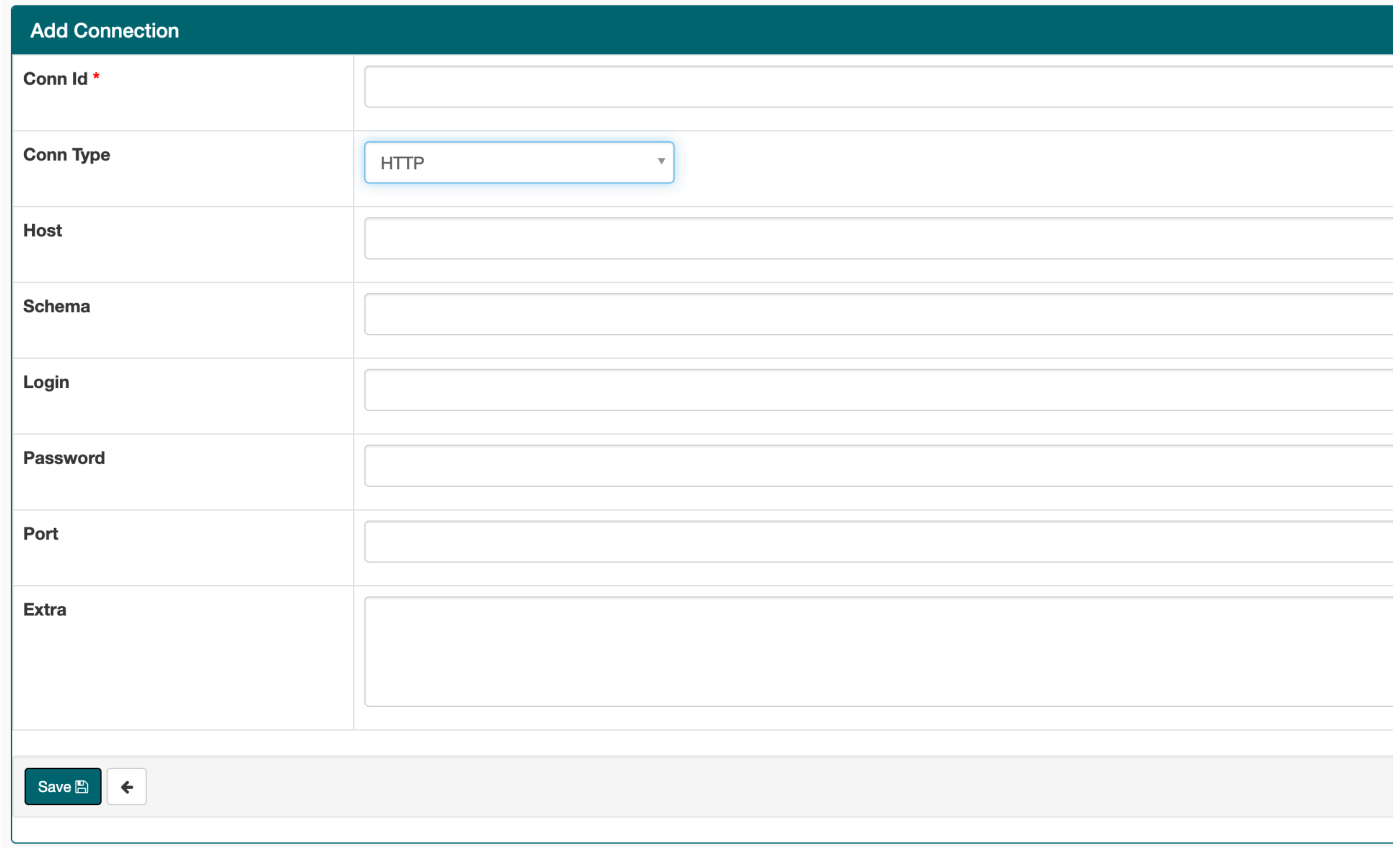
Apache Airflow v2

다음 예제는 Apache Airflow UI의 Apache Airflow v2에 대한 HTTP 연결 템플릿을 보여줍니다.



Apache Airflow v1

다음 예제는 Apache Airflow UI의 Apache Airflow v1에 대한 HTTP 연결 템플릿을 보여줍니다.



Jdbc 연결에 HTTP 연결 템플릿을 사용하는 예

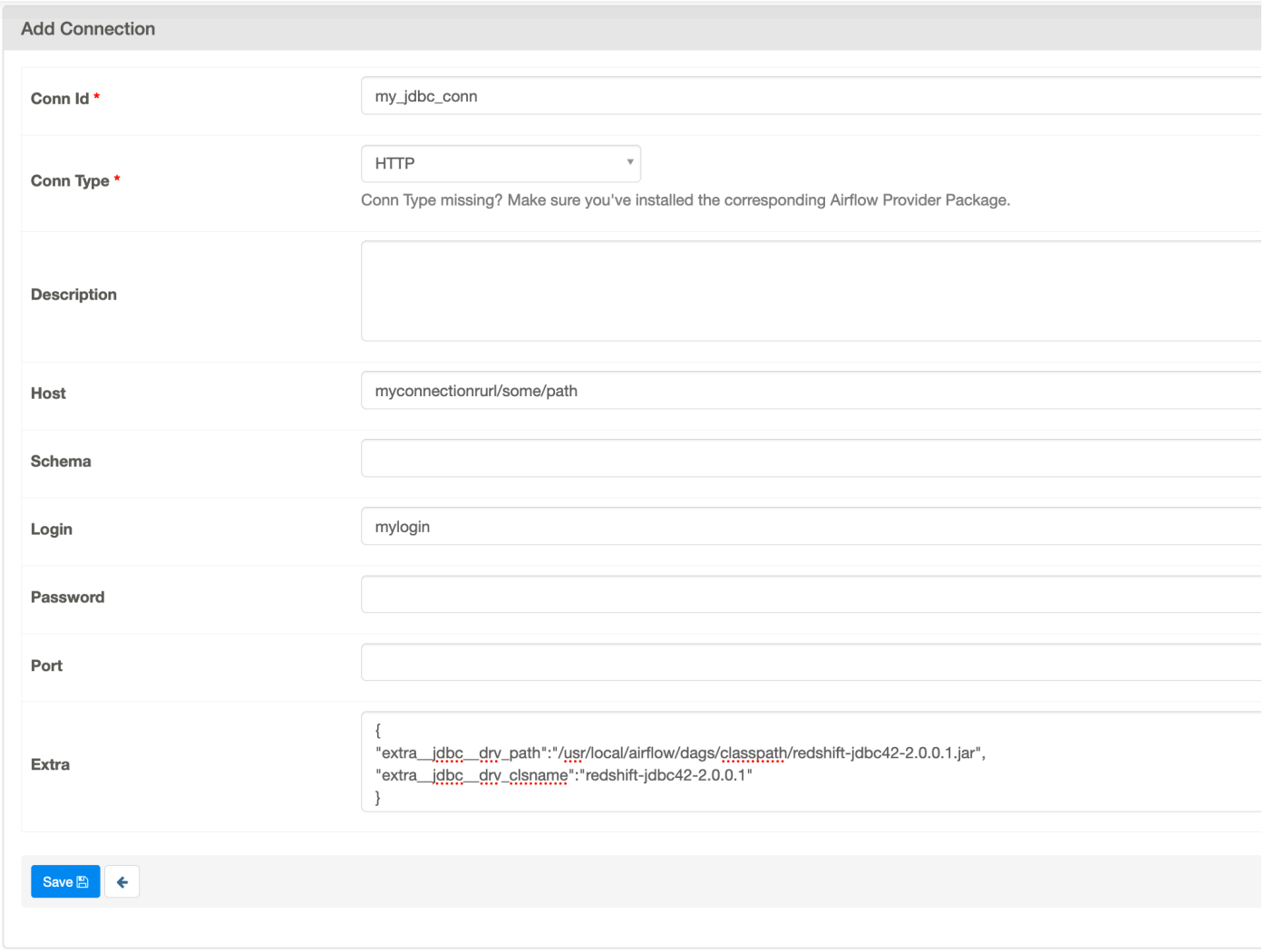
다음 예제는 Apache Airflow v2.0.2의 Jdbc 연결 유형에 HTTP 연결 템플릿을 사용하고 Apache Airflow UI의 Apache Airflow v1.10.12에 대한 Jdbc 연결 템플릿에서 동일한 값을 사용하는 방법을 보여줍니다.

Apache Airflow v2

다음 예제는 이 섹션의 예제에 대해 Apache Airflow에서 생성한 연결 URI 문자열을 보여줍니다.

```
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
```

다음 예제는 Apache Airflow UI의 Apache Airflow v2에 대한 Jdbc 연결에 HTTP 연결 템플릿을 사용하는 방법을 보여줍니다.

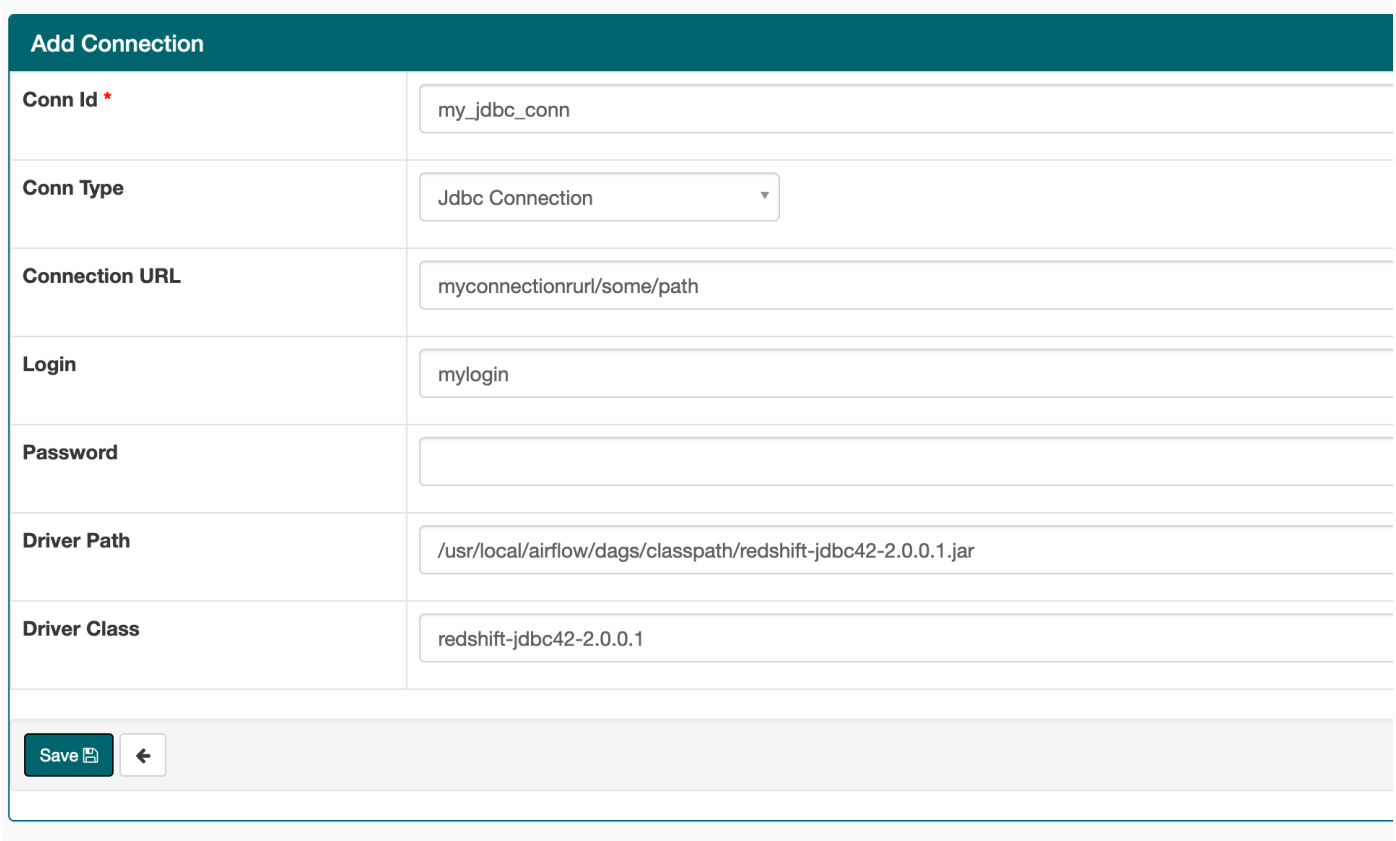


Apache Airflow v1

다음 예제는 이 섹션의 예제에 대해 Apache Airflow에서 생성한 연결 URI 문자열을 보여줍니다.

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

다음 예제는 Apache Airflow UI의 Apache Airflow v1.10.12에 대한 Jdbc 연결 템플릿을 보여줍니다.



시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager

AWS Secrets Manager Apache Airflow용 Amazon 관리형 워크플로우 환경에서 지원되는 대체 아파치 에어플로우 백엔드입니다. 이 가이드에서는 Apache Airflow용 Amazon Managed Workflows에 Apache Airflow 변수 및 Apache Airflow 연결에 대한 암호를 안전하게 저장하는 데 사용하는 AWS Secrets Manager 방법을 보여줍니다.

> **Note**
>
> - 생성한 암호에 대해 요금이 부과됩니다. Secrets Manager 요금에 대한 자세한 내용은 AWS 요금을 참조하십시오.

목차

1단계: Amazon MWAA에 Secrets Manager 암호 키에 액세스할 수 있는 권한을 제공합니다.

Amazon MWAA 환경의 실행 역할에는 AWS Secrets Manager의 암호 키를 읽을 수 있는 권한이 필요합니다. 다음 IAM 정책은 관리형 정책을 사용한 읽기-쓰기 액세스를 허용합니다. AWS SecretsManagerReadWrite

실행 역할에 정책을 연결하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 권한 창에서 실행 역할을 선택합니다.
4. 정책 연결을 선택합니다.
5. 정책 필터링 텍스트 필드에 `SecretsManagerReadWrite`을(를) 입력합니다.
6. 정책 연결을 선택합니다.

AWS 관리형 권한 정책을 사용하지 않으려면 환경의 실행 역할을 직접 업데이트하여 Secrets Manager 리소스에 대한 모든 수준의 액세스를 허용할 수 있습니다. 예를 들어 다음 정책 설명은 Secrets Manager에서 특정 AWS 지역에 생성한 모든 비밀에 대한 읽기 액세스 권한을 부여합니다.

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

```
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:012345678910:secret:*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        }
    ]
}
```

2단계: Secrets Manager 백엔드를 Apache Airflow 구성 옵션으로 생성

다음 섹션에서는 Amazon MWAA 콘솔에서 백엔드용 Apache Airflow 구성 옵션을 생성하는 방법을 설명합니다. AWS Secrets Manager `airflow.cfg`에서 동일한 이름의 구성 설정을 사용하는 경우 다음 단계에서 생성하는 구성이 우선하며 구성 설정을 재정의합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. 다음을 선택합니다.
5. Airflow 구성 옵션 창에서 사용자 지정 구성 추가를 선택합니다. 다음 키-값 쌍을 추가합니다:

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

    b. **`secrets.backend_kwargs`**: **`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`** 이렇게 하면 Apache Airflow가 `airflow/connections/*` 및 `airflow/variables/*` 경로에서 연결 문자열과 변수를 찾도록 구성됩니다.

    조회 패턴을 사용하면 Amazon MWAA가 사용자를 대신하여 Secrets Manager에 보내는 API 호출 횟수를 줄일 수 있습니다. 조회 패턴을 지정하지 않으면 Apache Airflow는 구성된 백엔드에서 모든 연결과 변수를 검색합니다. 패턴을 지정하면 Apache Airflow가 찾는 가능한 경로를 좁힐 수 있습니다. 따라서 Secrets Manager를 Amazon MWAA와 함께 사용할 때 비용이 절감됩니다.

조회 패턴을 지정하려면 connections_lookup_pattern 및 variables_lookup_pattern 파라미터를 지정합니다. 이러한 파라미터는 문자열을 입력으로 받아들입니다. RegEx 예를 들어 test(으)로 시작하는 암호를 찾으려면 secrets.backend_kwargs에 다음을 입력합니다.

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

> **Note**
>
> connections_lookup_pattern 및 variables_lookup_pattern을(를) 사용하려면 apache-airflow-providers-amazon 버전 7.3.0 이상을 설치해야 합니다. Provider 패키지를 최신 버전으로 업데이트하는 방법에 대한 자세한 내용은 the section called “새 제공자 패키지 지정” 단원을 참조하십시오.

6. 저장을 선택합니다.

3단계: 아파치 에어플로우 AWS 연결 URI 문자열 생성

연결 문자열을 만들려면 키보드의 “탭” 키를 사용하여 Connection 개체의 키-값 쌍을 들여쓰십시오. 또한 쉘 세션에서 extra 객체에 대한 변수를 만드는 것이 좋습니다. 다음 섹션에서는 Apache Airflow 또는 Python 스크립트를 사용하여 Amazon MWAA 환경을 위한 Apache Airflow 연결 URI 문자열을 생성하는 단계를 안내합니다.

Apache Airflow CLI

다음 쉘 세션은 로컬 Airflow CLI를 사용하여 연결 문자열을 생성합니다. CLI가 설치되어 있지 않은 경우 Python 스크립트를 사용하는 것이 좋습니다.

1. Python 쉘 세션을 엽니다.

```
python3
```

2. 다음 명령을 입력합니다.

```
>>> import json
```

3. 다음 명령을 입력합니다.

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

4. 쉘 세션에서 `extra` 객체에 대한 변수를 생성합니다. *YOUR_EXECUTION_ROLE_ARN*의 샘플 값을 실행 역할 ARN으로 대체하고 *YOUR_REGION*의 리전을 `us-east-1`처럼 대체합니다.

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

5. 연결 객체를 생성합니다. `myconn`의 샘플 값을 Apache Airflow 연결 이름으로 대체합니다.

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

6. 키보드의 '탭' 키를 사용하여 연결 객체에 다음 키-값 쌍을 각각 들여쓰기할 수 있습니다. 샘플 값을 *###*으로 대체합니다.

   a. 연결 유형을 지정하십시오. AWS

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

   b. Apache Airflow 데이터베이스 옵션을 지정합니다.

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

   c. Amazon MWAA의 Apache Airflow UI URL을 지정합니다.

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

   d. Amazon AWS MWAA에 로그인하기 위한 액세스 키 ID (사용자 이름) 를 지정합니다.

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

   e. Amazon MWAA에 로그인하기 위한 AWS 보안 액세스 키 (암호) 를 지정합니다.

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

f. `extra` 쉘 세션 변수를 지정합니다.

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

g. 연결 객체를 종료합니다.

```
... )
```

7. 연결 URI 문자열을 인쇄합니다.

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

응답에 연결 URI 문자열이 표시되어야 합니다.

```
'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

다음 Python 스크립트에는 Apache Airflow CLI가 필요하지 않습니다.

1. 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 `mwaa_connection.py`로 저장합니다.

```
import urllib.parse

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

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

2. 자리 표시자를 ###으로 대체합니다.
3. 다음 스크립트를 실행하여 연결 문자열을 생성합니다.

```
python3 mwaa_connection.py
```

4단계: Secrets Manager에서 변수 추가

다음 섹션에서는 Secrets Manager에서 변수에 대한 암호를 생성하는 방법을 설명합니다.

암호를 생성하려면

1. AWS Secrets Manager 콘솔을 엽니다.
2. 새 암호 저장을 선택합니다.
3. 다른 유형의 암호를 선택합니다.
4. 이 암호에 저장할 키/값 쌍 지정 창에서 일반 텍스트를 선택합니다.
5. 변수 값을 다음 형식의 일반 텍스트로 추가합니다.

```
"YOUR_VARIABLE_VALUE"
```

예를 들어, 정수를 지정하려면:

```
14
```

예를 들어 문자열을 지정하려면:

```
"mystring"
```

6. 암호화 키의 경우 드롭다운 목록에서 AWS KMS 키 옵션을 선택합니다.
7. 암호 이름의 텍스트 필드에 다음 형식으로 이름을 입력합니다.

```
airflow/variables/YOUR_VARIABLE_NAME
```

예:

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

8. 다음을 선택합니다.
9. 암호 구성 페이지의 암호 이름 및 설명 창에서 다음을 수행합니다.

    a. 암호 이름에 암호 이름을 입력합니다.
    b. (선택 사항) 설명에 암호에 대한 설명을 입력합니다.

    다음을 선택합니다.
10. 순환 구성 - 옵션에서 기본 옵션을 그대로 두고 다음을 선택합니다.
11. Secrets Manager에서 이 단계를 반복하여 원하는 만큼 변수를 추가합니다.
12. 검토 페이지에서 암호를 검토한 후 저장을 선택합니다.

5단계: Secrets Manager에서 연결 추가

다음 섹션에서는 Secrets Manager에서 연결 문자열 URI에 대한 암호를 생성하는 방법을 설명합니다.

암호를 생성하려면

1. AWS Secrets Manager 콘솔을 엽니다.
2. 새 암호 저장을 선택합니다.
3. 다른 유형의 암호를 선택합니다.
4. 이 암호에 저장할 키/값 쌍 지정 창에서 일반 텍스트를 선택합니다.
5. 연결 URI 문자열을 다음 형식의 일반 텍스트로 추가합니다.

```
YOUR_CONNECTION_URI_STRING
```

예:

```
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 Airflow는 연결 문자열의 각 값을 구문 분석합니다. 작은따옴표나 큰따옴표를 사용해서는 안 됩니다. 그렇지 않으면 연결을 단일 문자열로 구문 분석합니다.

6. 암호화 키의 경우 드롭다운 AWS KMS 목록에서 키 옵션을 선택합니다.
7. 암호 이름의 텍스트 필드에 다음 형식으로 이름을 입력합니다.

   ```
   airflow/connections/YOUR_CONNECTION_NAME
   ```

   예:

   ```
   airflow/connections/myconn
   ```

8. 다음을 선택합니다.
9. 암호 구성 페이지의 암호 이름 및 설명 창에서 다음을 수행합니다.

   a. 암호 이름에 암호 이름을 입력합니다.

   b. (선택 사항) 설명에 암호에 대한 설명을 입력합니다.

   다음을 선택합니다.
10. 순환 구성 - 옵션에서 기본 옵션을 그대로 두고 다음을 선택합니다.
11. Secrets Manager에서 이 단계를 반복하여 원하는 만큼 변수를 추가합니다.
12. 검토 페이지에서 암호를 검토한 후 저장을 선택합니다.

샘플 코드

- Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용의 샘플 코드를 사용하여 이 페이지의 Apache Airflow 연결(`myconn`)에 암호 키를 사용하는 방법을 알아봅니다.
- `test-variable`의 샘플 코드를 사용하여 이 페이지의 Apache Airflow 변수(Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용)에 암호 키를 사용하는 방법을 알아봅니다.

리소스

- 콘솔 및 를 사용하여 Secrets Manager 암호를 구성하는 방법에 대한 자세한 내용은 사용 AWS Secrets Manager 설명서에서 암호 만들기를 참조하십시오. AWS CLI
- Apache Airflow 연결 및 변수를 AWS Secrets Manager으로 이동에서 Python 스크립트를 사용하여 대량의 Apache Airflow 변수 및 연결을 Secrets Manager로 마이그레이션합니다.

다음 단계

- 아파치 에어플로우 액세스에서 Apache Airflow UI에 액세스하기 위한 토큰을 생성하는 방법을 알아봅니다.

Amazon MWAA 환경 관리

Amazon Managed Workflows for Apache Airflow 콘솔은 Apache Airflow UI에 대한 프라이빗 또는 퍼블릭 액세스를 구성하는 내장 옵션이 포함되어 있습니다. 또한 환경 규모, 작업자 규모 조정 시기 등을 구성하는 기본 제공 옵션과 일반적으로 `airflow.cfg`에서만 액세스할 수 있는 Apache Airflow 구성을 재정의할 수 있는 Apache Airflow 구성 옵션도 포함되어 있습니다. 이 가이드에서는 Amazon MWAA 콘솔에서 이러한 구성을 사용하는 방법을 설명합니다.

주제

Amazon MWAA 환경 클래스 구성

Amazon MWAA 환경에 맞게 선택한 환경 클래스에 따라 셀러리 실행기가 실행되는 AWS관리형 AWS Fargate 컨테이너와 Apache Airflow 스케줄러가 작업 인스턴스를 생성하는 관리형 Amazon AWS Aurora PostgreSQL 메타데이터 데이터베이스의 크기가 결정됩니다. 이 페이지에서는 각 Amazon MWAA 환경 클래스와 Amazon MWAA 콘솔에서 환경 클래스를 업데이트하는 단계를 설명합니다.

Sections

환경 기능

다음 섹션에는 각 환경 클래스에 대한 기본 동시 Apache Airflow 작업, RAM(랜덤 액세스 메모리) 및 vCPU(가상 중앙 처리 장치)가 포함되어 있습니다. 나열된 동시 작업은 작업 동시성이 해당 환경의 Apache Airflow Worker 용량을 초과하지 않는다고 가정합니다.

다음 표에서 DAG 용량은 실행이 아닌 DAG 정의를 의미하며 DAG가 단일 Python 파일에서 동적이고 Apache Airflow 모범 사례로 작성되었다고 가정합니다.

작업 실행은 동시에 예약된 작업 수에 따라 달라지며, 동시에 시작되도록 설정된 DAG 실행 수가 기본값 max_dagruns_per_loop_to_schedule을(를) 초과하지 않는 것으로 가정합니다. 이 항목에서 설명하는 작업자 크기 및 개수도 마찬가지입니다.

mw1.small

- 최대 50개의 DAG 용량
- 5개의 동시 작업(기본값)
- 1개의 vCPU
- 2GB RAM

mw1.medium

- 최대 200개의 DAG 용량
- 10개의 동시 작업(기본값)
- 2개의 vCPU
- 4GB RAM

mw1.large

- 최대 1,000개의 DAG 용량
- 20개의 동시 작업(기본값)
- 4개의 vCPU
- 8GB RAM

mw1.xlarge

- 최대 2000DAG 용량
- 40개의 동시 작업 (기본값)
- 8개의 vCPU
- 24기가바이트 램

mw1.2xlarge

- 최대 4000 DAG 용량

- 동시 작업 80개 (기본값)
- vCPU 16개
- 48기가바이트 램

`celery.worker_autoscale`을(를) 작업자당 작업을 늘리는 데 사용할 수 있습니다. 자세한 내용은 the section called “고성능 사용 사례 예시”을 참조하세요.

Apache Airflow 스케줄러

다음 섹션에는 Amazon MWAA에서 사용할 수 있는 Apache Airflow 스케줄러 옵션과 스케줄러 수가 트리거 수에 미치는 영향이 나와 있습니다.

Apache Airflow에서 트리거는 트리거를 사용하여 지정된 특정 조건이 충족될 때까지 연기되는 작업을 관리합니다. Amazon MWAA에서 트리거는 동일한 Fargate 작업에서 스케줄러와 함께 실행됩니다. 이에 따라 스케줄러 수를 늘리면 사용 가능한 트리거 수가 늘어나며, 환경에서 지연된 작업을 관리하는 방식이 최적화됩니다. 이렇게 하면 작업을 효율적으로 처리할 수 있어 조건이 충족될 때 작업이 실행되도록 즉시 일정을 잡을 수 있습니다.

Apache Airflow v2

- v2 - 2 ~ 5를 수락합니다. 기본값은 2입니다.

Amazon MWAA 작업자 자동 크기 조정 구성

Auto Scaling 메커니즘은 Apache Airflow용 Amazon Managed Workflows 환경에서 실행 중인 작업과 대기 중인 작업에 대한 응답으로 Apache Airflow 작업자 수를 자동으로 늘리고, 대기 중이거나 실행 중인 작업이 더 이상 없을 경우 추가 작업자를 처리합니다. 이 페이지에서는 Amazon MWAA 콘솔을 사용하여 사용자 환경에서 실행되는 Apache Airflow 작업자의 최대 수를 지정하여 Auto Scaling을 구성하는 방법을 설명합니다.

> **Note**
>
> Amazon MWAA는 Apache Airflow 지표를 사용하여 Celery Executor 작업자가 추가로 필요한 시기를 결정하고 필요에 따라 Fargate 작업자 수를 `max-workers`에서 지정한 값까지 늘립니다. 추가 작업자가 작업을 완료하고 작업량이 감소하면 Amazon MWAA에서 해당 작업자를 제거하여 설정된 값으로 축소합니다. `min-workers`

축소하는 동안 작업자가 새 작업을 배정하는 경우 Amazon MWAA는 Fargate 리소스를 유지하고 작업자를 제거하지 않습니다. 자세한 내용은 Amazon MWAA 자동 크기 조정 작동 방식을 참조하십시오.

Sections

작업자 규모 조정 작동 방식

Amazon MWAA는 `RunningTasks` 및 `QueuedTasks` 지표를 사용합니다. 여기서 (실행 중인 작업 + 대기 중인 작업)/(작업자당 작업) = (필요한 작업자)입니다. 필요한 작업자 수가 현재 작업자 수보다 많으면 Amazon MWAA는 Fargate 작업자 컨테이너를 이 값에 `max-workers`에서 지정한 최대값까지 추가합니다.

워크로드가 감소하고 `QueuedTasks` 측정치 합계가 감소함에 따라 Amazon MWAA는 Fargate에 작업 환경을 위해 작업자 규모를 축소하도록 요청합니다. `RunningTasks` 아직 작업을 완료하고 있는 모든 작업자는 작업을 완료할 때까지 축소 중에도 보호 상태를 유지합니다. 작업량에 따라 작업은 대기열에 남아 있고 작업자는 규모를 축소할 수 있습니다.

Amazon MWAA 콘솔 사용

Amazon MWAA 콘솔에서 사용자 환경에서 동시에 실행할 최대 작업자 수를 선택할 수 있습니다. 기본적으로 최대값을 최대 25까지 지정할 수 있습니다.

작업자 수를 구성하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. 다음을 선택합니다.

5. 환경 클래스 창의 최대 작업자 수에 값을 입력합니다.
6. 저장을 선택합니다.

> **Note**
>
> 환경에 변경 사항이 적용되려면 몇 분 정도 걸릴 수 있습니다.

고성능 사용 사례 예시

다음 섹션에서는 환경에서 고성능 및 병렬 처리를 활성화하는 데 사용할 수 있는 구성 유형에 대해 설명합니다.

온프레미스 Apache Airflow

일반적으로 온프레미스 Apache Airflow 플랫폼에서는 파일에서 작업 병렬성, 자동 크기 조정 및 동시성 설정을 구성합니다. `airflow.cfg`

- `core.parallelism` – 스케줄러당 동시에 실행할 수 있는 최대 작업 인스턴스 수입니다.
- `core.dag_concurrency` – DAG의 최대 동시성(작업자 아님)
- `celery.worker_autoscale` – 모든 작업자에서 동시에 실행할 수 있는 최대 및 최소 작업 수입니다.

예를 들어 `core.parallelism`을(를) `100`(으)로 설정하고 `core.dag_concurrency`을(를) `7`(으)로 설정한 경우 DAG가 2개인 경우에도 총 14개의 작업만 동시에 실행할 수 있습니다. 전체 병렬 처리가 `core.parallelism`에서 `100`(으)로 설정되어 있더라도 각 DAG는 `core.dag_concurrency`에서 7개의 작업만 동시에 실행하도록 설정되어 있습니다.

Amazon MWAA 환경

Amazon MWAA 환경에서는 Amazon MWAA에서 Apache Airflow 구성 옵션 사용Amazon MWAA 환경 클래스 구성, 및 최대 작업자 수 자동 조정 메커니즘을 사용하여 Amazon MWAA 콘솔에서 직접 이러한 설정을 구성할 수 있습니다. Amazon MWAA 콘솔의 Apache Airflow 구성 옵션으로는 드롭다운 목록에서 사용할 수 없지만 `core.dag_concurrency` 사용자 지정 Apache Airflow 구성 옵션으로 추가할 수 있습니다.

환경을 만들 때 다음 설정을 선택했다고 가정해 보겠습니다.

1. 각 워커가 기본적으로 실행할 수 있는 최대 동시 작업 수와 컨테이너의 vCPU를 제어하는 mw1.small 환경 클래스입니다.
2. 최대 작업자 수의 10 작업자 기본 설정입니다.
3. 작업자당 5,5 작업 중 celery.worker_autoscale 작업에 대한 Apache Airflow 구성 옵션입니다.

즉, 해당 환경에서 50개의 작업을 동시에 실행할 수 있습니다. 50개를 초과하는 모든 작업은 대기열에 추가되며 실행 중인 작업이 완료될 때까지 기다립니다.

더 많은 동시 작업을 실행합니다. 다음 구성을 사용하여 더 많은 작업을 동시에 실행하도록 환경을 수정할 수 있습니다.

1. mw1.medium (기본 동시 작업 10개) 환경 클래스를 선택하여 각 작업자가 기본적으로 실행할 수 있는 최대 동시 작업 수와 컨테이너의 vCPU를 늘립니다.
2. celery.worker_autoscale을(를) Apache Airflow 구성 옵션으로 추가합니다.
3. 최대 작업자 수를 늘립니다. 이 예제에서 최대 작업자를 10에서 20(으)로 늘리면 환경에서 실행할 수 있는 동시 작업 수가 두 배로 늘어납니다.

최소 작업자를 지정합니다. () 를 사용하여 사용자 환경에서 실행되는 Apache Airflow 작업자의 최소 및 최대 수를 지정할 수도 있습니다. AWS Command Line Interface AWS CLI예:

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

자세한 내용은 AWS CLI의 update-environment 명령을 참조하십시오.

실행 상태에서 멈춘 작업 문제 해결

드문 경우이긴 하지만 Apache Airflow는 작업이 아직 실행 중이라고 생각할 수 있습니다. 이 문제를 해결하려면 Apache Airflow UI에서 문제가 있는 작업을 지워야 합니다. 자세한 정보는 작업이 중단되거나 완료되지 않은 것으로 보임 문제 해결 단원을 참조하십시오.

다음 단계

- Amazon MWAA의 Apache Airflow 성능 튜닝에서 환경 성능 조정을 위해 권장하는 모범 사례에 대해 자세히 알아봅니다.

Amazon MWAA 웹 서버 자동 크기 조정 구성

Apache Airflow Apache Airflow v2.2.2 이상을 실행하는 환경에서 Amazon MWAA는 변동이 심한 워크로드를 처리하도록 웹 서버를 동적으로 확장하여 최대 부하 시 성능 문제를 방지합니다. Amazon MWAA는 CPU 사용률 및 활성 연결 수를 기반으로 웹 서버 수를 자동으로 조정함으로써 Apache Airflow 환경이 REST API 요청, CLI 사용 또는 더 많은 동시 Apache Airflow 사용자 인터페이스 사용자 등으로 인한 수요 증가를 원활하게 수용할 수 있도록 합니다.

Sections

웹 서버 스케일링 작동 방식

Amazon MWAA는 컨테이너 지표와 로드 밸런서 지표를 사용하여 트래픽 양에 따라 웹 서버 확장이 필요한지 여부를 결정합니다. `CPUUtilization` `ActiveConnectionCount` `CPUUtilization`가 70보다 높거나 `ActiveConnectionCount` 15보다 높은 경우 Amazon MWAA는 에서 지정한 최대값까지 Fargate 웹 서버 컨테이너를 추가합니다. `MaxWebservers`

트래픽이 감소하고 `CPUUtilization` 및 `ActiveConnectionCount` 값이 감소함에 따라 Amazon MWAA는 Fargate에 환경용 웹 서버 컨테이너를 에서 설정한 최소값으로 축소하도록 요청합니다. `MinimumWebservers`

Amazon MWAA 콘솔 사용

Amazon MWAA 콘솔에서 사용자 환경에서 동시에 실행할 수 있는 웹 서버의 수를 선택할 수 있습니다. 기본적으로 최소 웹 서버 수는 2개이고 최대 웹 서버 수는 5개입니다.

웹 서버 수를 구성하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. 다음을 선택합니다.
5. 환경 클래스 창의 최대 웹 서버 수에 값을 입력합니다.
6. 그런 다음 최소 웹 서버 수에 값을 입력합니다.

7. 저장을 선택합니다.

> **Note**
>
> 환경에 변경 사항이 적용되려면 몇 분 정도 걸릴 수 있습니다.

Amazon MWAA에서 Apache Airflow 구성 옵션 사용

Apache Airflow 구성 옵션은 Amazon Managed Workflows for Apache Airflow 환경에 환경 변수로 연결할 수 있습니다. 제안된 드롭다운 목록에서 선택하거나 Amazon MWAA 콘솔에서 Apache Airflow 버전의 사용자 지정 구성 옵션을 지정할 수 있습니다. 이 페이지에서는 사용 가능한 Apache Airflow 구성 옵션과 이러한 옵션을 사용하여 사용자 환경의 Apache Airflow 구성 설정을 재정의하는 방법을 설명합니다.

목차

- 다음 단계

필수 조건

이 페이지의 단계를 완료하려면 먼저 다음이 필요합니다.

- 권한 — 관리자는 사용자 환경에 대한 AmazonMWAA 액세스 액세스 제어 정책에 FullConsole 대한 액세스 권한을 사용자 AWS 계정에 부여했어야 합니다. 또한 실행 역할이 Amazon MWAA 환경에서 사용 중인 AWS 리소스에 액세스할 수 있도록 허용해야 합니다.
- 액세스 — 종속성을 웹 서버에 직접 설치하기 위해 퍼블릭 리포지토리에 액세스해야 하는 경우 퍼블릭 네트워크 웹 서버 액세스로 환경을 구성해야 합니다. 자세한 내용은 the section called “Apache Airflow 액세스 모드” 단원을 참조하십시오.
- Amazon S3 구성 — DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 저장하는 데 사용되는 Amazon S3 버킷은 퍼블릭 액세스가 차단되고 버전 관리가 활성화된 상태로 구성되어야 합니다.

작동 방식

환경을 생성할 때 Amazon MWAA는 사용자가 Amazon MWAA 콘솔의 Airflow 구성 옵션에 지정한 구성 설정을 사용자 환경의 컨테이너에 환경 변수로 연결합니다. AWS Fargate `airflow.cfg`에서 같은 이름의 설정을 사용하는 경우 Amazon MWAA 콘솔에서 지정하는 옵션이 `airflow.cfg`의 값보다 우선합니다.

기본적으로 Amazon MWAA 환경의 Apache Airflow UI에는 표시되지 않지만 구성을 노출하는 설정을 포함하여 Amazon MWAA 콘솔에서 직접 Apache Airflow 구성 옵션을 변경할 수 있습니다. `airflow.cfg` `webserver.expose_config`

구성 옵션을 사용하여 Apache Airflow v2에서 플러그인 로드

Apache Airflow v2에서는 기본적으로 `core.lazy_load_plugins : True` 설정을 사용하여 플러그인이 ‘느리게’ 로드되도록 구성됩니다. Apache Airflow v2에서 사용자 정의 플러그인을 사용하는 경우 각 Airflow 프로세스를 시작할 때 플러그인을 로드하여 기본 설정을 재정의하기 위해 `core.lazy_load_plugins : False`을(를) Apache Airflow 구성 옵션으로 추가해야 합니다.

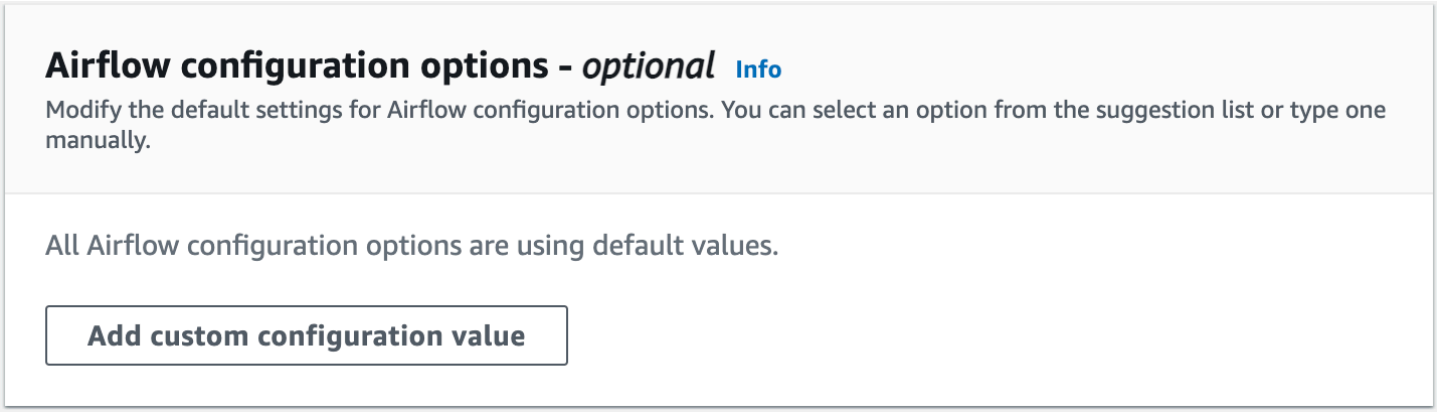
구성 옵션 개요

Amazon MWAA 콘솔에 구성을 추가하면 Amazon MWAA가 구성을 환경 변수로 작성합니다.

- 나열된 옵션. 드롭다운 목록에서 Apache Airflow 버전에 사용할 수 있는 구성 설정 중 하나를 선택할 수 있습니다. 예: `dag_concurrency` : 16. 구성 설정은 사용자 환경의 Fargate 컨테이너에 `AIRFLOW__CORE__DAG_CONCURRENCY : 16`와(과) 같이 변환됩니다.
- 사용자 지정 옵션. 드롭다운 목록에서 Apache Airflow 버전에 대해 나열되지 않은 Airflow 구성 옵션을 지정할 수도 있습니다. 예: `foo.user` : `YOUR_USER_NAME`. 구성 설정은 사용자 환경의 Fargate 컨테이너에 `AIRFLOW__FOO__USER : YOUR_USER_NAME`와(과) 같이 변환됩니다.

Apache Airflow 구성 옵션

다음 이미지는 Amazon MWAA 콘솔에서 Apache Airflow 구성 옵션을 사용자 지정할 수 있는 위치를 보여줍니다.



Apache Airflow 레퍼런스

Apache Airflow에서 지원하는 구성 옵션 목록은 Apache Airflow 참조 가이드의 구성 참조를 참조하십시오. Amazon MWAA에서 실행 중인 Apache Airflow 버전의 옵션을 보려면 드롭다운 목록에서 버전을 선택합니다.

Amazon MWAA 콘솔 사용

다음 절차에서는 Airflow 구성 옵션을 환경에 추가하는 단계를 안내합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. 다음을 선택합니다.
5. Airflow 구성 옵션 창에서 사용자 지정 구성 추가를 선택합니다.

6. 드롭다운 목록에서 구성을 선택하고 값을 입력하거나, 사용자 정의 구성을 입력하고 값을 입력합니다.
7. 추가하려는 각 구성에 대해 사용자 정의 구성 추가를 선택합니다.
8. 저장을 선택합니다.

구성 참조

다음 섹션에는 Amazon MWAA 콘솔의 드롭다운 목록에 있는 사용 가능한 Apache Airflow 구성 목록이 포함되어 있습니다.

이메일 구성

다음 목록은 Amazon MWAA에서 사용할 수 있는 Airflow 이메일 알림 구성 옵션을 보여줍니다.

SMTP 트래픽에는 포트 587을 사용하는 것이 좋습니다. 기본적으로 모든 Amazon EC2 인스턴스의 포트 25에서 아웃바운드 SMTP 트래픽을 AWS 차단합니다. 포트 25에서 아웃바운드 트래픽을 보내려는 경우 이 제한을 제거하도록 요청할 수 있습니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	email.email_backend	email_backend의 이메일 알림에 사용되는 Apache Airflow 유틸리티입니다.	airflow.utils.email.send_email_smtp
v2	smtp.smtp_host	smtp_host의 이메일 주소에 사용되는 아웃바운드 서버의 이름입니다.	localhost
v2	smtp.smtp_starttls	전송 계층 보안(TLS)은 smtp_Starttls에서 인터넷을 통해 이메일을 암호화하는 데 사용됩니다.	False

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	smtp.smtp_ssl	보안 소켓 계층(SSL)은 smtp_ssl에서 서버와 이메일 클라이언트를 연결하는 데 사용됩니다.	True
v2	smtp.smtp_port	smtp_port에서 서버에 지정된 전송 제어 프로토콜(TCP) 포트입니다.	587
v2	smtp.smtp_mail_from	smtp_mail_from의 아웃바운드 이메일 주소입니다.	myemail@domain.com

작업 구성

다음 목록은 Amazon MWAA의 Airflow 작업에 대해 드롭다운 목록에서 사용할 수 있는 구성을 보여줍니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	core.default_task_retries	default_task_retries에서 Apache Airflow 작업을 재시도하는 횟수입니다.	3
v2	core.parallelism	전체 환경에서 병렬로 동시에 실행할 수 있는 작업 인스턴스의 최대 수(병렬 처리).	40

스케줄러 구성

다음 목록은 Amazon MWAA의 드롭다운 목록에서 사용할 수 있는 Apache Airflow 스케줄러 구성을 보여줍니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	scheduler.catchup_by_default	catchup_by_default의 특정 시간 간격을 '따라잡기' 위해 DAG 실행을 생성하도록 스케줄러에 지시합니다.	False
v2	scheduler.scheduler_zombie_task_threshold	스케줄러에게 작업 인스턴스를 실패로 표시하고 scheduler_zombie_task_threshold에 작업을 다시 예약할지 여부를 알려줍니다.	300

작업자 구성

다음 목록은 Amazon MWAA의 드롭다운 목록에서 사용할 수 있는 Airflow 작업자 구성을 보여줍니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	celery.worker_autoscale	worker_autoscale의 Celery Executor를 사용하여 모든 작업자에서 동시에 실행할 수 있는 최대 및 최소 작업 수입니다.	16,12

Airflow 버전	Airflow 구성 옵션	설명	예시 값
		값은 `max_concurrency,min_concurrency` 와(과) 같은 순서로 쉼표로 구분해야 합니다.	

웹 서버 구성

다음 목록은 Amazon MWAA의 드롭다운 목록에서 사용할 수 있는 Airflow 웹 서버 구성을 보여줍니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2	webserver.default_ui_timezone	default_ui_timezone의 기본 Apache Airflow UI 날짜/시간 설정입니다. **Note** `default_ui_timezone` 옵션을 설정해도 DAG가 실행되도록 예약된 시간대는 변경되지 않습니다. DAG의 시간대를 변경하려면 사용자 지정 플러그인을 사용	America/New_York

Airflow 버전	Airflow 구성 옵션	설명	예시 값
		합니다. 자세한 정보는 the section called "DAG 시간대 변경"을 참조하세요.	

트리거 구성

다음 목록은 Amazon MWA에서 사용할 수 있는 Apache Airflow 트리거 구성을 보여줍니다.

Apache Airflow v2

Airflow 버전	Airflow 구성 옵션	설명	예시 값
v2.7	mwaa.triggerer_enabled	Amazon MWAA에서 트리거를 활성화 및 비활성화하는 데 사용됩니다. 기본적으로 이 값은 `True`로 설정됩니다. `False`(으)로 설정하면 Amazon MWAA는 스케줄러에서 트리거 프로세스를 시작하지 않습니다.	True
v2.7	triggerer.default_capacity	각 트리거가 병렬로 실행할 수 있는 트리거 수를 정의합니다. Amazon MWAA에서는 두 구성 요소가 나란히 실행되므로 각 트리거와 각 스케줄러별로 이 용량이 설정	125

Airflow 버전	Airflow 구성 옵션	설명	예시 값
		됩니다. 스케줄러별 기본값은 각각60,,, 및 소형 125 250500, 중형 및 1000 대형, xlarge 및 2xlarge 인스턴스로 설정되어 있습니다.	

예제 및 샘플 코드

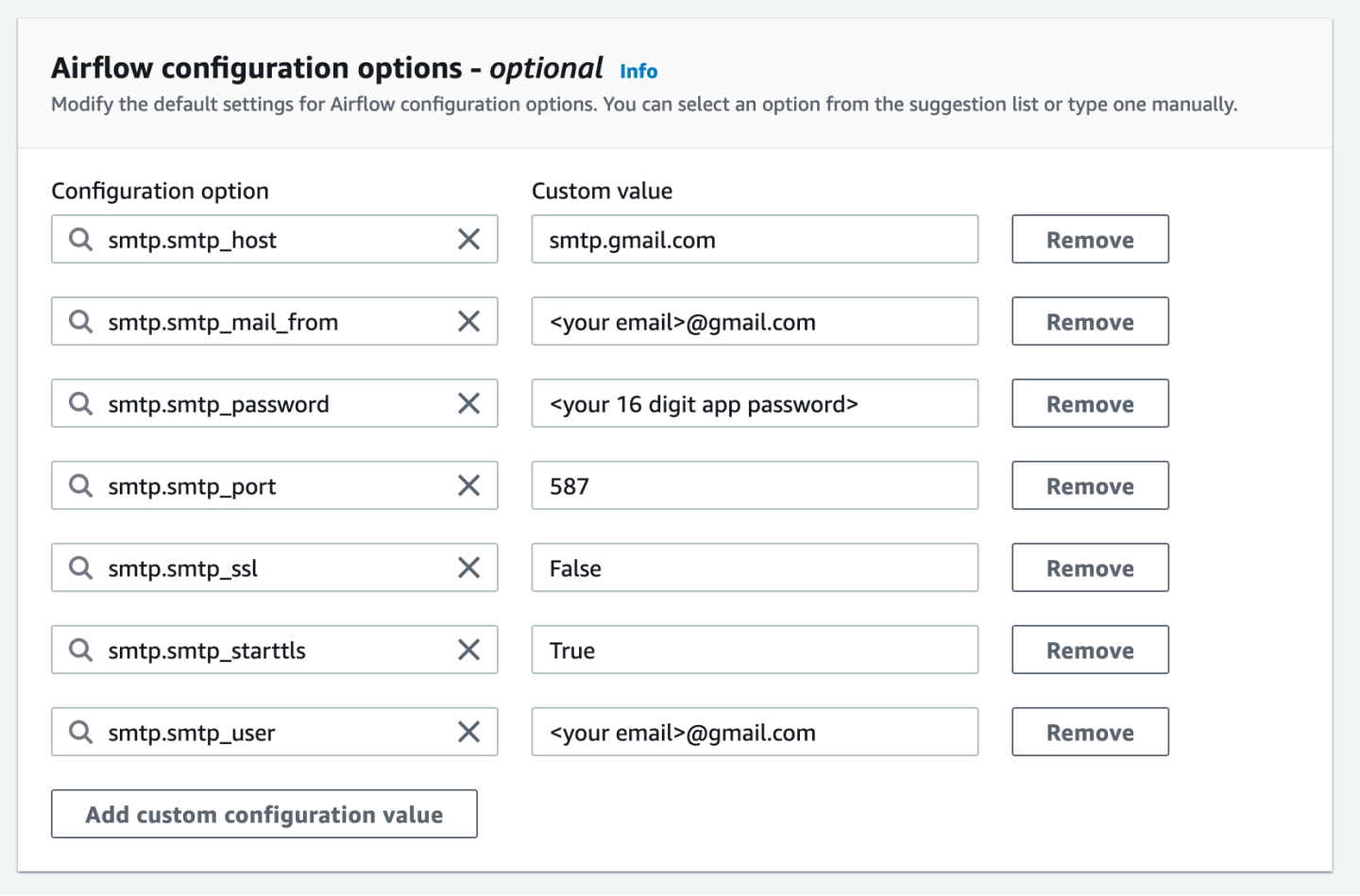
DAG 예제

다음 DAG를 사용하여 email_backend Apache Airflow 구성 옵션을 인쇄할 수 있습니다. Amazon MWAA 이벤트에 대한 응답으로 실행하려면 Amazon S3 스토리지 버킷에서 사용자 환경의 DAG 폴더에 코드를 복사합니다.

```
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()
```

이메일 알림 설정 예시

앱 암호를 사용하여 Gmail.com 이메일 계정에 다음과 같은 Apache Airflow 구성 옵션을 사용할 수 있습니다. 자세한 내용은 Gmail 도움말 참조 가이드의 앱 암호를 사용하여 로그인하기를 참조하십시오.



다음 단계

- DAG 추가 또는 업데이트의 Amazon S3 버킷에 DAG 폴더를 업로드하는 방법을 알아봅니다.

Apache Airflow 버전 업그레이드

Amazon MWAA는 마이너 버전 업그레이드를 지원합니다. 즉, 환경을 버전 x.4.z에서 버전 x.5.z(으)로 업그레이드할 수 있습니다. 예를 들어 버전 1.y.z에서 버전 2.y.z로 업그레이드하는 메이저 버전 업그레이드를 수행하려면 새 환경을 만들고 리소스를 마이그레이션해야 합니다. Apache Airflow의 새 메이저 버전으로 업그레이드하는 방법에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드의 새 Amazon MWAA 환경으로 마이그레이션을 참조하십시오.

업그레이드 프로세스 중에 Amazon MWAA는 환경 메타데이터의 스냅샷을 캡처하고 작업자, 스케줄러, 웹 서버를 새 Apache Airflow 버전으로 업그레이드한 다음 최종적으로 스냅샷을 사용하여 메타데이터 데이터베이스를 복원합니다.

> **Note**
>
> 사용자 환경에 맞게 Apache Airflow 버전을 다운그레이드할 수 없습니다.

업그레이드하기 전에 DAG 및 기타 워크플로우 리소스가 업그레이드하려는 새 Apache Airflow 버전과 호환되는지 확인합니다. `requirements.txt`를 사용하여 종속성을 관리하는 경우 요구 사항에 지정한 종속성이 새 버전과 호환되는지도 확인해야 합니다.

주제

- 워크플로우 리소스를 업그레이드합니다.
- 새 버전 지정

워크플로우 리소스를 업그레이드합니다.

Apache Airflow 버전을 변경할 때마다 `requirements.txt`에서 올바른 `--constraint` URL을 참조하는지 확인합니다.

> **Warning**
>
> 업그레이드 중에 대상 Apache Airflow 버전과 호환되지 않는 요구 사항을 지정하면 이전 요구 사항 버전이 있는 이전 버전의 Apache Airflow로 롤백하는 데 오랜 시간이 걸릴 수 있습니다.

워크플로우 리소스를 마이그레이션하려면

1. aws-mwaa-local-runner 리포지토리의 포크를 생성하고 Amazon MWAA 로컬 러너의 사본을 복제합니다.
2. 업그레이드하려는 버전과 일치하는 aws-mwaa-local-runner 리포지토리의 브랜치를 확인합니다.
3. Amazon MWAA 로컬 러너 CLI 도구를 사용하여 도커 이미지를 빌드하고 로컬에서 Apache Airflow를 실행할 수 있습니다. 자세한 내용은 GitHub 리포지토리에서 로컬 러너 README를 참조하십시오.

4. `requirements.txt`을(를) 업데이트하려면 Amazon MWAA 사용 설명서의 Python 종속성 관리에서 권장하는 모범 사례를 따르십시오.
5. (선택 사항) 업그레이드 프로세스의 속도를 높이려면 환경의 메타데이터 데이터베이스를 정리합니다. 메타데이터 양이 많은 환경은 업그레이드에 상당히 오랜 시간이 걸릴 수 있습니다.
6. 워크플로우 리소스를 성공적으로 테스트한 후 DAG와 `requirements.txt`, 플러그인을 환경의 Amazon S3 버킷으로 복사합니다.

이제 환경을 편집하고, 새 Apache Airflow 버전을 지정하고, 업데이트 절차를 시작할 준비가 되었습니다.

새 버전 지정

새 Apache Airflow 버전과의 호환성을 보장하기 위해 워크플로우 리소스를 업데이트한 후, 다음을 수행하여 환경의 세부 정보를 편집하고 업그레이드하려는 Apache Airflow 버전을 지정합니다.

> **Note**
>
> 업그레이드를 수행하면 그 과정 동안 환경에서 현재 실행 중인 모든 작업이 종료됩니다. 업데이트 절차에는 최대 2시간이 소요될 수 있으며, 이 기간 동안에는 환경을 사용할 수 없습니다.

콘솔을 사용하여 새 버전을 지정하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경 목록에서 업그레이드할 환경을 선택합니다.
3. 환경 페이지에서 편집을 선택하여 환경을 편집합니다.
4. 환경 세부 정보 섹션의 Airflow 버전에 대해 드롭다운 목록에서 환경을 업그레이드하려는 새 Apache Airflow 버전 번호를 선택합니다.
5. 검토 및 저장 페이지가 표시될 때까지 다음을 선택합니다.
6. 검토 및 저장 페이지에서 변경 사항을 검토한 후 저장을 선택합니다.

변경 사항을 적용하면 해당 환경에서 업그레이드 절차가 시작됩니다. 이 기간 동안 사용자 환경의 상태는 Amazon MWAA가 취하는 조치와 절차의 성공 여부를 나타냅니다.

업그레이드가 성공하면 상태가 UPDATING(으)로 표시되고, Amazon MWAA가 메타데이터 백업을 캡처하면 상태가 CREATING_SNAPSHOT(으)로 표시됩니다. 마지막으로 상태가 처음에는 UPDATING(으)로 돌아가고, 절차가 완료되면 AVAILABLE(으)로 돌아갑니다.

환경 업그레이드에 실패할 경우 환경 상태가 ROLLING_BACK(으)로 표시됩니다. 롤백이 성공하면 업데이트가 실패했지만 환경을 사용할 수 있음을 나타내는 UPDATE_FAILED 상태가 먼저 표시됩니다. 롤백이 실패하면 UNAVAILABLE 상태가 표시되며 환경에 액세스할 수 없음을 나타냅니다.

Amazon MWAA에서 시작 스크립트 사용

시작 스크립트는 DAG, 요구 사항 및 플러그인과 유사하게 환경의 Amazon S3 버킷에서 호스팅하는 쉘(.sh) 스크립트입니다. Amazon MWAA는 모든 개별 Apache Airflow 구성 요소(작업자, 스케줄러, 웹 서버)에서 시작 중에 요구 사항을 설치하고 Apache Airflow 프로세스를 초기화하기 전에 이 스크립트를 실행합니다. 시작 스크립트를 사용하여 다음을 수행합니다.

- 런타임 설치 – 워크플로우와 연결에 필요한 Linux 런타임을 설치합니다.
- 환경 변수 구성 – 각 Apache Airflow 구성 요소에 대한 환경 변수를 설정합니다. 일반 변수를 덮어씁니다(예: PATH, PYTHONPATH 및 LD_LIBRARY_PATH).
- 키 및 토큰 관리 – 사용자 지정 리포지토리의 액세스 토큰을 requirements.txt에 전달하고 보안 키를 구성합니다.

다음 항목에서는 Linux 런타임을 설치하도록 시작 스크립트를 구성하고, 환경 변수를 설정하며, CloudWatch Logs를 사용하여 관련 문제를 해결하는 방법을 설명합니다.

주제

- 시작 스크립트 구성
- 시작 스크립트를 사용하여 Linux 런타임을 설치합니다.
- 시작 스크립트를 사용하여 환경 변수를 설정합니다.

시작 스크립트 구성

기존 Amazon MWAA 환경에서 시작 스크립트를 사용하려면 사용자 환경의 Amazon S3 버킷에 .sh 파일을 업로드합니다. 그런 다음 스크립트를 환경에 연결하려면 환경 세부 정보에 다음을 지정합니다.

- 스크립트에 대한 Amazon S3 URL 경로 – 버킷에 호스팅된 스크립트의 상대 경로입니다. 예를 들어, s3://mwaa-environment/*startup.sh*

- 스크립트의 Amazon S3 버전 ID – Amazon S3 버킷에 있는 시작 쉘 스크립트의 버전입니다. 스크립트를 업데이트할 때마다 Amazon S3가 파일에 할당하는 버전 ID를 지정해야 합니다. 버전 ID는 유니코드, UTF-8 인코딩, URL 지원, 불투명 문자열이며 길이가 1,024바이트를 넘지 않습니다(예: `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`).

이 섹션의 단계를 완료하려면 다음 샘플 스크립트를 사용합니다. 스크립트는 할당된 값을 `MWAA_AIRFLOW_COMPONENT`에 출력합니다. 이 환경 변수는 스크립트가 실행되는 각 Apache Airflow 구성 요소를 식별합니다.

코드를 복사하고 로컬에 `startup.sh`로 저장합니다.

```
#!/bin/sh

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

다음으로 Amazon S3 버킷에 스크립트를 업로드합니다.

AWS Management Console

쉘 스크립트를 업로드하려면(콘솔)

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 환경에 연결된 버킷의 이름을 선택합니다.
3. 객체 탭에서 업로드를 선택합니다.
4. 업로드 페이지에서 생성한 쉘 스크립트를 드래그 앤 드롭합니다.
5. 업로드를 선택합니다.

스크립트는 객체 목록에 나타납니다. Amazon S3는 파일에 대한 새 버전 ID를 만듭니다. 스크립트를 업데이트하고 동일한 파일 이름을 사용하여 다시 업로드하면 새 버전 ID가 파일에 할당됩니다.

AWS CLI

쉘 스크립트(CLI)를 생성하고 업로드하려면

1. 새 명령 프롬프트를 열고 Amazon S3 `ls` 명령을 실행하여 환경과 관련된 버킷을 나열하고 식별합니다.

```
$ aws s3 ls
```

2. 쉘 스크립트를 저장한 폴더로 이동합니다. 새 프롬프트 창에서 cp를 사용하여 스크립트를 버킷에 업로드합니다. *your-s3-bucket*을 자신의 정보로 바꿉니다.

```
$ aws s3 cp startup.sh s3://your-s3-bucket/startup.sh
```

성공하면 Amazon S3가 객체에 대한 URL 경로를 출력합니다.

```
upload: ./startup.sh to s3://your-s3-bucket/startup.sh
```

3. 다음 명령을 사용하여 스크립트의 최신 버전 ID를 검색합니다.

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

```
BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

스크립트를 환경에 연결할 때 이 버전 ID를 지정합니다.

이제 스크립트를 환경에 연결합니다.

AWS Management Console

스크립트를 환경과 연결하려면(콘솔)

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 업데이트하려는 환경의 행을 선택한 다음 편집을 선택합니다.
3. 세부 정보 지정 페이지에서 시작 스크립트 파일(선택 사항)에 스크립트의 Amazon S3 URL을 입력합니다(예: s3://*your-mwaa-bucket*/startup-sh.).
4. 드롭다운 목록에서 최신 버전을 선택하거나 S3 찾아보기를 사용하여 스크립트를 찾습니다.
5. 다음을 선택하여 검토 후 저장 페이지로 이동합니다.
6. 변경 사항을 검토한 다음 저장을 선택합니다.

환경 업데이트는 10~30분 정도 걸릴 수 있습니다. Amazon MWAA는 사용자 환경의 각 구성 요소가 재시작될 때 시작 스크립트를 실행합니다.

AWS CLI

스크립트를 환경과 연결하려면(CLI)

- 명령 프롬프트를 열고 `update-environment`를 사용하여 스크립트의 Amazon S3 URL과 버전 ID를 지정합니다.

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

성공할 경우 Amazon MWAA는 환경에 대한 Amazon 리소스 이름(ARN) 을 반환합니다.

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

환경 업데이트는 10~30분 정도 걸릴 수 있습니다. Amazon MWAA는 사용자 환경의 각 구성 요소가 재시작될 때 시작 스크립트를 실행합니다.

마지막으로 로그 이벤트를 검색하여 스크립트가 예상대로 작동하는지 확인합니다. 각 Apache Airflow 구성 요소에 대한 로깅을 활성화하면 Amazon MWAA가 새 로그 그룹과 로그 스트림을 생성합니다. 자세한 내용은 Apache Airflow 로그 유형을 참조하십시오.

AWS Management Console

Apache Airflow 로그 스트림을 확인하려면(콘솔)

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 모니터링 창에서 로그를 보려는 로그 그룹을 선택합니다(예: Airflow 스케줄러 로그 그룹).
4. CloudWatch 콘솔의 로그 스트림 목록에서 접두사가 다음과 같은 스트림을 선택합니다: `startup_script_exection_ip`.
5. 로그 이벤트 창에는 `MWAA_AIRFLOW_COMPONENT` 값을 인쇄하는 명령의 출력이 표시됩니다. 예를 들어 스케줄러 로그의 경우 다음과 같은 결과를 얻을 수 있습니다.

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

이전 단계를 반복하여 작업자 및 웹 서버 로그를 볼 수 있습니다.

시작 스크립트를 사용하여 Linux 런타임을 설치합니다.

시작 스크립트를 사용하여 Apache Airflow 구성 요소의 운영 체제를 업데이트하고 워크플로우에 사용할 추가 런타임 라이브러리를 설치합니다. 예를 들어, 다음 스크립트는 `yum update`를 실행하여 운영 체제를 업데이트합니다.

시작 스크립트에서 `yum update`를 실행할 때는 예제와 `--exclude=python*`을 사용하여 Python을 제외해야 합니다. 실행 환경을 위해 Amazon MWAA는 사용자 환경과 호환되는 특정 버전의 Python을 설치합니다. 따라서 시작 스크립트를 사용하여 환경의 Python 버전을 업데이트할 수 없습니다.

```
#!/bin/sh

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

특정 Apache Airflow 구성 요소에 런타임을 설치하려면 `MWAA_AIRFLOW_COMPONENT`, `if` 및 `fi` 조건문을 사용합니다. 이 예제에서는 단일 명령을 실행하여 웹 서버가 아닌 스케줄러와 작업자에는 `libaio` 라이브러리를 설치합니다.

> **Important**
>
> - 프라이빗 웹 서버를 구성한 경우 설치 시간 초과를 방지하려면 다음 조건을 사용하거나 모든 설치 파일을 로컬로 제공해야 합니다.
> - sudo를 사용하여 관리자 권한이 필요한 작업을 실행합니다.

```
#!/bin/sh
```

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

시작 스크립트를 사용하여 Python 버전을 확인할 수 있습니다.

```
#!/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 MWAA는 기본 Python 버전 재정의를 지원하지 않습니다. 이렇게 하면 설치된 Apache Airflow 라이브러리와 호환되지 않을 수 있기 때문입니다.

시작 스크립트를 사용하여 환경 변수를 설정합니다.

시작 스크립트를 사용하여 환경 변수를 설정하고 Apache Airflow 구성을 수정할 수 있습니다. 다음에서는 새 변수 `ENVIRONMENT_STAGE`을 정의합니다. DAG 또는 사용자 지정 모듈에서 이 변수를 참조할 수 있습니다.

```
#!/bin/sh

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

시작 스크립트를 사용하여 일반적인 Apache Airflow 또는 시스템 변수를 덮어씁니다. 예를 들어, 지정한 경로에서 바이너리를 찾도록 Python에 지시하도록 `LD_LIBRARY_PATH`를 설정합니다. 이렇게 하면 플러그인을 사용하여 워크플로우에 사용자 지정 바이너리를 제공할 수 있습니다.

```
#!/bin/sh

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

예약된 환경 변수

Amazon MWAA는 중요한 환경 변수 세트를 예약합니다. 예약된 변수를 덮어쓰면 Amazon MWAA가 해당 변수를 기본값으로 복원합니다. 예약된 변수의 목록은 다음과 같습니다.

- MWAA__AIRFLOW__COMPONENT – : scheduler, worker 또는 webserver 값 중 하나로 Apache Airflow 구성 요소를 식별하는 데 사용됩니다.
- AIRFLOW__WEBSERVER__SECRET_KEY – Apache Airflow 웹 서버에서 세션 쿠키에 안전하게 서명하는 데 사용되는 암호 키입니다.
- AIRFLOW__CORE__FERNET_KEY – 메타데이터 데이터베이스에 저장된 민감한 데이터(예: 연결 암호)를 암호화하고 해독하는 데 사용되는 키입니다.
- AIRFLOW_HOME – 구성 파일 및 DAG 파일이 로컬로 저장되는 Apache Airflow 홈 디렉터리의 경로입니다.
- AIRFLOW__CELERY__BROKER_URL – Apache Airflow 스케줄러와 Cellery 작업자 노드 간의 통신에 사용되는 메시지 브로커의 URL입니다.
- AIRFLOW__CELERY__RESULT_BACKEND – Celery 작업의 결과를 저장하는 데 사용되는 데이터베이스의 URL입니다.
- AIRFLOW__CORE__EXECUTOR – Apache Airflow가 사용해야 하는 실행자 클래스입니다. Amazon MWAA에서는 이것은 CeleryExecutor
- AIRFLOW__CORE__LOAD_EXAMPLES – 예제 DAG 로드를 활성화하거나 비활성화하는 데 사용됩니다.
- AIRFLOW__METRICS__METRICS_BLOCK_LIST – CloudWatch에서 Amazon MWAA가 내보내고 캡처하는 Apache Airflow 지표를 관리하는 데 사용됩니다.
- SQL_ALCHEMY_CONN – Amazon MWAA에 Apache Airflow 메타데이터를 저장하는 데 사용되는 PostgreSQL용 RDS 데이터베이스의 연결 문자열입니다.
- AIRFLOW__CORE__SQL_ALCHEMY_CONN – SQL_ALCHEMY_CONN와 동일한 용도로 사용되지만 새 Apache Airflow 이름 지정 규칙을 따릅니다.
- AIRFLOW__CELERY__DEFAULT_QUEUE – Apache Airflow의 Celery 작업에 대한 기본 대기열입니다.
- AIRFLOW__OPERATORS__DEFAULT_QUEUE – 특정 Apache Airflow 연산자를 사용하는 작업의 기본 대기열입니다.
- AIRFLOW_VERSION – Amazon MWAA 환경에 설치된 Apache Airflow 버전.
- AIRFLOW_CONN_AWS_DEFAULT – 다른 AWS 서비스와 통합하는 데 사용되는 기본 AWS 보안 인증입니다.
- AWS_DEFAULT_REGION – 다른 AWS 서비스와 통합하기 위해 기본 보안 인증과 함께 사용할 기본 AWS 리전을 설정합니다.
- AWS_REGION – 정의된 경우 이 환경 변수는 환경 변수 AWS_DEFAULT_REGION 및 프로파일 설정의 값을 재정의합니다.

- PYTHONUNBUFFERED – stdout 및 stderr 스트림을 컨테이너 로그로 전송하는 데 사용됩니다.
- AIRFLOW__METRICS__STATSD_ALLOW_LIST – 목록의 요소로 시작하는 지표를 전송하기 위해 쉼표로 구분된 접두사의 허용 목록을 구성하는 데 사용됩니다.
- AIRFLOW__METRICS__STATSD_ON – StatsD로 지표 전송을 활성화합니다.
- AIRFLOW__METRICS__STATSD_HOST – StatSD 대몬(daemon)에 연결하는 데 사용됩니다.
- AIRFLOW__METRICS__STATSD_PORT – StatSD 대몬(daemon)에 연결하는 데 사용됩니다.
- AIRFLOW__METRICS__STATSD_PREFIX – StatSD 대몬(daemon)에 연결하는 데 사용됩니다.
- AIRFLOW__CELERY__WORKER_AUTOSCALE – 최대 및 최소 동시성을 설정합니다.
- AIRFLOW__CORE__DAG_CONCURRENCY – 한 DAG에서 스케줄러가 동시에 실행할 수 있는 작업 인스턴스 수를 설정합니다.
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG – DAG당 최대 활성 작업 수를 설정합니다.
- AIRFLOW__CORE__PARALLELISM – 동시에 실행할 수 있는 최대 작업 인스턴스 수를 정의합니다.
- AIRFLOW__SCHEDULER__PARSING_PROCESSES – DAG를 스케줄링하기 위해 스케줄러가 파싱하는 최대 프로세스 수를 설정합니다.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT – 메시지가 다른 작업자에게 다시 전달되기 전에 작업자가 작업을 승인하기 위해 대기하는 시간(초)을 정의합니다.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION – 기본 셀러리 전송의 AWS 리전을 설정합니다.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES – 기본 셀러리 전송의 큐를 설정합니다.
- AIRFLOW_SCHEDULER_ALLOWED_RUN_ID_PATTERN – DAG를 트리거할 때 run_id 파라미터에 대한 입력의 유효성을 확인하는 데 사용됩니다.
- AIRFLOW__WEBSERVER__BASE_URL – Apache Airflow UI를 호스팅하는 데 사용되는 웹 서버의 URL입니다.

예약되지 않은 환경 변수

시작 스크립트를 사용하여 예약되지 않은 환경 변수를 덮어쓸 수 있습니다. 다음은 이러한 일반적인 변수의 일부를 나열한 것입니다.

- PATH – 운영 체제가 실행 파일 및 스크립트를 검색하는 디렉터리 목록을 지정합니다. 명령줄에서 명령을 실행하면 시스템은 명령을 찾아 실행하기 위해 PATH의 디렉터리를 확인합니다. Apache Airflow에서 사용자 지정 연산자 또는 작업을 생성할 때는 외부 스크립트나 실행 파일을 사용해야 할 수 있습니다. 이러한 파일을 포함한 디렉터리가 PATH 변수에 지정된 디렉터리에 없는 경우, 시스템

이 해당 디렉터리를 찾을 수 없으면 작업이 실행되지 않습니다. PATH에 적절한 디렉터리를 추가하면 Apache Airflow 작업에서 필요한 실행 파일을 찾아 실행할 수 있습니다.

- PYTHONPATH – Python 인터프리터가 가져온 모듈과 패키지를 검색할 디렉터리를 결정하는 데 사용됩니다. 기본 검색 경로에 추가할 수 있는 디렉터리 목록입니다. 이를 통해 인터프리터는 표준 라이브러리에 포함되어 있지 않거나 시스템 디렉터리에 설치되어 있지 않은 Python 라이브러리를 찾아 로드할 수 있습니다. 이 변수를 사용하여 모듈과 사용자 지정 Python 패키지를 추가하고 DAG와 함께 사용할 수 있습니다.
- LD_LIBRARY_PATH – Linux의 동적 링커 및 로더가 공유 라이브러리를 찾고 로드하는 데 사용하는 환경 변수입니다. 기본 시스템 라이브러리 디렉터리보다 먼저 검색되는 공유 라이브러리가 포함된 디렉터리 목록을 지정합니다. 이 변수를 사용하여 사용자 지정 바이너리를 지정할 수 있습니다.
- CLASSPATH – Java 런타임 환경(JRE) 및 Java 개발 키트(JDK) 에서 런타임 시 Java 클래스, 라이브러리 및 리소스를 찾고 로드하는 데 사용됩니다. 컴파일된 Java 코드가 포함된 디렉터리, JAR 파일 및 ZIP 아카이브의 목록입니다.

Amazon MWAA에서 DAG 작업

Amazon Managed Workflows for Apache Airflow 환경에서 DAG(방향성 비순환 그래프)를 실행하려면 환경에 연결된 Amazon S3 스토리지 버킷에 파일을 복사한 다음, DAG 및 지원 파일이 Amazon MWAA 콘솔에서 어디에 있는지 Amazon MWAA에 알려 줍니다. Amazon MWAA는 작업자, 스케줄러 및 웹 서버 간의 DAG 동기화를 처리합니다. 이 가이드에서는 DAG를 추가 또는 업데이트하고 Amazon MWAA 환경에 사용자 지정 플러그인 및 Python 종속성을 설치하는 방법을 설명합니다.

주제

Amazon S3 버킷 개요

Amazon MWAA 환경용 Amazon S3 버킷에는 퍼블릭 액세스 차단이 있어야 합니다. 기본적으로 버킷, 객체 및 관련 하위 리소스(예: 수명 주기 구성)를 비롯한 모든 Amazon S3 리소스는 비공개입니다.

- 즉, 버킷을 만든 AWS 계정인 리소스 소유자만 해당 리소스에 액세스할 수 있습니다. 리소스 소유자(예: 관리자)는 액세스 제어 정책을 작성하여 다른 사람에게 액세스 권한을 부여할 수도 있습니다.
- 설정한 액세스 정책에는 Amazon S3 버킷에 DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 추가할 수 있는 권한이 있어야 합니다. 필요한 권한이 포함된 정책의 예제는 AmazonMWAAFullConsoleAccess를 참조하십시오.

Amazon MWAA 환경용 Amazon S3 버킷에는 버전 관리가 활성화되어 있어야 합니다. Amazon S3 버킷 버전 관리가 활성화하면 새 버전이 생성될 때마다 새 사본이 생성됩니다.

- Amazon S3 버킷에서 `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성에 대해 버전 관리가 활성화됩니다.
- Amazon S3 버킷에서 이러한 파일이 업데이트될 때마다 Amazon MWAA 콘솔에서 `plugins.zip` 및 `requirements.txt`의 버전을 지정해야 합니다.

DAG 추가 또는 업데이트

방향성 비순환 그래프(DAG)는 DAG 구조를 코드로 정의하는 Python 파일 내에 정의됩니다. AWS CLI 또는 Amazon S3 콘솔을 사용하여 DAG를 사용자 환경에 업로드할 수 있습니다. 이 페이지에서는 Amazon S3 버킷의 `dags` 폴더를 사용하여 Amazon Managed Workflows for Apache Airflow 환경에서 Apache Airflow DAG를 추가하거나 업데이트하는 단계를 설명합니다.

섹션

필수 조건

이 페이지의 단계를 완료하려면 먼저 다음이 필요합니다.

- 권한 — 관리자가 사용자 환경의 AmazonMWAAFullConsoleAccess 액세스 제어 정책에 대한 액세스 권한을 AWS 계정에 부여했어야 합니다. 또한 실행 역할이 Amazon MWAA 환경에서 사용자 환경에서 사용하는 AWS 리소스에 액세스할 수 있도록 허용해야 합니다.
- 액세스 — 종속성을 웹 서버에 직접 설치하기 위해 퍼블릭 리포지토리에 액세스해야 하는 경우 퍼블릭 네트워크 웹 서버 액세스로 환경을 구성해야 합니다. 자세한 내용은 the section called “Apache Airflow 액세스 모드” 단원을 참조하십시오.
- Amazon S3 구성 — DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 저장하는 데 사용되는 Amazon S3 버킷은 퍼블릭 액세스가 차단되고 버전 관리가 활성화된 상태로 구성되어야 합니다.

작동 방식

방향비순환 그래프(DAG)는 DAG의 구조를 코드로 정의하는 단일 Python 파일 내에 정의됩니다. 다음 항목으로 구성됩니다.

- DAG 정의.
- DAG 실행 방법과 실행할 작업을 설명하는 연산자.
- 작업 실행 순서를 설명하는 연산자 관계.

Amazon MWAA 환경에서 Apache Airflow 플랫폼을 실행하려면 DAG 정의를 스토리지 버킷의 dags 폴더에 복사해야 합니다. 예를 들어 스토리지 버킷의 DAG 폴더는 다음과 같을 수 있습니다.

Example DAG 폴더

```
dags/
# dag_def.py
```

Amazon MWAA는 Amazon S3 버킷의 새 객체 및 변경된 객체를 Amazon MWAA 스케줄러 및 작업자 컨테이너의 `/usr/local/airflow/dags` 폴더로 30초마다 자동으로 동기화하여 파일 유형에 상관없이 Amazon S3 소스의 파일 계층 구조를 유지합니다. 새 DAG가 Apache Airflow UI에 나타나는 데 걸리는 시간은 `scheduler.dag_dir_list_interval`에 의해 제어됩니다. 기존 DAG에 대한 변경 사항은 다음 DAG 처리 루프에서 적용됩니다.

> **Note**
>
> DAG 폴더에 `airflow.cfg` 구성 파일을 포함할 필요는 없습니다. Amazon MWAA 콘솔에서 기본 Apache Airflow 구성을 재정의할 수 있습니다. 자세한 내용은 Amazon MWAA에서 Apache Airflow 구성 옵션 사용 단원을 참조하십시오.

v2에서 변경된 사항

- 신규: 연산자, 후크, 실행자. DAG의 가져오기 문과 Amazon MWAA에서 사용자가 `plugins.zip`에 지정한 사용자 지정 플러그인이 Apache Airflow v1과 Apache Airflow v2 사이에서 변경되었습니다. 예를 들어, Apache Airflow v1의 `from airflow.contrib.hooks.aws_hook import AwsHook`이 Apache Airflow v2의 `from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook`로 변경되었습니다. 자세한 내용은 Apache Airflow 참조 가이드의 Python API 참조를 참조하십시오.

Amazon MWAA CLI 유틸리티를 사용한 DAG 테스트

- 명령줄 인터페이스(CLI) 유틸리티는 Amazon Managed Workflows for Apache Airflow 환경을 로컬로 복제합니다.
- CLI는 Amazon MWAA 프로덕션 이미지와 유사한 Docker 컨테이너 이미지를 로컬로 구축합니다. 이를 통해 Amazon MWAA에 배포하기 전에 로컬 Apache Airflow 환경을 실행하여 DAG, 사용자 지정 플러그인 및 종속성을 개발하고 테스트할 수 있습니다.
- CLI를 실행하려면 GitHub의 aws-mwaa-local-runner를 참조하십시오.

Amazon S3에 DAG 코드 업로드

Amazon S3 콘솔 또는 AWS Command Line Interface (AWS CLI)를 사용하여 Amazon S3 버킷에 DAG 코드를 업로드할 수 있습니다. 다음 단계에서는 Amazon S3 버킷의 이름이 지정된 dags 폴더에 코드(.py)를 업로드한다고 가정합니다.

AWS CLI 사용

AWS Command Line Interface(AWS CLI)는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI - 버전 2 설치
- AWS CLI – aws configure을 통한 빠른 구성

AWS CLI를 사용하여 업로드하려면

1. 다음 명령을 사용하여 Amazon S3 버킷을 모두 나열합니다.

```
aws s3 ls
```

2. 다음 명령을 사용하여 사용자 환경의 Amazon S3 버킷에 있는 파일과 폴더를 나열합니다.

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

3. 다음 명령은 dag_def.py 파일을 dags 폴더에 업로드합니다.

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

이름이 지정된 dags 폴더가 Amazon S3 버킷에 아직 없는 경우 이 명령은 dags 폴더를 생성하고 이름이 지정된 dag_def.py 파일을 새 폴더에 업로드합니다.

Amazon S3 콘솔 사용

Amazon S3 콘솔은 Amazon S3 버킷의 리소스를 생성 및 관리할 수 있는 웹 기반 사용자 인터페이스입니다. 다음 단계에서는 이름이 지정된 dags DAG 폴더가 있다고 가정합니다.

Amazon S3 콘솔을 사용하여 업로드하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. S3 창의 DAG 코드에서 S3 버킷 링크를 선택하여 Amazon S3 콘솔에서 스토리지 버킷을 엽니다.
4. dags 폴더를 선택합니다.
5. 업로드를 선택합니다.
6. 파일 추가를 선택합니다.
7. dag_def.py의 로컬 사본을 선택하고 업로드를 선택합니다.

Amazon MWAA 콘솔에서 DAG 폴더의 경로 지정(처음)

다음 단계에서는 이름이 지정된 dags Amazon S3 버킷의 폴더에 대한 경로를 지정한다고 가정합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. DAG를 실행할 환경을 선택합니다.
3. 편집(Edit)을 선택합니다.
4. Amazon S3의 DAG 코드 창에서 DAG 폴더 필드 옆에 있는 S3 찾아보기를 선택합니다.
5. dags 폴더를 선택합니다.
6. 선택을 선택합니다.
7. 다음, 환경 업데이트를 선택합니다.

Apache Airflow UI에서 변경 사항 보기

Apache Airflow에 로그인

Apache Airflow UI를 보려면 AWS Identity and Access Management(IAM)의 AWS 계정에 대한 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer 권한이 필요합니다.

Apache Airflow UI에 액세스하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. Airflow UI 열기를 선택합니다.

다음 단계

- GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.

사용자 지정 플러그인 설치

Amazon Managed Workflows for Apache Airflow는 Apache Airflow의 내장 플러그인 관리자를 지원하므로 사용자는 사용자 지정 Apache Airflow 운영자, 후크, 센서 또는 인터페이스를 사용할 수 있습니다. 이 페이지에서는 `plugins.zip` 파일을 사용하여 Amazon MWAA 환경에 Apache Airflow 사용자 지정 플러그인을 설치하는 단계를 설명합니다.

목차

필수 조건

이 페이지의 단계를 완료하려면 먼저 다음이 필요합니다.

- 권한 — 관리자가 사용자 환경의 AmazonMWAAFullConsoleAccess 액세스 제어 정책에 대한 액세스 권한을 AWS 계정에 부여했어야 합니다. 또한 실행 역할이 Amazon MWAA 환경에서 사용자 환경에서 사용하는 AWS 리소스에 액세스할 수 있도록 허용해야 합니다.
- 액세스 — 종속성을 웹 서버에 직접 설치하기 위해 퍼블릭 리포지토리에 액세스해야 하는 경우 퍼블릭 네트워크 웹 서버 액세스로 환경을 구성해야 합니다. 자세한 내용은 the section called “Apache Airflow 액세스 모드” 단원을 참조하십시오.
- Amazon S3 구성 — DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 저장하는 데 사용되는 Amazon S3 버킷은 퍼블릭 액세스가 차단되고 버전 관리가 활성화된 상태로 구성되어야 합니다.

작동 방식

사용자 환경에서 사용자 지정 플러그인을 실행하려면 다음 세 가지 작업을 수행해야 합니다.

1. 로컬에서 `plugins.zip` 파일을 생성합니다.
2. Amazon S3 버킷에 로컬 `plugins.zip` 파일을 업로드합니다.
3. Amazon MWAA 콘솔의 플러그인 파일 필드에 이 파일의 버전을 지정합니다.

> **Note**
>
> `plugins.zip`을 처음으로 Amazon S3 버킷에 업로드하는 경우 Amazon MWAA 콘솔에 파일 경로도 지정해야 합니다. 이 단계는 한 번만 완료하면 됩니다.

v2에서 변경된 사항

- 신규: 연산자, 후크, 실행자. DAG의 가져오기 문과 Amazon MWAA에서 사용자가 `plugins.zip`에 지정한 사용자 지정 플러그인이 Apache Airflow v1과 Apache Airflow v2 사이에서 변경되었습니다. 예를 들어, Apache Airflow v1의 `from airflow.contrib.hooks.aws_hook import AwsHook`이 Apache Airflow v2의 `from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook`로 변경되었습니다. 자세한 내용은 Apache Airflow 참조 가이드의 Python API 참조를 참조하십시오.
- 신규: 플러그인으로 가져오기. `airflow.{operators,sensors,hooks}.<plugin_name>`를 사용하여 플러그인에 추가된 운영자, 센서, 후크 가져오기는 더 이상 지원되지 않습니다. 이러한 확장은 일반 Python 모듈로 가져와야 합니다. v2 이상에서는 DAG 디렉터리에 이들을 배치하고 .airflowignore 파일을 작성해 사용하여 DAG로 구문 분석되지 않도록 제외하는 것이 좋습니다. 자세한 내용은 Apache Airflow 참조 가이드의 모듈 관리 및 사용자 지정 운영자 생성을 참조하십시오.

사용자 지정 플러그인 개요

Apache Airflow의 내장 플러그인 관리자는 파일을 `$AIRFLOW_HOME/plugins` 폴더에 놓기만 하면 외부 기능을 코어에 통합할 수 있습니다. 이를 통해 사용자 지정 Apache Airflow 운영자, 후크, 센서 또는 인터페이스를 사용할 수 있습니다. 다음 섹션에는 로컬 개발 환경의 플랫 및 중첩 디렉터리 구조의 예제 및 plugins.zip 내의 디렉터리 구조를 결정하는 결과 가져오기 문이 나와 있습니다.

사용자 지정 플러그인 디렉터리 및 크기 제한

Apache Airflow 스케줄러와 작업자는 시작하는 동안 `/usr/local/airflow/plugins/*`에 사용자 환경의 AWS 관리형 Fargate 컨테이너에서 사용자 지정 플러그인을 찾습니다.

- 디렉터리 구조. (`/*`의) 디렉터리 구조는 `plugins.zip` 파일 내용을 기반으로 합니다. 예를 들어, 사용자의 `plugins.zip`에 최상위 디렉터리로 `operators` 디렉터리가 포함되어 있는 경우 해당 디렉터리는 사용자 환경의 `/usr/local/airflow/plugins/operators`로 추출됩니다.
- 크기 제한. 1GB 미만의 `plugins.zip` 파일을 사용하는 것이 좋습니다. `plugins.zip` 파일 크기가 클수록 환경의 시작 시간이 길어집니다. Amazon MWAA가 `plugins.zip` 파일의 크기를 명시적으

로 제한하지 않지만, 종속성을 10분 이내에 설치할 수 없는 경우 Fargate 서비스는 시간을 초과하여 환경을 안정적인 상태로 롤백하려고 시도합니다.

> **Note**
>
> Apache Airflow v1.10.12 또는 Apache Airflow v2.0.2를 사용하는 환경의 경우 Amazon MWAA는 Apache Airflow 웹 서버에서 아웃바운드 트래픽을 제한하며, 웹 서버에 직접 플러그인이나 Python 종속성을 설치할 수 없습니다. Apache Airflow v2.2.2부터 Amazon MWAA는 플러그인 및 종속 항목을 웹 서버에 직접 설치할 수 있습니다.

사용자 지정 플러그인의 예제

다음 섹션에서는 Apache Airflow 참조 가이드의 샘플 코드를 사용하여 로컬 개발 환경을 구성하는 방법을 보여줍니다.

plugins.zip 내 플랫 디렉터리 구조 사용 예제

Apache Airflow v2

다음 예제에서는 Apache Airflow v2에 플랫 디렉터리 구조가 있는 `plugins.zip` 파일을 보여줍니다.

Example PythonVirtualEnvOperator plugins.zip이 있는 플랫 디렉터리

다음 예제에서는 Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성에서 PythonVirtualEnvOperator 사용자 지정 플러그인에 대한 plugins.zip 파일의 최상위 트리를 보여줍니다.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

다음 예제에서는 PythonVirtualEnvOperator 사용자 지정 플러그인을 보여줍니다.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v1

다음 예제에서는 Apache Airflow v1에 플랫 디렉터리 구조가 있는 `plugins.zip` 파일을 보여줍니다.

Example PythonVirtualEnvOperator plugins.zip이 있는 플랫 디렉터리

다음 예제에서는 Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성에서 PythonVirtualEnvOperator 사용자 지정 플러그인에 대한 plugins.zip 파일의 최상위 트리를 보여줍니다.

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

다음 예제에서는 PythonVirtualEnvOperator 사용자 지정 플러그인을 보여줍니다.

```
from airflow.plugins_manager import AirflowPlugin
from airflow.operators.python_operator import PythonVirtualenvOperator

def _generate_virtualenv_cmd(self, tmp_dir):
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if self.system_site_packages:
        cmd.append('--system-site-packages')
    if self.python_version is not None:
        cmd.append('--python=python{}'.format(self.python_version))
    return cmd
PythonVirtualenvOperator._generate_virtualenv_cmd=_generate_virtualenv_cmd

class EnvVarPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

plugins.zip 내 중첩 디렉터리 구조 사용 예제

Apache Airflow v2

다음 예제에서는 Apache Airflow v2의 `hooks`, `operators` 및 `sensors` 디렉터리에 별도의 디렉터리가 있는 `plugins.zip` 파일을 보여줍니다.

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
```

다음 예제에서는 사용자 지정 플러그인을 사용하는 DAG(DAGs 폴더)의 가져오기 문을 보여줍니다.

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]
```

다음 예제에서는 사용자 지정 플러그인 파일에 필요한 각 가져오기 문을 보여줍니다.

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
```

Amazon MWAA CLI 유틸리티를 사용하여 사용자 지정 플러그인 테스트를 하고 `plugins` 디렉터리 내의 내용을 압축하는 plugins.zip 파일을 생성하는 단계를 따릅니다. 예: `cd plugins`.

Apache Airflow v1

다음 예제에서는 Apache Airflow v1.10.12의 `hooks`, `operators` 및 `sensors` 디렉터리에 별도의 디렉터리가 있는 `plugins.zip` 파일을 보여줍니다.

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
```

다음 예제에서는 사용자 지정 플러그인을 사용하는 DAG(DAGs 폴더)의 가져오기 문을 보여줍니다.

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_operator import MyOperator
from sensors.my_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}
```

```
with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
  task_id='taskB',
  my_field='some text'
 )

 hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



 sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
from operators.my_airflow_operator import *
from utils.my_utils import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

다음 예제에서는 사용자 지정 플러그인 파일에 필요한 각 가져오기 문을 보여줍니다.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base_hook import BaseHook
```

```
class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensors/my_airflow_sensor.py

```
from airflow.sensors.base_sensor_operator import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example operators/my_airflow_operator.py

```
from airflow.operators.bash_operator import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example operators/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

Amazon MWAA CLI 유틸리티를 사용하여 사용자 지정 플러그인 테스트를 하고 `plugins` 디렉터리 내의 내용을 압축하는 plugins.zip 파일을 생성하는 단계를 따릅니다. 예: `cd plugins`.

plugins.zip 파일 생성

다음 단계에서는 로컬에서 plugins.zip 파일을 생성할 때 권장하는 단계를 설명합니다.

1단계: Amazon MWAA CLI 유틸리티를 사용하여 사용자 지정 플러그인 테스트

- 명령줄 인터페이스(CLI) 유틸리티는 Amazon Managed Workflows for Apache Airflow 환경을 로컬로 복제합니다.
- CLI는 Amazon MWAA 프로덕션 이미지와 유사한 Docker 컨테이너 이미지를 로컬로 구축합니다. 이를 통해 Amazon MWAA에 배포하기 전에 로컬 Apache Airflow 환경을 실행하여 DAG, 사용자 지정 플러그인 및 종속성을 개발하고 테스트할 수 있습니다.
- CLI를 실행하려면 GitHub의 aws-mwaa-local-runner를 참조하십시오.

2단계: plugins.zip 파일 생성

내장된 ZIP 아카이브 유틸리티 또는 기타 ZIP 유틸리티(예: 7zip)를 사용하여.zip 파일을 생성할 수 있습니다.

> **Note**
>
> Windows OS용 내장 zip 유틸리티는 .zip 파일을 생성할 때 하위 폴더를 추가할 수 있습니다. Amazon S3 버킷에 업로드하기 전에 plugins.zip 파일의 내용을 확인하여 추가 디렉터리가 추가되지 않도록 하는 것이 좋습니다.

1. 디렉터리를 로컬 Airflow 플러그인 디렉터리로 변경합니다. 예:

```
myproject$ cd plugins
```

2. 다음 명령을 실행하여 내용에 실행 권한이 있는지 확인합니다(macOS 및 Linux만 해당).

```
plugins$ chmod -R 755 .
```

3. plugins 폴더 내 내용을 압축합니다.

```
plugins$ zip -r plugins.zip .
```

Amazon S3에 **plugins.zip** 업로드

Amazon S3 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon S3 버킷에 plugins.zip 파일을 업로드할 수 있습니다.

AWS CLI 사용

AWS Command Line Interface(AWS CLI)는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI - 버전 2 설치
- AWS CLI - aws configure을 통한 빠른 구성.

AWS CLI를 사용하여 업로드하려면

1. 명령 프롬프트에서 `plugins.zip` 파일이 저장된 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. 다음 명령을 사용하여 Amazon S3 버킷을 모두 나열합니다.

```
aws s3 ls
```

3. 다음 명령을 사용하여 사용자 환경의 Amazon S3 버킷에 있는 파일과 폴더를 나열합니다.

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

4. 다음 명령을 사용해서 사용자 환경의 Amazon S3 버킷에 `plugins.zip` 파일을 업로드합니다.

```
aws s3 cp plugins.zip s3://YOUR_S3_BUCKET_NAME/plugins.zip
```

Amazon S3 콘솔 사용

Amazon S3 콘솔은 Amazon S3 버킷의 리소스를 생성 및 관리할 수 있는 웹 기반 사용자 인터페이스입니다.

Amazon S3 콘솔을 사용하여 업로드하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. S3 창의 DAG 코드에서 S3 버킷 링크를 선택하여 Amazon S3 콘솔에서 스토리지 버킷을 엽니다.
4. 업로드를 선택합니다.
5. 파일 추가를 선택합니다.
6. `plugins.zip`의 로컬 사본을 선택하고 업로드를 선택합니다.

사용자 환경에 사용자 지정 플러그인 설치

이 섹션에서는 plugins.zip 파일의 경로를 지정하고 zip 파일이 업데이트될 때마다 plugins.zip 파일의 버전을 지정하여 Amazon S3 버킷에 업로드한 사용자 지정 플러그인을 설치하는 방법을 설명합니다.

Amazon MWAA 콘솔에서 **plugins.zip**의 경로 지정(처음)

plugins.zip을 처음으로 Amazon S3 버킷에 업로드하는 경우 Amazon MWAA 콘솔에 파일 경로도 지정해야 합니다. 이 단계는 한 번만 완료하면 됩니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. Amazon S3의 DAG 코드 창에서 플러그인 파일 - 옵션 필드 옆의 S3 찾아보기를 선택합니다.
5. Amazon S3 버킷에 있는 plugins.zip 파일을 선택합니다.
6. 선택을 선택합니다.
7. 다음, 환경 업데이트를 선택합니다.

Amazon MWAA 콘솔에서 **plugins.zip** 버전 지정

Amazon S3 버킷에 사용자 plugins.zip의 새 버전을 업로드할 때마다 Amazon MWAA 콘솔에서 plugins.zip 파일의 버전을 지정해야 합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. Amazon S3의 DAG 코드 창의 드롭다운 목록에서 plugins.zip 버전을 선택합니다.
5. 다음(Next)을 선택합니다.

plugins.zip 사용 사례 예제

- Apache Hive 및 Hadoop을 사용한 사용자 지정 플러그인 에서 사용자 지정 플러그인을 생성하는 방법을 알아봅니다.
- PythonVirtualenvOperator를 패치하는 사용자 지정 플러그인 에서 사용자 지정 플러그인을 생성하는 방법을 알아봅니다.
- Oracle을 이용한 사용자 지정 플러그인에서 사용자 지정 플러그인을 생성하는 방법을 알아봅니다.
- the section called “DAG 시간대 변경”에서 사용자 지정 플러그인을 생성하는 방법을 알아봅니다.

다음 단계

- GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.

Python 종속성 설치

Python 종속성이란 Amazon Managed Workflows for Apache Airflow 환경에서 Apache Airflow 버전의 Apache Airflow 기본 설치에 포함되지 않은 모든 패키지 또는 배포를 말합니다. 이 페이지에서는 Amazon S3 버킷의 `requirements.txt` 파일을 사용하여 Amazon MWAA 환경에서 Apache Airflow Python 종속성을 설치하는 단계를 설명합니다.

목차

필수 조건

이 페이지의 단계를 완료하려면 먼저 다음이 필요합니다.

- 권한 — 관리자는 사용자 환경에 대한 AmazonMWAA 액세스 액세스 제어 정책에 FullConsole 대한 액세스 권한을 사용자 AWS 계정에 부여했어야 합니다. 또한 실행 역할이 Amazon MWAA 환경에서 사용 중인 AWS 리소스에 액세스할 수 있도록 허용해야 합니다.
- 액세스 — 종속성을 웹 서버에 직접 설치하기 위해 퍼블릭 리포지토리에 액세스해야 하는 경우 퍼블릭 네트워크 웹 서버 액세스로 환경을 구성해야 합니다. 자세한 내용은 the section called “Apache Airflow 액세스 모드” 단원을 참조하십시오.
- Amazon S3 구성 — DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 저장하는 데 사용되는 Amazon S3 버킷은 퍼블릭 액세스가 차단되고 버전 관리가 활성화된 상태로 구성되어야 합니다.

작동 방식

Amazon S3 버킷에 `requirements.txt` 파일을 업로드한 다음 파일을 업데이트할 때마다 Amazon MWAA 콘솔에서 파일 버전을 지정하여 모든 Python 종속성을 Amazon MWAA에 설치합니다. Apache Airflow 스케줄러와 각 작업자에 Python 종속성을 `pip3 install -r requirements.txt` 설치하기 위해 Amazon MWAA를 실행합니다.

사용자 환경에서 Python 종속성을 실행하려면 다음 세 가지 작업을 수행해야 합니다.

1. 로컬에서 `requirements.txt` 파일을 생성합니다.
2. Amazon S3 버킷에 로컬 `requirements.txt`을 업로드합니다.
3. Amazon MWAA 콘솔의 요구 사항 파일 필드에 이 파일의 버전을 지정합니다.

> **Note**
>
> `requirements.txt`을 처음으로 생성하여 Amazon S3 버킷에 업로드하는 경우 Amazon MWAA 콘솔에 파일 경로도 지정해야 합니다. 이 단계는 한 번만 완료하면 됩니다.

Python 종속성 개요

사용자 환경의 PyPi 사설 /PEP-503 호환 리포지토리에서 호스팅되는 Python 패키지 색인 (PyPi.org), Python wheel `.whl` () 또는 Python 종속 항목에서 아파치 에어플로우 추가 기능 및 기타 Python 종속성을 설치할 수 있습니다.

Python 종속성 위치와 크기 제한

Apache Airflow 스케줄러와 작업자는 시작 중에 사용자 환경에 맞는 -managed AWS Fargate 컨테이너에서 사용자 지정 플러그인을 찾습니다. `/usr/local/airflow/plugins`

- 크기 제한. 전체 크기가 1GB 미만인 라이브러리를 참조하는 `requirements.txt` 파일을 사용하는 것이 좋습니다. Amazon MWAA에 설치해야 하는 라이브러리가 많을수록 환경에서 시작 시간이 길어집니다. Amazon MWAA가 설치된 라이브러리의 크기를 명시적으로 제한하지 않지만, 종속성을 10분 이내에 설치할 수 없는 경우 Fargate 서비스는 시간을 초과하여 환경을 안정적인 상태로 롤백하려고 시도합니다.

requirements.txt 파일 생성

다음 단계에서는 로컬에서 requirements.txt 파일을 생성할 때 권장하는 단계를 설명합니다.

1단계: Amazon MWAA CLI 유틸리티를 사용하여 Python 종속성 테스트

- 명령줄 인터페이스(CLI) 유틸리티는 Amazon Managed Workflows for Apache Airflow 환경을 로컬로 복제합니다.
- CLI는 Amazon MWAA 프로덕션 이미지와 유사한 Docker 컨테이너 이미지를 로컬로 구축합니다. 이를 통해 Amazon MWAA에 배포하기 전에 로컬 Apache Airflow 환경을 실행하여 DAG, 사용자 지정 플러그인 및 종속성을 개발하고 테스트할 수 있습니다.
- CLI를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub

2단계: **`requirements.txt`** 생성

다음 섹션에서는 `requirements.txt` 파일의 Python 패키지 인덱스에서 Python 종속성을 지정하는 방법을 설명합니다.

Apache Airflow v2

1. 로컬에서 테스트. `requirements.txt` 파일을 생성하기 전에 라이브러리를 반복적으로 추가하여 패키지와 버전의 적절한 조합을 찾습니다. Amazon MWAA CLI 유틸리티를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub
2. Apache Airflow 패키지 엑스트라를 검토합니다. Amazon MWAA에서 아파치 에어플로우 v2용으로 설치된 패키지 목록을 보려면 웹 사이트의 Amazon MWAA 로컬 러너를 참조하십시오. `requirements.txt` GitHub

3. 제약 조건 문을 추가합니다. `requirements.txt` 파일 상단에 Apache Airflow v2 환경에 대한 제약 조건 파일을 추가합니다. Apache Airflow 제약 조건 파일은 Apache Airflow 릴리스 당시 사용할 수 있는 공급자 버전을 지정합니다.

   Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.

   다음 예제에서 *{environment-version} # ### ### ##* 번호로 바꾸고, *{Python-version}* 을 사용자 환경과 호환되는 Python 버전으로 바꾸십시오.

   Apache Airflow 환경과 호환되는 Python 버전에 대한 자세한 내용은 Apache Airflow 버전을 참조하십시오.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   제약 조건 파일에서 `xyz==1.0` 패키지가 사용자 환경의 다른 패키지와 호환되지 않는 것으로 확인되면 `pip3 install`은 호환되지 않는 라이브러리가 환경에 설치되는 것을 막을 수 없습니다. 패키지 설치가 실패하는 경우 로그의 해당 로그 스트림에서 각 Apache Airflow 구성 요소(스케줄러, 작업자, 웹 서버) 에 대한 오류 로그를 볼 수 있습니다. CloudWatch 로그 형식에 대한 자세한 내용은 the section called “Airflow 로그 확인” 단원을 참조하십시오.

4. Apache Airflow 패키지 패키지 엑스트라 및 버전(==)을 추가합니다. 이렇게 하면 이름은 같지만 버전이 다른 패키지가 사용자 환경에 설치되는 것을 방지할 수 있습니다.

   ```
   apache-airflow[package-extra]==2.5.1
   ```

5. Python 라이브러리. `requirements.txt` 파일에 패키지 이름과 버전(==)을 추가합니다. 이렇게 하면 PyPi.org의 향후 주요 업데이트가 자동으로 적용되는 것을 방지할 수 있습니다.

   ```
   library == version
   ```

   Example Boto3 및 psycopg2-binary

   이 예제는 데모용으로 제공됩니다. boto 및 psycopg2-binary 라이브러리는 Apache Airflow v2 기본 설치에 포함되어 있으며 `requirements.txt` 파일에서 지정할 필요가 없습니다.

   ```
   boto3==1.17.54
   boto==2.49.0
   ```

```
botocore==1.20.54
psycopg2-binary==2.8.6
```

버전 없이 패키지를 지정한 경우 Amazon MWAA는.org에서 패키지의 최신 버전을 설치합니다. PyPi 이 버전은 `requirements.txt`에 있는 다른 패키지와 충돌할 수 있습니다.

Apache Airflow v1

1. 로컬에서 테스트. `requirements.txt` 파일을 생성하기 전에 라이브러리를 반복적으로 추가하여 패키지와 버전의 적절한 조합을 찾습니다. Amazon MWAA CLI 유틸리티를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub
2. Airflow 패키지 추가 내용을 검토합니다. https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt에서 Apache Airflow v1.10.12에 사용할 수 있는 패키지 목록을 검토합니다.
3. 제약조건 파일 추가 `requirements.txt` 파일 상단에 Apache Airflow v1.10.12의 제약 조건 파일을 추가합니다. 제약 조건 파일에서 `xyz==1.0` 패키지가 사용자 환경의 다른 패키지와 호환되지 않는 것으로 확인되면 `pip3 install`은 호환되지 않는 라이브러리가 환경에 설치되는 것을 막을 수 없습니다.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-1.10.12/constraints-3.7.txt"
   ```
4. Apache Airflow v1.10.12 패키지. Airflow 패키지 엑스트라 및 Apache Airflow v1.10.12 버전(==)을 추가합니다. 이렇게 하면 이름은 같지만 버전이 다른 패키지가 사용자 환경에 설치되는 것을 방지할 수 있습니다.

   ```
   apache-airflow[package]==1.10.12
   ```

   Example Secure Shell(SSH)

   다음 예제 `requirements.txt` 파일은 Apache Airflow v1.10.12용 SSH를 설치합니다.

   ```
   apache-airflow[ssh]==1.10.12
   ```
5. Python 라이브러리. `requirements.txt` 파일에 패키지 이름과 버전(==)을 추가합니다. 이렇게 하면 PyPi.org의 향후 주요 업데이트가 자동으로 적용되는 것을 방지할 수 있습니다.

```
library == version
```

Example Boto3

다음 예제 `requirements.txt` 파일은 Apache Airflow v1.10.12용 Boto3 라이브러리를 설치합니다.

```
boto3 == 1.17.4
```

버전 없이 패키지를 지정한 경우 Amazon MWAA는.org에서 패키지의 최신 버전을 설치합니다. PyPi 이 버전은 `requirements.txt`에 있는 다른 패키지와 충돌할 수 있습니다.

Amazon S3에 `requirements.txt` 업로드

Amazon S3 콘솔 또는 AWS Command Line Interface (AWS CLI) 를 사용하여 Amazon S3 버킷에 `requirements.txt` 파일을 업로드할 수 있습니다.

사용: AWS CLI

AWS Command Line Interface (AWS CLI) 는 명령줄 셸의 명령을 사용하여 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 이 페이지에서 단계를 완료하려면 다음이 필요합니다.

- AWS CLI — 버전 2를 설치합니다.
- AWS CLI — 빠른 구성 `aws configure`.

를 사용하여 업로드하려면 AWS CLI

1. 다음 명령을 사용하여 Amazon S3 버킷을 모두 나열합니다.

   ```
   aws s3 ls
   ```

2. 다음 명령을 사용하여 사용자 환경의 Amazon S3 버킷에 있는 파일과 폴더를 나열합니다.

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

3. 다음 명령은 Amazon S3 버킷에 `requirements.txt` 파일을 업로드합니다.

```
aws s3 cp requirements.txt s3://YOUR_S3_BUCKET_NAME/requirements.txt
```

Amazon S3 콘솔 사용

Amazon S3 콘솔은 Amazon S3 버킷의 리소스를 생성 및 관리할 수 있는 웹 기반 사용자 인터페이스입니다.

Amazon S3 콘솔을 사용하여 업로드하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. S3 창의 DAG 코드에서 S3 버킷 링크를 선택하여 Amazon S3 콘솔에서 스토리지 버킷을 엽니다.
4. 업로드를 선택합니다.
5. 파일 추가를 선택합니다.
6. `requirements.txt`의 로컬 사본을 선택하고 업로드를 선택합니다.

사용자 환경에 Python 종속성 설치

이 섹션에서는 requirements.txt 파일의 경로를 지정하고 업데이트될 때마다 requirements.txt 파일의 버전을 지정하여 Amazon S3 버킷에 업로드한 종속성을 설치하는 방법을 설명합니다.

Amazon MWAA 콘솔에서 **`requirements.txt`**의 경로 지정(처음)

`requirements.txt`을 처음으로 생성하여 Amazon S3 버킷에 업로드하는 경우 Amazon MWAA 콘솔에 파일 경로도 지정해야 합니다. 이 단계는 한 번만 완료하면 됩니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. Amazon S3의 DAG 코드 창에서 요구 사항 파일 - 옵션 필드 옆의 S3 찾아보기를 선택합니다.
5. Amazon S3 버킷에 있는 `requirements.txt` 파일을 선택합니다.
6. 선택을 선택합니다.
7. 다음, 환경 업데이트를 선택합니다.

환경 업데이트가 완료된 후 즉시 새 패키지 사용을 시작할 수 있습니다.

Amazon MWAA 콘솔에서 **requirements.txt** 버전 지정

Amazon S3 버킷에 사용자 requirements.txt의 새 버전을 업로드할 때마다 Amazon MWAA 콘솔에서 requirements.txt 파일의 버전을 지정해야 합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. Amazon S3의 DAG 코드 창의 드롭다운 목록에서 requirements.txt 버전을 선택합니다.
5. 다음, 환경 업데이트를 선택합니다.

환경 업데이트가 완료된 후 즉시 새 패키지 사용을 시작할 수 있습니다.

사용자 **requirements.txt**의 로그 보기

워크플로우를 예약하고 dags 폴더를 구문 분석하는 스케줄러에 대한 Apache Airflow 로그를 볼 수 있습니다. 다음 단계는 Amazon MWAA 콘솔에서 스케줄러의 로그 그룹을 여는 방법과 로그 콘솔에서 Apache Airflow 로그를 보는 방법을 설명합니다. CloudWatch

requirements.txt에 대한 로그를 보려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 모니터링 창에서 Airflow 스케줄러 로그 그룹을 선택합니다.
4. 로그 스트림에서 requirements_install_ip 로그를 선택합니다.
5. /usr/local/airflow/.local/bin에서 환경에 설치된 패키지 목록을 볼 수 있습니다. 예:

```
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. 패키지 목록을 검토하고 설치 중에 오류가 발생했는지 여부를 검토합니다. 문제가 발생한 경우, 다음과 비슷한 오류가 표시될 수 있습니다.

```
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))
```

다음 단계

- aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인 및 Python 종속성을 로컬에서 테스트하십시오. GitHub

Amazon S3에서 파일 삭제

이 페이지에서는 Amazon Managed Workflows for Apache Airflow 환경의 Amazon S3 버킷에서 버전 관리가 작동하는 방식과 DAG, `plugins.zip` 또는 `requirements.txt` 파일을 삭제하는 단계를 설명합니다.

목차

필수 조건

이 페이지의 단계를 완료하려면 먼저 다음이 필요합니다.

- 권한 — 관리자가 사용자 환경의 AmazonMWAAFullConsoleAccess 액세스 제어 정책에 대한 액세스 권한을 AWS 계정에 부여했어야 합니다. 또한 실행 역할이 Amazon MWAA 환경에서 사용자 환경에서 사용하는 AWS 리소스에 액세스할 수 있도록 허용해야 합니다.
- 액세스 — 종속성을 웹 서버에 직접 설치하기 위해 퍼블릭 리포지토리에 액세스해야 하는 경우 퍼블릭 네트워크 웹 서버 액세스로 환경을 구성해야 합니다. 자세한 내용은 the section called “Apache Airflow 액세스 모드” 단원을 참조하십시오.
- Amazon S3 구성 — DAG, `plugins.zip`의 사용자 지정 플러그인 및 `requirements.txt`의 Python 종속성을 저장하는 데 사용되는 Amazon S3 버킷은 퍼블릭 액세스가 차단되고 버전 관리가 활성화된 상태로 구성되어야 합니다.

버전 관리 개요

Amazon S3 버킷의 `requirements.txt` 및 `plugins.zip`에는 버전이 지정되어 있습니다. 객체에 대해 Amazon S3 버킷 버전 관리를 활성화하고 Amazon S3 버킷에서 아티팩트(예: plugins.zip)를 삭제해도 파일이 완전히 삭제되지는 않습니다. Amazon S3에서 아티팩트가 삭제될 때마다 “I'm not here.”라는 404(객체를 찾을 수 없음) 오류/0k 파일인 새 파일 사본이 생성됩니다. Amazon S3에서는 이를 삭제 마커라고 합니다. 삭제 마커는 다른 객체들과 마찬가지로 키 이름(또는 키)과 버전 ID를 가진 파일의 "null" 버전입니다.

Amazon S3 버킷의 스토리지 비용을 줄이려면 파일 버전과 삭제 마커를 주기적으로 삭제하는 것이 좋습니다. “최신 버전이 아닌" (이전) 파일 버전을 완전히 삭제하려면 파일 버전을 삭제한 다음 해당 버전의 삭제 마커를 삭제해야 합니다.

작동 방식

Amazon MWAA는 30초마다 Amazon S3 버킷에서 동기화 작업을 실행합니다. 그러면 Amazon S3 버킷의 모든 DAG 삭제가 Fargate 컨테이너의 에어플로 이미지에 동기화됩니다.

`plugins.zip` 및 `requirements.txt` 파일의 경우 Amazon MWAA가 사용자 지정 플러그인 및 Python 종속성을 사용하여 Fargate 컨테이너의 새 Airflow 이미지를 빌드할 때 환경 업데이트 이후에만 변경이 발생합니다. `requirements.txt` 또는 `plugins.zip` 파일의 현재 버전을 삭제한 다음 삭제된 파일에 대한 새 버전을 제공하지 않고 환경을 업데이트하면 업데이트가 실패하고 “`{file}` 파일의 버전 `{version}`을 읽을 수 없습니다”와 같은 오류 메시지가 표시됩니다.

Amazon S3에서 DAG 삭제

DAG 파일(`.py`)은 버전이 지정되지 않으며 Amazon S3 콘솔에서 직접 삭제할 수 있습니다. 다음 단계는 Amazon S3 버킷에서 DAG를 삭제하는 방법을 설명합니다.

DAG를 삭제하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. S3 창의 DAG 코드에서 S3 버킷 링크를 선택하여 Amazon S3 콘솔에서 스토리지 버킷을 엽니다.
4. dags 폴더를 선택합니다.
5. DAG를 선택하고 삭제를 선택합니다.
6. 객체를 삭제하시겠습니까?에서 delete를 입력합니다.
7. 객체 삭제를 선택합니다.

> **Note**
>
> Apache Airflow는 과거 DAG 실행을 보존합니다. Apache Airflow에서 DAG를 실행한 후에는 Apache Airflow에서 DAG를 삭제할 때까지 파일 상태에 관계없이 Airflow DAG 목록에 남아 있습니다. Apache Airflow에서 DAG를 삭제하려면 링크 열에서 빨간색 “삭제” 버튼을 선택합니다.

환경에서 “현재” requirements.txt 또는 plugins.zip 제거

현재 plugins.zip 또는 requirements.txt 를 추가한 후 해당 환경에서 제거할 수 있는 방법은 없지만, 현재 해결 중입니다. 그 동안 해결 방법은 빈 텍스트 또는 zip 파일을 각각 가리키는 것입니다.

“최신이 아닌" (이전) requirements.txt 또는 plugins.zip 버전 삭제

Amazon S3 버킷의 requirements.txt 및 plugins.zip 파일은 Amazon MWAA에서 버전이 관리됩니다. Amazon S3 버킷에서 이러한 파일을 완전히 삭제하려면 객체의 현재 버전(121212)(예: plugins.zip)을 검색하고 버전을 삭제한 다음 파일 버전의 삭제 마커를 제거해야 합니다.

Amazon S3 콘솔에서 “최신이 아닌” (이전) 파일 버전을 삭제할 수도 있지만, 그래도 다음 옵션 중 하나를 사용하여 삭제 마커를 삭제해야 합니다.

- 객체 버전을 검색하려면 Amazon S3 가이드의 버전 관리를 사용하는 버킷에서 객체 버전 검색을 참조하십시오.
- 객체 버전을 삭제하려면 Amazon S3 가이드의 버전 관리를 사용하는 버킷에서 객체 버전 삭제를 참조하십시오.
- 삭제 마커를 제거하려면 Amazon S3 가이드의 삭제 마커 관리를 참조하십시오.

수명 주기를 사용하여 “최신이 아닌" (이전) 버전을 삭제하고 마커를 자동으로 삭제합니다.

특정 일수가 지나면 Amazon S3 버킷에서 plugins.zip 및 requirements.txt 파일의 “최신이 아닌” (이전) 버전을 삭제하거나 만료된 객체의 삭제 마커를 제거하도록 Amazon S3 버킷에 대한 수명 주기 정책을 구성할 수 있습니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. Amazon S3의 DAG 코드에서 Amazon S3 버킷을 선택합니다.
4. 수명 주기 규칙 생성을 선택합니다.

requirements.txt “최신이 아닌” 버전을 삭제하고 마커를 자동으로 삭제하는 수명 주기 정책 예시

다음 예제는 requirements.txt 파일의 “최신이 아닌” 버전과 30일 후 삭제 마커를 영구 삭제하는 수명 주기 규칙을 생성하는 방법을 보여줍니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. Amazon S3의 DAG 코드에서 Amazon S3 버킷을 선택합니다.
4. 수명 주기 규칙 생성을 선택합니다.
5. 수명 주기 규칙 이름에 `Delete previous requirements.txt versions and delete markers after thirty days`를 입력합니다.
6. 접두사, 요구 사항에서.
7. 수명 주기 규칙 작업에서 이전 버전의 객체 영구 삭제 및 만료된 삭제 마커 또는 미완료 멀티파트 업로드 삭제를 선택합니다.
8. 객체가 이전 버전이 된 후 경과한 일수에 `30`을 입력합니다.
9. 만료된 객체 삭제 마커에서 만료된 객체 삭제 마커 삭제를 선택하면 30일 후에 객체가 영구적으로 삭제됩니다.

다음 단계

- 삭제 마커 관리에서 Amazon S3 삭제 마커에 대해 자세히 알아봅니다.

- Amazon S3 수명 주기에 대해 자세히 알아보려면 객체 만료를 참조하십시오.

네트워킹

이 가이드에서는 Amazon MWAA 환경에 필요한 Amazon VPC 네트워크 설정에 대해 설명합니다.

Sections

Amazon MWAA에서의 네트워킹에 대해

Amazon VPC는 사용자 AWS 계정에 연결된 가상 네트워크입니다. 가상 인프라 및 네트워크 트래픽 세분화를 세밀하게 제어하여 클라우드 보안과 동적으로 확장할 수 있는 기능을 제공합니다. 이 페이지에서는 Amazon Managed Workflows for Apache Airflow 환경을 지원하는 데 필요한 퍼블릭 라우팅 또는 프라이빗 라우팅이 있는 Amazon VPC 인프라를 설명합니다.

목차

용어

퍼블릭 라우팅

인터넷에 액세스할 수 있는 Amazon VPC 네트워크.

프라이빗 라우팅

인터넷에 액세스할 수 없는 Amazon VPC 네트워크입니다.

지원되는 항목

다음 표에서는 Amazon MWAA가 지원하는 Amazon VPC 유형에 대해 설명합니다.

Amazon VPC 유형	지원	
환경을 생성하려는 계정이 소유한 Amazon VPC입니다.	예	
여러 AWS 계정이 리소스를 생성하는 공유 Amazon VPC입니다. AWS	예	

VPC 인프라 개요

Amazon MWAA 환경을 생성할 때 Amazon MWAA는 환경에 맞게 선택한 Apache Airflow 액세스 모드를 기반으로 환경에 대해 1~2개의 VPC 엔드포인트를 생성합니다. 이러한 엔드포인트는 Amazon VPC에 프라이빗 IP가 있는 탄력적 네트워크 인터페이스(ENI)로 표시됩니다. 이러한 엔드포인트가 생성되면 해당 IP로 향하는 모든 트래픽이 사용자 환경에서 사용되는 해당 AWS 서비스로 비공개 또는 공개적으로 라우팅됩니다.

다음 섹션에서는 인터넷을 통해 트래픽을 공개적으로 라우팅하거나 Amazon VPC 내에서 비공개로 라우팅하는 데 필요한 Amazon VPC 인프라에 대해 설명합니다.

인터넷을 통한 퍼블릭 라우팅

이 섹션에서는 퍼블릭 라우팅을 사용하는 환경의 Amazon VPC 인프라에 대해 설명합니다. 다음과 같은 VPC 인프라가 필요합니다.

- 하나의 VPC 보안 그룹. VPC 보안 그룹은 인스턴스의 수신(인바운드) 및 송신(아웃바운드) 네트워크 트래픽을 제어하는 가상 방화벽 역할을 합니다.
  - 최대 5개의 보안 그룹을 지정할 수 있습니다.
  - 보안 그룹은 자체 참조 인바운드 규칙을 지정해야 합니다.
  - 보안 그룹은 모든 트래픽에 대한 아웃바운드 규칙을 지정해야 합니다(`0.0.0.0/0`).
  - 보안 그룹은 자체 참조 규칙의 모든 트래픽을 허용해야 합니다. 예를 들어 (권장) 모든 액세스 자체 참조 보안 그룹의 예제 입니다.
  - 보안 그룹은 선택적으로 HTTPS 포트 범위 443 및 TCP 포트 범위 5432에 대한 포트 범위를 지정하여 트래픽을 추가로 제한할 수도 있습니다. 예: (선택 사항) 포트 5432에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제 및 (선택 사항) 포트 443에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제.
- 2개의 퍼블릭 서브넷. 퍼블릭 서브넷은 인터넷 게이트웨이로 이어지는 경로가 있는 라우팅 테이블과 연결된 서브넷입니다.
  - 2개의 퍼블릭 서브넷이 필요합니다. 이를 통해 Amazon MWAA는 하나의 컨테이너가 실패하는 경우 다른 가용 영역에서 사용자 환경에 대한 새 컨테이너 이미지를 구축할 수 있습니다.
  - 서브넷이 서로 다른 가용 영역에 있어야 합니다. 예, `us-east-1a`, `us-east-1b`.
  - 서브넷은 탄력적 IP 주소(EIP)가 있는 NAT 게이트웨이(또는 NAT 인스턴스)로 라우팅해야 합니다.
  - 서브넷에는 인터넷 바인딩 트래픽을 인터넷 게이트웨이로 전달하는 라우팅 테이블이 있어야 합니다.
- 2개의 프라이빗 서브넷. 프라이빗 서브넷은 인터넷 게이트웨이에 대한 경로가 있는 라우팅 테이블과 연결되지 않은 서브넷입니다.
  - 두 개의 프라이빗 서브넷이 필요합니다. 이를 통해 Amazon MWAA는 하나의 컨테이너가 실패하는 경우 다른 가용 영역에서 사용자 환경에 대한 새 컨테이너 이미지를 구축할 수 있습니다.
  - 서브넷이 서로 다른 가용 영역에 있어야 합니다. 예, `us-east-1a`, `us-east-1b`.
  - 서브넷에 NAT 장치(게이트웨이 또는 인스턴스)에 대한 라우팅 테이블이 있어야 합니다.
  - 서브넷은 인터넷 게이트웨이로 라우팅되어서는 안 됩니다.
- 네트워크 액세스 제어 목록(ACL). NACL은 허용 또는 거부 규칙을 통해 서브넷 수준에서 인바운드 및 아웃바운드 트래픽을 관리합니다.
  - NACL에는 모든 트래픽을 허용하는 인바운드 규칙이 있어야 합니다(`0.0.0.0/0`).
  - NACL에는 모든 트래픽을 거부하는 아웃바운드 규칙이 있어야 합니다(`0.0.0.0/0`).
  - 예를 들어 (권장) 예제 ACL입니다.

- 2개의 NAT 게이트웨이(또는 NAT 인스턴스). NAT 디바이스는 프라이빗 서브넷의 인스턴스에서 인터넷 또는 기타 AWS 서비스로 트래픽을 전달한 다음 응답을 다시 인스턴스로 라우팅합니다.
  - NAT 장치는 퍼블릭 서브넷에 연결되어야 합니다. (퍼블릭 서브넷당 하나의 NAT 장치.)
  - NAT 장치에는 각 퍼블릭 서브넷에 연결된 EIP(탄력적 IPv4 주소)가 있어야 합니다.
- 인터넷 게이트웨이. 인터넷 게이트웨이는 Amazon VPC를 인터넷 및 기타 AWS 서비스에 연결합니다.
  - 인터넷 게이트웨이는 Amazon VPC에 연결되어야 합니다.

인터넷 접속이 필요 없는 프라이빗 라우팅

이 섹션에서는 프라이빗 라우팅을 사용하는 환경의 Amazon VPC 인프라에 대해 설명합니다. 다음과 같은 VPC 인프라가 필요합니다.

- 하나의 VPC 보안 그룹. VPC 보안 그룹은 인스턴스의 수신(인바운드) 및 송신(아웃바운드) 네트워크 트래픽을 제어하는 가상 방화벽 역할을 합니다.
  - 최대 5개의 보안 그룹을 지정할 수 있습니다.
  - 보안 그룹은 자체 참조 인바운드 규칙을 지정해야 합니다.
  - 보안 그룹은 모든 트래픽에 대한 아웃바운드 규칙을 지정해야 합니다(`0.0.0.0/0`).
  - 보안 그룹은 자체 참조 규칙의 모든 트래픽을 허용해야 합니다. 예를 들어 (권장) 모든 액세스 자체 참조 보안 그룹의 예제 입니다.
  - 보안 그룹은 선택적으로 HTTPS 포트 범위 443 및 TCP 포트 범위 5432에 대한 포트 범위를 지정하여 트래픽을 추가로 제한할 수도 있습니다. 예: (선택 사항) 포트 5432에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제 및 (선택 사항) 포트 443에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제.
- 2개의 프라이빗 서브넷. 프라이빗 서브넷은 인터넷 게이트웨이에 대한 경로가 있는 라우팅 테이블과 연결되지 않은 서브넷입니다.
  - 두 개의 프라이빗 서브넷이 필요합니다. 이를 통해 Amazon MWAA는 하나의 컨테이너가 실패하는 경우 다른 가용 영역에서 사용자 환경에 대한 새 컨테이너 이미지를 구축할 수 있습니다.
  - 서브넷이 서로 다른 가용 영역에 있어야 합니다. 예, `us-east-1a`, `us-east-1b`.
  - 서브넷에는 VPC 엔드포인트에 대한 라우팅 테이블이 있어야 합니다.
  - 서브넷에 NAT 장치(게이트웨이 또는 인스턴스)에 대한 라우팅 테이블이나 인터넷 게이트웨이가 있어서는 안 됩니다.
- 네트워크 액세스 제어 목록(ACL). NACL은 허용 또는 거부 규칙을 통해 서브넷 수준에서 인바운드 및 아웃바운드 트래픽을 관리합니다.

  - NACL에는 모든 트래픽을 허용하는 인바운드 규칙이 있어야 합니다(`0.0.0.0/0`).
  - NACL에는 모든 트래픽을 거부하는 아웃바운드 규칙이 있어야 합니다(`0.0.0.0/0`).
  - 예를 들어 (권장) 예제 ACL입니다.
- 로컬 라우팅 테이블. 로컬 라우팅 테이블은 VPC 내 통신을 위한 기본 경로입니다.
  - 로컬 라우팅 테이블은 프라이빗 서브넷에 연결되어야 합니다.
  - 로컬 라우팅 테이블은 VPC의 인스턴스가 자체 네트워크와 통신할 수 있도록 해야 합니다. 예를 들어 를 사용하여 Apache Airflow 웹 서버의 VPC 인터페이스 엔드포인트에 액세스하는 경우 라우팅 테이블은 VPC 엔드포인트로 라우팅되어야 합니다. AWS Client VPN
- 사용자 환경에서 사용하는 각 AWS 서비스의 VPC 엔드포인트 및 Amazon MWAA 환경과 동일한 AWS 지역 및 Amazon VPC에 있는 Apache Airflow VPC 엔드포인트.
  - 환경에서 사용하는 각 AWS 서비스의 VPC 엔드포인트와 Apache Airflow용 VPC 엔드포인트입니다. 예를 들어 (필수) VPC 엔드포인트입니다.
  - VPC 엔드포인트에는 프라이빗 DNS가 활성화되어 있어야 합니다.
  - VPC 엔드포인트는 사용자 환경의 두 개의 프라이빗 서브넷에 연결되어야 합니다.
  - VPC 엔드포인트는 사용자 환경의 보안 그룹에 연결되어야 합니다.
  - 각 엔드포인트에 대한 VPC 엔드포인트 정책은 환경에서 사용하는 AWS 서비스에 대한 액세스를 허용하도록 구성되어야 합니다. 예를 들어 (권장) 모든 액세스를 허용하는 VPC 엔드포인트 정책 예제입니다.
  - Amazon S3에 대한 VPC 엔드포인트 정책은 버킷 액세스를 허용하도록 구성되어야 합니다. 예를 들어 (권장) 버킷 액세스를 허용하는 Amazon S3 게이트웨이 엔드포인트 정책의 예제입니다.

Amazon VPC 및 Apache Airflow 액세스 모드의 사용 사례 예시

이 섹션에서는 Amazon VPC의 네트워크 액세스에 대한 다양한 사용 사례와 Amazon MWAA 콘솔에서 선택해야 하는 Apache Airflow 웹 서버 액세스 모드를 설명합니다.

인터넷 액세스 허용 - 새로운 Amazon VPC 네트워크

조직에서 VPC의 인터넷 액세스를 허용하고 사용자가 인터넷을 통해 Apache Airflow 웹 서버에 액세스하도록 하려는 경우:

1. 인터넷에 액세스할 수 있는 Amazon VPC 네트워크를 생성합니다.
2. Apache Airflow 웹 서버에 대한 퍼블릭 네트워크 액세스 모드를 사용하여 환경을 생성합니다.

3. 권장 사항: Amazon VPC 인프라, Amazon S3 버킷 및 Amazon MWAA 환경을 동시에 생성하는 AWS CloudFormation 퀵 스타트 템플릿을 사용하는 것이 좋습니다. 자세한 내용은 Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼 단원을 참조하십시오.

조직에서 VPC의 인터넷 액세스를 허용하고 Apache Airflow 웹 서버 액세스를 VPC 내 사용자로 제한하려는 경우:

1. 인터넷에 액세스할 수 있는 Amazon VPC 네트워크를 생성합니다.
2. 컴퓨터에서 Apache Airflow 웹 서버의 VPC 인터페이스 엔드포인트에 액세스하는 메커니즘을 생성합니다.
3. Apache Airflow 웹 서버에 대한 프라이빗 네트워크 액세스 모드를 사용하여 환경을 생성합니다.
4. 권장 사항:
    a. 에서 Amazon MWAA 콘솔을 사용하거나 에서 옵션 1: Amazon MWAA 콘솔에서 VPC 네트워크 생성 AWS CloudFormation 템플릿을 사용하는 것이 좋습니다. 옵션 2: 인터넷에 액세스할 수 있는 Amazon VPC 네트워크 생성
    b. 에서 Apache Airflow 웹 AWS Client VPN 서버에 대한 액세스를 사용하여 구성하는 것이 좋습니다. 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성

인터넷 액세스가 허용되지 않음 - 새 Amazon VPC 네트워크

조직에서 VPC의 인터넷 액세스를 허용하지 않는 경우:

1. 인터넷 액세스 없이 Amazon VPC 네트워크를 생성합니다.
2. 컴퓨터에서 Apache Airflow 웹 서버의 VPC 인터페이스 엔드포인트에 액세스하는 메커니즘을 생성합니다.
3. 환경에서 사용하는 각 AWS 서비스에 대한 VPC 엔드포인트를 생성합니다.
4. Apache Airflow 웹 서버에 대한 프라이빗 네트워크 액세스 모드를 사용하여 환경을 생성합니다.
5. 권장 사항:
    a. AWS CloudFormation 템플릿을 사용하여 인터넷에 액세스할 수 없는 Amazon VPC를 생성하고 Amazon MWAA에서 AWS 사용하는 각 서비스에 대한 VPC 엔드포인트를 생성하는 것이 좋습니다. 옵션 3: 인터넷에 액세스할 수 없는 Amazon VPC 네트워크 생성
    b. 에서 Apache Airflow 웹 서버에 대한 액세스를 사용하여 구성하는 AWS Client VPN 것이 좋습니다. 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성

인터넷 액세스가 허용되지 않음 - 기존 Amazon VPC 네트워크

조직에서 VPC에서의 인터넷 액세스를 허용하지 않고 인터넷 액세스 없이 필요한 Amazon VPC 네트워크가 이미 있는 경우:

1. 환경에서 사용하는 각 AWS 서비스에 대한 VPC 엔드포인트를 생성합니다.
2. Apache Airflow에 대한 VPC 엔드포인트를 생성합니다.
3. 컴퓨터에서 Apache Airflow 웹 서버의 VPC 인터페이스 엔드포인트에 액세스하는 메커니즘을 생성합니다.
4. Apache Airflow 웹 서버에 대한 프라이빗 네트워크 액세스 모드를 사용하여 환경을 생성합니다.
5. 권장 사항:

    a. Amazon MWAA에서 사용하는 AWS 각 서비스에 필요한 VPC 엔드포인트와 Apache Airflow에 필요한 VPC 엔드포인트를 생성하여 연결하는 것이 좋습니다. 프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 서비스 엔드포인트 생성

    b. 에서 Apache Airflow 웹 서버에 대한 액세스를 사용하여 구성하는 것이 좋습니다. AWS Client VPN 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성

Amazon MWAA에서 VPC 보안

이 페이지에서는 Amazon Managed Workflows for Apache Airflow 환경을 보호하는 데 사용되는 Amazon VPC 구성 요소와 이러한 구성 요소에 필요한 구성을 설명합니다.

목차

- (권장) 버킷 액세스를 허용하는 Amazon S3 게이트웨이 엔드포인트 정책의 예제

용어

퍼블릭 라우팅

인터넷에 액세스할 수 있는 Amazon VPC 네트워크.

프라이빗 라우팅

인터넷에 액세스할 수 없는 Amazon VPC 네트워크.

보안 개요

보안 그룹과 액세스 제어 목록(ACL) 은 지정한 규칙을 사용하여 Amazon VPC의 서브넷 및 인스턴스 전반에서 네트워크 트래픽을 제어하는 방법을 제공합니다.

- 액세스 제어 목록(ACL)으로 서브넷을 오가는 네트워크 트래픽을 제어할 수 있습니다. ACL은 하나만 필요하며 여러 환경에서 동일한 ACL을 사용할 수 있습니다.
- Amazon VPC 보안 그룹은 인스턴스로 들어오고 나가는 네트워크 트래픽을 제어할 수 있습니다. 환경당 1~5개의 보안 그룹을 사용할 수 있습니다.
- 인스턴스로 들어오고 나가는 네트워크 트래픽은 VPC 엔드포인트 정책으로 제어할 수도 있습니다. 조직에서 Amazon VPC 내의 인터넷 액세스를 허용하지 않고 Amazon VPC 네트워크를 프라이빗 라우팅과 함께 사용하는 경우, AWS VPC 엔드포인트 및 Apache Airflow VPC 엔드포인트에 대한 VPC 엔드포인트 정책이 필요합니다.

네트워크 액세스 제어 목록(ACL)

네트워크 액세스 제어 목록(ACL) 은 인바운드 트래픽과 아웃바운드 트래픽을 서브넷 수준에서 (허용 또는 거부 규칙에 따라) 관리할 수 있습니다. ACL은 상태 비저장식이므로 인바운드 규칙과 아웃바운드 규칙을 별도로 명시적으로 지정해야 합니다. VPC 네트워크의 인스턴스에서 들어오거나 나가는 네트워크 트래픽의 유형을 지정하는 데 사용됩니다.

모든 Amazon VPC에는 인바운드와 아웃바운드 트래픽을 모두 허용하는 기본 ACL이 있습니다. 기본 ACL 규칙을 편집하거나 사용자 지정 ACL을 생성하여 서브넷에 연결할 수 있습니다. 서브넷에는 한 번에 하나의 ACL만 연결할 수 있지만 하나의 ACL을 여러 서브넷에 연결할 수 있습니다.

(권장) 예제 ACL

다음 예제는 퍼블릭 라우팅 또는 프라이빗 라우팅을 사용하는 Amazon VPC용 Amazon VPC에 사용할 수 있는 인바운드 및 아웃바운드 ACL 규칙을 보여줍니다.

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
100	모든 IPv4 트래픽	모두	모두	0.0.0.0/0	허용
*	모든 IPv4 트래픽	모두	모두	0.0.0.0/0	거부

VPC 보안 그룹

VPC보안 그룹은 인스턴스 레벨에서 네트워크 트래픽을 제어하는 가상 방화벽 역할을 합니다. 보안 그룹은 스테이트풀(stateful) 그룹입니다. 즉, 인바운드 연결이 허용되면 응답할 수 있습니다. VPC 네트워크의 인스턴스에서 들어오는 네트워크 트래픽의 유형을 지정하는 데 사용됩니다.

모든 Amazon VPC에는 기본 보안 그룹이 있습니다. 기본적으로 인바운드 규칙이 없습니다. 모든 아웃바운드 트래픽을 허용하는 아웃바운드 규칙이 있습니다. 기본 보안 그룹 규칙을 편집하거나 사용자 지정 보안 그룹을 생성하여 Amazon VPC에 연결할 수 있습니다. Amazon MWAA에서는 트래픽을 NAT 게이트웨이로 전달하도록 인바운드 및 아웃바운드 규칙을 구성해야 합니다.

(권장) 모든 액세스 자체 참조 보안 그룹의 예제

다음 예제는 퍼블릭 라우팅 또는 프라이빗 라우팅을 사용하는 Amazon VPC의 Amazon VPC에 대한 모든 트래픽을 허용하는 인바운드 보안 그룹 규칙을 보여줍니다. 이 예제의 보안 그룹은 그 자체에 대한 자체 참조 규칙입니다.

유형	프로토콜	소스 유형	소스		
모든 트래픽	모두	모두	sg-0909e8e81919/-그룹 my-mwaa-vpc-security		

다음 예제는 아웃바운드 보안 그룹 규칙을 보여줍니다.

유형	프로토콜	소스 유형	소스		
모든 트래픽	모두	모두	0.0.0.0/0		

(선택 사항) 포트 5432에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제

다음 예제는 사용자 환경의 Amazon Aurora PostgreSQL 메타데이터 데이터베이스(Amazon MWAA 소유)의 포트 5432에서 모든 HTTPS 트래픽을 허용하는 인바운드 보안 그룹 규칙을 보여줍니다.

> **Note**
>
> 이 규칙을 사용하여 트래픽을 제한하려면 포트 443에서 TCP 트래픽을 허용하는 다른 규칙을 추가해야 합니다.

유형	프로토콜	포트 범위	소스 유형	소스	
사용자 지정 TCP	TCP	5432	사용자 지정 (Custom)	sg-0909e8e81919/-그룹 my-mwaa-vpc-security	

(선택 사항) 포트 443에 대한 인바운드 액세스를 제한하는 보안 그룹의 예제

다음 예제는 Apache Airflow 웹 서버의 포트 443에서 모든 TCP 트래픽을 허용하는 인바운드 보안 그룹 규칙을 보여줍니다.

유형	프로토콜	포트 범위	소스 유형	소스	
HTTPS	TCP	443	사용자 지정 (Custom)	sg-0909e8e81919/-그룹 my-mwaa-vpc-security	

VPC 엔드포인트 정책(프라이빗 라우팅만 해당)

VPC 엔드포인트 (AWS PrivateLink) 정책은 프라이빗 서브넷의 AWS 서비스 액세스를 제어합니다. VPC 엔드포인트 정책은 VPC 게이트웨이 또는 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스 정책입니다. 이 섹션에서는 각 VPC 엔드포인트의 VPC 엔드포인트 정책에 필요한 권한을 설명합니다.

생성한 각 VPC 엔드포인트에 대해 모든 AWS 서비스에 대한 전체 액세스를 허용하는 VPC 인터페이스 엔드포인트 정책을 사용하고 실행 역할을 권한에만 사용하는 것이 좋습니다. AWS

(권장) 모든 액세스를 허용하는 VPC 엔드포인트 정책 예제

다음 예제는 프라이빗 라우팅을 사용하는 Amazon VPC에 대한 VPC 인터페이스 엔드포인트 정책을 보여줍니다.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

(권장) 버킷 액세스를 허용하는 Amazon S3 게이트웨이 엔드포인트 정책의 예제

다음 예제는 프라이빗 라우팅을 사용하는 Amazon VPC의 Amazon ECR 작업에 필요한 Amazon S3 버킷에 액세스 권한을 부여하는 VPC 게이트웨이 엔드포인트 정책을 보여줍니다. 이는 DAG와 지원 파일이 저장되는 버킷 외에도 Amazon ECR 이미지를 검색하는 데 필요합니다.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
```

```
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

아마존 MWAA의 서비스별 아마존 VPC 엔드포인트에 대한 액세스 관리

VPC 엔드포인트 (AWS PrivateLink) 를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 또는 방화벽 프록시 AWS 없이도 VPC를 호스팅된 서비스에 비공개로 연결할 수 있습니다. 이러한 엔드포인트는 VPC와 서비스의 인스턴스 간 통신을 허용하는 수평적으로 확장 가능하고 가용성이 높은 가상 디바이스입니다. AWS 이 페이지에서는 Amazon MWAA에서 생성한 VPC 엔드포인트와 Amazon Managed Workflows for Apache Airflow에서 프라이빗 네트워크 액세스 모드를 선택한 경우 Apache Airflow 웹 서버의 VPC 엔드포인트에 액세스하는 방법을 설명합니다.

목차

요금

- AWS PrivateLink 요금

VPC 엔드포인트 개요

Amazon MWAA 환경을 생성할 때 Amazon MWAA는 사용자 환경에 대해 1~2개의 VPC 엔드포인트를 생성합니다. 이러한 엔드포인트는 Amazon VPC에 프라이빗 IP가 있는 탄력적 네트워크 인터페이스(ENI)로 표시됩니다. 이러한 엔드포인트가 생성되면 해당 IP로 향하는 모든 트래픽이 사용자 환경에서 사용되는 해당 서비스에 비공개 또는 공개적으로 라우팅됩니다. AWS

퍼블릭 네트워크 액세스 모드

Apache Airflow 웹 서버의 퍼블릭 네트워크 액세스 모드를 선택한 경우, 네트워크 트래픽은 인터넷을 통해 공개적으로 라우팅됩니다.

- Amazon MWAA는 Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 위한 VPC 인터페이스 엔드포인트를 생성합니다. 엔드포인트는 프라이빗 서브넷에 매핑된 가용 영역에 생성되며 다른 계정과는 독립적입니다. AWS
- 그러면 Amazon MWAA에서 프라이빗 서브넷의 IP 주소를 인터페이스 엔드포인트에 바인딩합니다. 이는 Amazon VPC의 각 가용 영역에서 단일 IP를 바인딩하는 모범 사례를 지원하도록 설계되었습니다.

프라이빗 네트워크 액세스 모드

Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택한 경우, 네트워크 트래픽은 Amazon VPC 내에서 프라이빗로 라우팅됩니다.

- Amazon MWAA는 Apache Airflow 웹 서버를 위한 VPC 인터페이스 엔드포인트와, Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 위한 인터페이스 엔드포인트를 생성합니다. 엔드포인트는 프라이빗 서브넷에 매핑된 가용 영역에서 생성되며 다른 계정과는 독립적입니다. AWS
- 그러면 Amazon MWAA에서 프라이빗 서브넷의 IP 주소를 인터페이스 엔드포인트에 바인딩합니다. 이는 Amazon VPC의 각 가용 영역에서 단일 IP를 바인딩하는 모범 사례를 지원하도록 설계되었습니다.

다른 서비스를 사용할 수 있는 권한 AWS

인터페이스 엔드포인트는 IAM AWS Identity and Access Management (환경) 의 실행 역할을 사용하여 사용자 환경에서 사용되는 AWS 리소스에 대한 권한을 관리합니다. 한 환경에서 더 많은 AWS 서비스를 사용할 수 있게 되면 각 서비스에서는 환경의 실행 역할을 사용하여 권한을 구성해야 합니다. 권한을 추가하려면 Amazon MWAA 실행 역할 단원을 참조하십시오.

Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택한 경우 VPC 엔드포인트 정책에서 각 엔드포인트에 대한 권한도 허용해야 합니다. 자세한 내용은 the section called “VPC 엔드포인트 정책(프라이빗 라우팅만 해당)” 섹션을 참조하세요.

VPC 엔드포인트 보기

이 섹션에서는 Amazon MWAA에서 생성한 VPC 엔드포인트를 보는 방법과 Apache Airflow VPC 엔드포인트의 프라이빗 IP 주소를 식별하는 방법을 설명합니다.

Amazon VPC 콘솔에서 VPC 엔드포인트 보기

다음 섹션에서는 Amazon MWAA에서 생성한 VPC 엔드포인트를 보는 단계와 Amazon VPC에 프라이빗 라우팅을 사용하는 경우 생성했을 수 있는 모든 VPC 엔드포인트를 확인하는 단계를 보여줍니다.

VPC 엔드포인트를 보려면

1. Amazon VPC 콘솔에서 엔드포인트 페이지를 엽니다.
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.
3. Amazon MWAA에서 생성한 VPC 인터페이스 엔드포인트와 Amazon VPC에서 프라이빗 라우팅을 사용하는 경우 생성했을 수 있는 모든 VPC 엔드포인트이 표시됩니다.

프라이빗 라우팅을 사용하는 Amazon VPC에 필요한 VPC 서비스 엔드포인트에 대한 자세한 내용은 프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 서비스 엔드포인트 생성 단원을 참조하십시오.

Apache Airflow 웹 서버 및 VPC 엔드포인트의 프라이빗 IP 주소 식별

다음 단계는 Apache Airflow 웹 서버의 호스트 이름과 VPC 인터페이스 엔드포인트, 프라이빗 IP 주소를 검색하는 방법을 설명합니다.

1. 다음 AWS Command Line Interface (AWS CLI) 명령을 사용하여 Apache Airflow 웹 서버의 호스트 이름을 검색합니다.

   ```
   aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query
    'Environment.WebserverUrl'
   ```

   다음 응답과 유사한 결과가 출력되어야 합니다.

   ```
   "99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
   ```

2. 이전 명령의 응답에서 반환된 호스트 이름에 대해 dig 명령을 실행합니다. 예:

```
dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-
west-2.airflow.amazonaws.com
```

다음 응답과 유사한 결과가 출력되어야 합니다.

```
vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com.
```

3. 다음 AWS Command Line Interface (AWS CLI) 명령을 사용하여 이전 명령의 응답에서 반환된 VPC 엔드포인트 DNS 이름을 검색합니다. 예:

```
aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-
svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

다음 응답과 유사한 결과가 출력되어야 합니다.

```
"DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com",
```

4. Apache Airflow 호스트 이름과 VPC 엔드포인트 DNS 이름에서 nslookup 또는 dig 명령을 실행하여 IP 주소를 검색합니다. 예:

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

다음 응답과 유사한 결과가 출력되어야 합니다.

```
192.0.5.1
  192.0.6.1
```

Apache Airflow 웹 서버의 VPC 엔드포인트 액세스(프라이빗 네트워크 액세스)

Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택한 경우 Apache Airflow 웹 서버의 VPC 인터페이스 엔드포인트에 액세스하는 메커니즘을 만들어야 합니다. 이러한 리소스에 대해 Amazon MWAA 환경과 동일한 Amazon VPC, VPC 보안 그룹 및 프라이빗 서브넷을 사용해야 합니다.

사용: AWS Client VPN

AWS Client VPN 온프레미스 네트워크의 AWS 리소스 및 리소스에 안전하게 액세스할 수 있게 해주는 관리형 클라이언트 기반 VPN 서비스입니다. OpenVPN 클라이언트를 사용하여 어떤 위치에서든 안전한 TLS 연결을 제공합니다.

Amazon MWAA 튜토리얼에 따라 Client VPN을 구성하는 것이 좋습니다: 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성.

Linux Bastion Host 사용

Bastion Host는 외부 네트워크를 통해(예: 컴퓨터에서 인터넷을 통해) 프라이빗 네트워크에 액세스할 수 있도록 하는 것을 목적으로 하는 서버입니다. Linux 인스턴스는 퍼블릭 서브넷에 있으며 Bastion Host를 실행하는 기본 Amazon EC2 인스턴스에 연결된 보안 그룹으로부터 SSH 액세스를 허용하는 보안 그룹으로 설정됩니다.

Amazon MWAA 튜토리얼에 따라 Linux Bastion Host를 구성하는 것이 좋습니다: 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성.

로드 밸런서 사용(고급)

다음 섹션에서는 Application Load Balancer에 적용해야 하는 구성을 보여줍니다.

1. 대상 그룹. Apache Airflow 웹 서버의 프라이빗 IP 주소와 VPC 인터페이스 엔드포인트를 가리키는 대상 그룹을 사용해야 합니다. 프라이빗 IP 주소를 하나만 사용하면 가용성이 떨어질 수 있으므로 두 프라이빗 IP 주소를 모두 등록된 대상으로 지정하는 것이 좋습니다. 프라이빗 IP 주소를 식별하는 방법에 대한 자세한 내용은 the section called “Apache Airflow 웹 서버 및 VPC 엔드포인트의 프라이빗 IP 주소 식별” 단원을 참조하십시오.
2. 상태 코드. 대상 그룹 설정에서 `200` 및 `302` 상태 코드를 사용하는 것이 좋습니다. 그렇지 않으면 Apache Airflow 웹 서버의 VPC 엔드포인트가 `302 Redirect` 오류로 응답하는 경우 대상이 비정상으로 플래그될 수 있습니다.
3. HTTPS 리스너. Apache Airflow 웹 서버의 대상 포트를 지정해야 합니다. 예:

프로토콜	Port
HTTPS	443

4. ACM 새 도메인. SSL/TLS 인증서를 연결하려면 로드 밸런서의 AWS Certificate Manager HTTPS 수신기용 새 도메인을 만들어야 합니다.

5. ACM 인증서 리전. 에 SSL/TLS 인증서를 연결하려면 환경과 동일한 지역에 AWS Certificate Manager업로드해야 합니다. AWS 예:

   - Example 인증서를 업로드할 리전

```
aws acm import-certificate --certificate fileb://Certificate.pem --certificate-
chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --
region us-west-2
```

프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 서비스 엔드포인트 생성

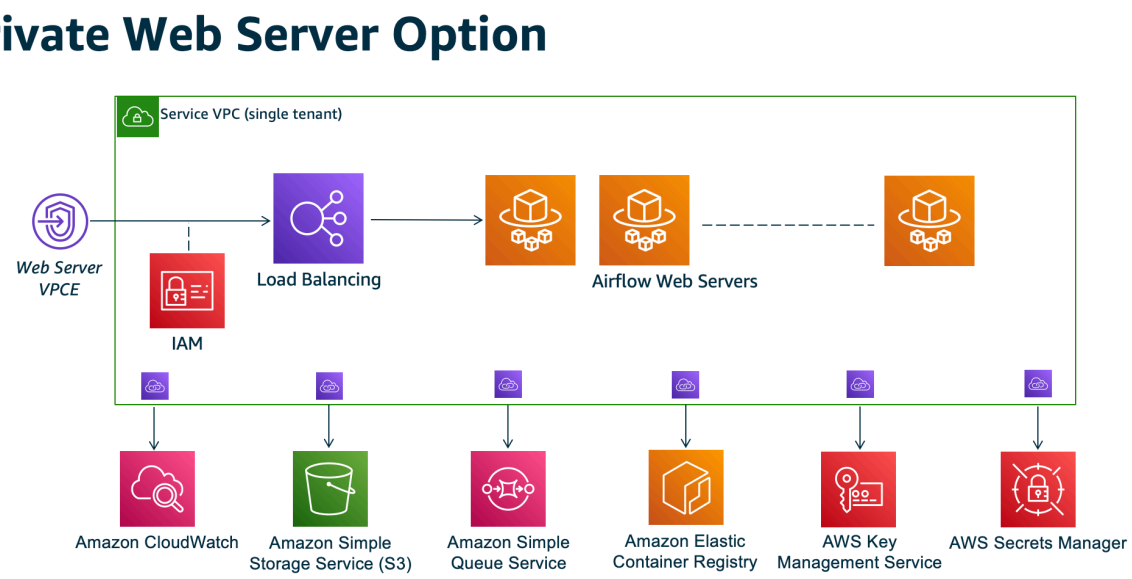
인터넷에 액세스할 수 없는 기존 Amazon VPC 네트워크에서는 Amazon Managed Workflows for Apache Airflow에서 Apache Airflow를 사용하려면 추가 VPC 서비스 엔드포인트(AWS PrivateLink)가 필요합니다. 이 페이지에서는 Amazon MWAA에서 사용하는 AWS 서비스에 필요한 VPC 엔드포인트, Apache Airflow에 필요한 VPC 엔드포인트, VPC 엔드포인트를 생성하여 프라이빗 라우팅을 사용하여 기존 Amazon VPC에 연결하는 방법을 설명합니다.

목차

요금

- AWS PrivateLink 요금

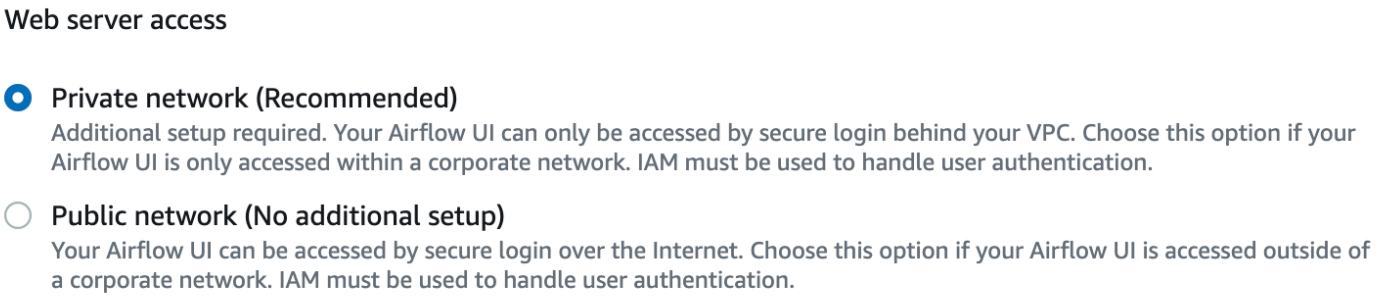
프라이빗 네트워크 및 프라이빗 라우팅



프라이빗 네트워크 액세스 모드를 사용하면 Amazon VPC 내에서 사용자 환경의 IAM 정책에 대한 액세스 권한을 부여 받은 사용자만 Apache Airflow UI에 액세스할 수 있도록 제한됩니다.

프라이빗 웹 서버 액세스가 가능한 환경을 만들 때는 모든 종속성을 Python 휠 아카이브(`.whl`)에서 패키징한 다음 `requirements.txt`에서 `.whl`을 참조해야 합니다. 휠을 사용하여 종속성을 패키징하고 설치하는 방법에 대한 지침은 Python 휠을 이용한 종속성 관리를 참조하십시오.

다음 이미지는 Amazon MWAA 콘솔에서 프라이빗 네트워크 옵션을 찾을 수 있는 위치를 보여줍니다.



- 프라이빗 라우팅. 인터넷에 액세스할 수 없는 Amazon VPC는 VPC 내의 네트워크 트래픽을 제한합니다. 이 페이지에서는 Amazon VPC가 인터넷에 액세스할 수 없고 사용자 환경에서 사용하는 AWS 각 서비스에 대해 VPC 엔드포인트가 필요하고 Amazon MWAA 환경과 동일한 지역 및 AWS Amazon VPC의 Apache Airflow용 VPC 엔드포인트가 필요하다고 가정합니다.

(필수) VPC 엔드포인트

다음 섹션에서는 인터넷에 액세스할 수 없는 Amazon VPC에 필요한 필수 VPC 엔드포인트를 보여줍니다. 여기에는 Apache Airflow에 필요한 VPC 엔드포인트를 포함하여 Amazon MWAA에서 사용하는 각 AWS 서비스의 VPC 엔드포인트가 나열되어 있습니다.

```
com.amazonaws.YOUR_REGION.s3
com.amazonaws.YOUR_REGION.monitoring
com.amazonaws.YOUR_REGION.ecr.dkr
com.amazonaws.YOUR_REGION.ecr.api
com.amazonaws.YOUR_REGION.logs
com.amazonaws.YOUR_REGION.sqs
com.amazonaws.YOUR_REGION.kms
com.amazonaws.YOUR_REGION.airflow.api
com.amazonaws.YOUR_REGION.airflow.env
com.amazonaws.YOUR_REGION.airflow.ops
```

필수 VPC 엔드포인트 연결

이 섹션에서는 프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 엔드포인트를 연결하는 단계를 설명합니다.

서비스에 필요한 VPC 엔드포인트 AWS

다음 섹션은 환경에서 사용하는 AWS 서비스의 VPC 엔드포인트를 기존 Amazon VPC에 연결하는 단계를 보여줍니다.

VPC 엔드포인트를 프라이빗 서브넷에 연결하려면

1. Amazon VPC 콘솔에서 엔드포인트 페이지를 엽니다.
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.
3. Amazon S3에 대한 엔드포인트 생성
    a. 엔드포인트 생성을 선택합니다.
    b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.s3**를 입력한 다음 키보드에서 Enter를 누릅니다.
    c. 게이트웨이 유형에 나열된 서비스 엔드포인트를 선택하는 것이 좋습니다.

        예제: **com.amazonaws.us-west-2.s3 amazon Gateway**

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택하고, DNS 이름 활성화를 선택하여 프라이빗 DNS가 활성화되었는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

4. Amazon ECR에 대한 첫 번째 엔드포인트 생성

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.ecr.dkr**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

5. Amazon ECR에 대한 두 번째 엔드포인트 생성

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.ecr.api**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

6. CloudWatch 로그용 엔드포인트 생성:

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.logs**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

7. CloudWatch 모니터링을 위한 엔드포인트 생성:

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.monitoring**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

8. Amazon SQS에 대한 VPC 엔드포인트 생성:

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.sqs**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

9. AWS KMS다음을 위한 엔드포인트 생성:

    a. 엔드포인트 생성을 선택합니다.

    b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.kms**를 입력한 다음 키보드에서 Enter를 누릅니다.

    c. 서비스 엔드포인트를 선택합니다.

    d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

    e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

    f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

    g. 정책에서 전체 액세스를 선택합니다.

    h. Create endpoint(엔드포인트 생성)을 선택합니다.

Apache Airflow에 필요한 VPC 엔드포인트

다음 섹션은 Apache Airflow의 VPC 엔드포인트를 기존 Amazon VPC에 연결하는 단계를 보여줍니다.

VPC 엔드포인트를 프라이빗 서브넷에 연결하려면

1. Amazon VPC 콘솔에서 엔드포인트 페이지를 엽니다.
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.
3. Apache Airflow API를 위한 엔드포인트 생성:

    a. 엔드포인트 생성을 선택합니다.

    b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.airflow.api**를 입력한 다음 키보드에서 Enter를 누릅니다.

    c. 서비스 엔드포인트를 선택합니다.

    d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

    e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

    f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

    g. 정책에서 전체 액세스를 선택합니다.

    h. Create endpoint(엔드포인트 생성)을 선택합니다.

4. Apache Airflow 환경을 위한 첫 번째 엔드포인트 생성:

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.airflow.env**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

5. Apache Airflow 작업을 위한 두 번째 엔드포인트 생성:

a. 엔드포인트 생성을 선택합니다.

b. 속성별 필터링 또는 키워드별 검색 텍스트 필드에 **.airflow.ops**를 입력한 다음 키보드에서 Enter를 누릅니다.

c. 서비스 엔드포인트를 선택합니다.

d. VPC에서 사용자 환경의 Amazon VPC를 선택합니다.

e. 서로 다른 가용 영역에 있는 두 개의 프라이빗 서브넷을 선택되어 있고 DNS 이름 활성화가 활성화되어 있는지 확인합니다.

f. 사용자 환경의 Amazon VPC 보안 그룹을 선택합니다.

g. 정책에서 전체 액세스를 선택합니다.

h. Create endpoint(엔드포인트 생성)을 선택합니다.

(선택 사항) Amazon S3 VPC 인터페이스 엔드포인트의 프라이빗 IP 주소를 활성화합니다.

Amazon S3 인터페이스 엔드포인트는 프라이빗 DNS를 지원하지 않습니다. S3 엔드포인트 요청이 여전히 퍼블릭 IP 주소로 확인됩니다. S3 주소를 프라이빗 IP 주소로 확인하려면 S3 리전 엔드포인트에 대해 Route 53에서 프라이빗 호스팅 영역을 추가해야 합니다.

Route 53 사용

이 섹션에서는 Route 53을 사용하여 S3 인터페이스 엔드포인트의 프라이빗 IP 주소를 활성화하는 단계를 설명합니다.

1. Amazon S3 VPC 인터페이스 엔드포인트(예: s3.eu-west-1.amazonaws.com)를 위한 프라이빗 호스팅 영역을 생성하고 이를 Amazon VPC와 연결합니다.
2. VPC 인터페이스 엔드포인트 DNS 이름으로 확인되는 Amazon S3 VPC 인터페이스 엔드포인트(예: s3.eu-west-1.amazonaws.com)에 대한 ALIAS A 레코드를 생성합니다.
3. VPC 인터페이스 엔드포인트 DNS 이름으로 확인되는 Amazon S3 인터페이스 엔드포인트(예: *.s3.eu-west-1.amazonaws.com)에 대한 ALIAS A 와일드카드 레코드를 생성합니다.

사용자 지정 DNS를 사용하는 VPC

Amazon VPC가 사용자 지정 DNS 라우팅을 사용하는 경우 CNAME 레코드를 생성하여 DNS 확인자(Route 53 아님, 일반적으로 DNS 서버를 실행하는 EC2 인스턴스)를 변경해야 합니다. 예:

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value:  *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

아마존 MWAA에서 자체 아마존 VPC 엔드포인트 관리

Amazon MWAA는 Amazon VPC 엔드포인트를 사용하여 아파치 에어플로우 환경을 설정하는 데 필요한 다양한 AWS 서비스와 통합합니다. 엔드포인트를 관리하는 주요 사용 사례는 두 가지입니다.

1. 즉, 를 사용하여 여러 AWS 계정을 관리하고 리소스를 공유하면 공유 Amazon VPC에서 Apache Airflow 환경을 만들 수 있습니다. AWS Organizations
2. 엔드포인트를 사용하는 특정 리소스로 권한 범위를 좁혀 보다 제한적인 액세스 정책을 사용할 수 있습니다.

VPC 엔드포인트를 직접 관리하기로 선택한 경우 PostgreSQL용 환경 RDS 데이터베이스와 환경 웹 서버를 위한 자체 엔드포인트를 만들어야 합니다.

Amazon MWAA가 클라우드에 Apache Airflow를 배포하는 방법에 대한 자세한 내용은 Amazon MWAA 아키텍처 다이어그램을 참조하십시오.

공유 Amazon VPC에서 환경 만들기

를 AWS Organizations사용하여 리소스를 공유하는 여러 AWS 계정을 관리하는 경우 Amazon MWAA의 고객 관리형 VPC 엔드포인트를 사용하여 조직의 다른 계정과 환경 리소스를 공유할 수 있습니다.

공유 VPC 액세스를 구성하면 기본 Amazon VPC를 소유한 계정 (소유자) 이 Amazon MWAA에 필요한 두 개의 프라이빗 서브넷을 동일한 조직에 속한 다른 계정 (참가자) 과 공유합니다. 해당 서브넷을 공유하는 참가자 계정은 공유된 Amazon VPC에서 환경을 보고, 만들고, 수정하고, 삭제할 수 있습니다.

조직에서 계정 역할을 하고 Amazon VPC 리소스를 소유하는 `Root` 계정과 동일한 조직의 구성원인 참가자 계정이 `Participant` 있다고 가정해 보겠습니다. `Owner` 공유하는 Amazon VPC에서 새 Amazon MWAA를 `Participant` 생성하면 `Owner` Amazon MWAA가 먼저 서비스 VPC 리소스를 생성한 다음 최대 72시간 동안 상태가 됩니다. `PENDING`

환경 상태가 에서 `CREATING` 로 변경되면`PENDING`, 이를 대신하는 보안 주체가 필요한 엔드포인트를 생성합니다. `Owner` 이를 위해 Amazon MWAA는 Amazon MWAA 콘솔에 데이터베이스 및 웹 서버 엔드포인트를 나열합니다. `GetEnvironment`API 작업을 호출하여 서비스 엔드포인트를 가져올 수도 있습니다.

> **Note**
>
> 리소스를 공유하는 데 사용하는 Amazon VPC가 프라이빗 Amazon VPC인 경우에도 에 설명된 단계를 완료해야 합니다. the section called “VPC 엔드포인트에 대한 액세스 관리” 이 주제에서는 Amazon ECR, Amazon ECS, Amazon SQS와 같이 AWS 통합되는 다른 AWS 서비스와 관련된 다양한 Amazon VPC 엔드포인트 세트를 설정하는 방법을 다룹니다. 이러한 서비스는 클라우드에서 Apache Airflow 환경을 운영 및 관리하는 데 필수적입니다.

사전 조건

공유 VPC에서 Amazon MWAA 환경을 생성하기 전에 다음 리소스가 필요합니다.

- Amazon `Owner` VPC를 소유하는 계정으로 사용할 계정입니다. AWS
- 루트로 *`MyOrganization`* 생성된 AWS Organizations조직 단위.
- 새 환경을 만드는 참가자 계정을 `MyOrganization` 제공하기 위한 두 번째 AWS 계정입니다. `Participant`

또한 Amazon VPC에서 리소스를 공유할 때는 소유자와 참여자의 책임과 권한을 숙지하는 것이 좋습니다.

아마존 VPC 생성

먼저 소유자와 참여자 계정이 공유할 새 Amazon VPC를 생성합니다.

1. 를 사용하여 Owner 콘솔에 로그인한 다음 AWS CloudFormation 콘솔을 엽니다. 다음 템플릿을 사용하여 스택을 생성합니다. 이 스택은 Amazon VPC와 이 시나리오에서 두 계정이 공유할 서브넷을 비롯한 여러 네트워킹 리소스를 제공합니다.

```
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. 새 Amazon VPC 리소스가 프로비저닝된 후 AWS Resource Access Manager 콘솔로 이동한 다음 [리소스 공유 생성] 을 선택합니다.
3. 공유할 수 있는 사용 가능한 서브넷 목록에서 첫 번째 단계에서 만든 서브넷을 선택합니다. `Participant`

환경 생성

다음 단계를 완료하여 고객 관리형 Amazon VPC 엔드포인트가 있는 Amazon MWAA 환경을 생성하십시오.

1. 를 사용하여 `Participant` 로그인하고 Amazon MWAA 콘솔을 엽니다. 1단계 완료: 세부 정보를 지정하여 Amazon S3 버킷, DAG 폴더 및 새 환경에 대한 종속성을 지정합니다. 자세한 내용은 시작하기를 참조하십시오.
2. 고급 설정 구성 페이지의 네트워킹에서 공유 Amazon VPC의 서브넷을 선택합니다.
3. 엔드포인트 관리에서 드롭다운 목록에서 CUSTOMER를 선택합니다.

4. 페이지의 나머지 옵션에 대한 기본값을 유지한 다음 검토 및 생성 페이지에서 환경 생성을 선택합니다.

환경은 한 CREATING 상태에서 시작하여 로 바뀝니다PENDING. 환경이 설정되면 콘솔을 사용하여 데이터베이스 엔드포인트 서비스 이름과 웹 서버 엔드포인트 서비스 이름 (사설 웹 서버를 설정한 경우)을 기록해 둡니다. PENDING

Amazon MWAA 콘솔을 사용하여 새 환경을 만들 때. Amazon MWAA는 필수 인바운드 및 아웃바운드 규칙을 사용하여 새 보안 그룹을 생성합니다. 보안 그룹 ID를 기록합니다.

다음 섹션에서는 서비스 엔드포인트와 보안 그룹 ID를 사용하여 공유 Amazon VPC에 새 Amazon VPC 엔드포인트를 생성합니다. 0wner

Amazon VPC 엔드포인트 생성

다음 단계를 완료하여 환경에 필요한 Amazon VPC 엔드포인트를 생성하십시오.

1. 열려 있는 https://console.aws.amazon.com/vpc/ 0wner 를 AWS Management Console 사용하여 로그인하십시오.
2. 왼쪽 탐색 패널에서 보안 그룹을 선택한 다음 다음 인바운드 및 아웃바운드 규칙을 사용하여 공유 Amazon VPC에서 새 보안 그룹을 생성합니다.

	유형	프로토콜	소스 유형	소스(Source)
인바운드	모든 트래픽	모두	모두	환경 보안 그룹
아웃바운드	모든 트래픽	모두	모두	0.0.0.0/0

> **Warning**
>
> 새 환경에서 공유 Amazon VPC로의 트래픽을 허용하려면 0wner 계정에 보안 그룹을 설정해야 합니다. 0wner 에서 0wner 새 보안 그룹을 만들거나 기존 보안 그룹을 편집하여 이 작업을 수행할 수 있습니다.

3. 엔드포인트를 선택한 다음 이전 단계의 엔드포인트 서비스 이름을 사용하여 환경 데이터베이스 및 웹 서버 (프라이빗 모드인 경우) 에 대한 새 엔드포인트를 생성합니다. 공유 Amazon VPC, 환경에 사용한 서브넷, 환경의 보안 그룹을 선택합니다.

성공하면 환경이 처음부터 PENDING 다시CREATING, 마지막으로 다시 바뀔 것입니다. AVAILABLE 그러면 Apache Airflow 콘솔에 로그인할 수 있습니다. AVAILABLE

공유 Amazon VPC 문제 해결

다음 참조를 사용하여 공유 Amazon VPC에서 환경을 생성할 때 발생하는 문제를 해결하십시오.

CREATE_FAILED사후 PENDING 상태의 환경

- 를 Participant 사용하여 AWS Resource Access Manager서브넷을 공유하고 Owner 있는지 확인하십시오.
- 데이터베이스 및 웹 서버의 Amazon VPC 엔드포인트가 환경과 연결된 동일한 서브넷에 생성되었는지 확인합니다.
- 엔드포인트와 함께 사용되는 보안 그룹이 환경에 사용되는 보안 그룹으로부터의 트래픽을 허용하는지 확인하십시오. Owner계정은 *account-number*/*security-group-id* 다음과 Participant 같이 보안 그룹을 참조하는 규칙을 생성합니다.

유형	프로토콜	소스 유형	소스(Source)
모든 트래픽	모두	모두	*123456789012/ sg-0909e8e81919*

자세한 내용은 소유자 및 참여자의 책임 및 권한을 참조하십시오.

환경 **PENDING** 상태가 멈췄습니다.

각 VPC 엔드포인트 상태를 확인하여 상태가 맞는지 확인합니다. Available 사설 웹 서버로 환경을 구성하는 경우 웹 서버용 엔드포인트도 만들어야 합니다. 환경이 중단되면 사설 웹 서버 엔드포인트가 누락되었음을 의미할 수 있습니다. PENDING

수신된 **The Vpc Endpoint Service '*vpce-service-name*' does not exist** 오류

다음 오류가 표시되면 공유 VPC를 소유한 계정에서 엔드포인트를 생성하는 Owner 계정을 확인하십시오.

```
ClientError: An error occurred (InvalidServiceName) when calling the
 CreateVpcEndpoint operation:

The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Amazon Managed Workflows for Apache Airflow 튜토리얼

이 안내서에는 Apache Airflow용 Amazon 관리형 워크플로 환경을 사용하고 구성하는 방법에 대한 step-by-step 자습서가 포함되어 있습니다.

주제

- 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성
- 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성
- 튜토리얼: DAG의 하위 집합에 대한 Amazon MWAA 사용자 액세스 제한
- 자습서: Amazon MWAA에서 자체 환경 엔드포인트를 자동으로 관리합니다.

튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성

이 튜토리얼은 Amazon Managed Workflows for Apache Airflow 환경을 위해 컴퓨터에서 Apache Airflow 웹 서버로 VPN 터널을 생성하는 단계를 안내합니다. VPN 터널을 통해 인터넷에 연결하려면 먼저 AWS Client VPN 엔드포인트를 만들어야 합니다. 일단 설정이 완료되면 Client VPN 엔드포인트는 VPN 서버 역할을 하여 컴퓨터와 VPC의 리소스에 안전하게 연결할 수 있습니다. 그런 다음 AWS Client VPN 데스크톱용을 사용하여 컴퓨터에서 Client VPN에 연결합니다.

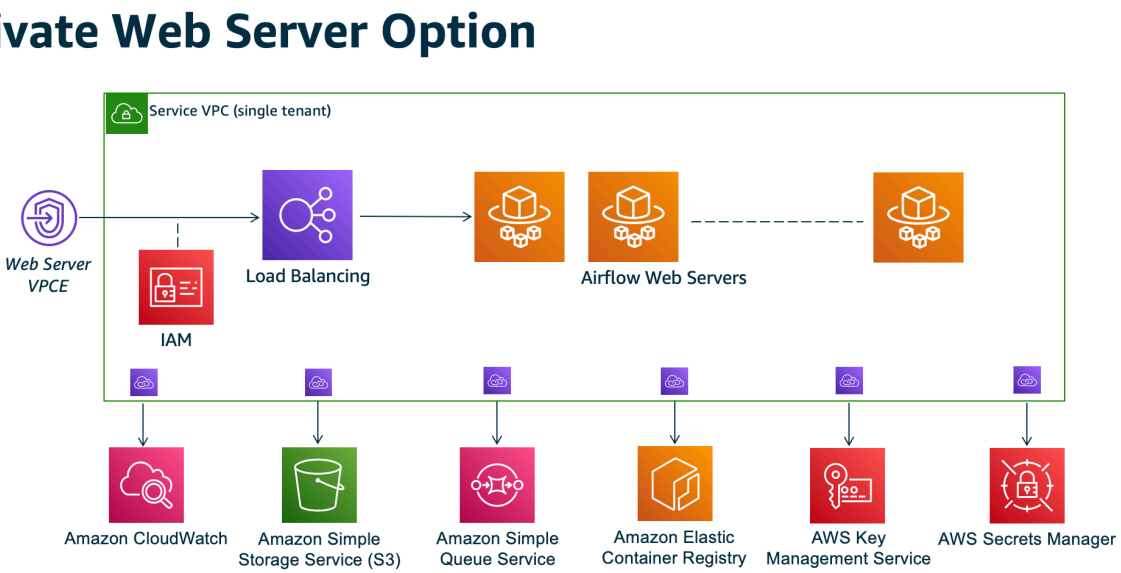
섹션

- 프라이빗 네트워크
- 사용 사례
- 시작하기 전에
- 목표
- (선택 사항) 1단계: VPC, CIDR 규칙 및 VPC 보안을 식별합니다.
- 2단계: 서버 및 클라이언트 인증서 생성
- 3단계: AWS CloudFormation 템플릿을 로컬에 저장
- 4단계: Client VPN AWS CloudFormation 스택 생성
- 5단계: Client VPN에 서브넷을 연결
- 6단계: Client VPN에 인증 수신 규칙 추가
- 7단계: Client VPN 엔드포인트 구성 파일 다운로드

- 8단계: AWS Client VPN에 연결
- 다음 단계

프라이빗 네트워크

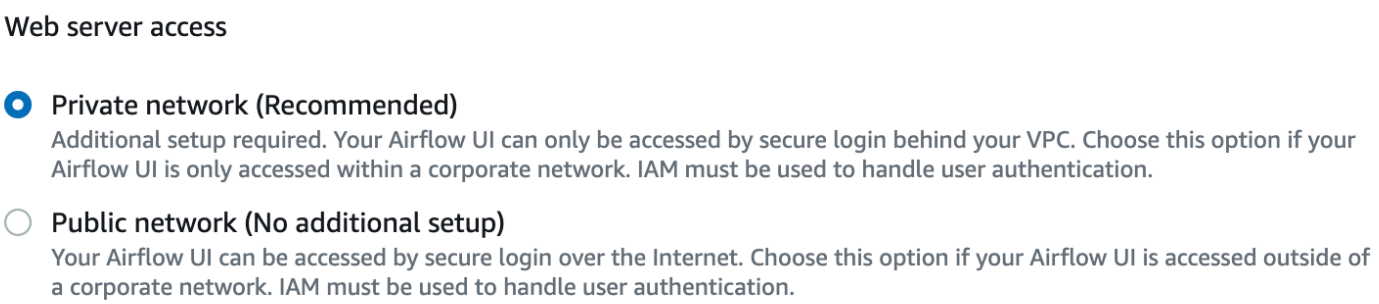
이 튜토리얼에서는 Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택했다고 가정합니다.



프라이빗 네트워크 액세스 모드를 사용하면 Amazon VPC 내에서 사용자 환경의 IAM 정책에 대한 액세스 권한을 부여 받은 사용자만 Apache Airflow UI에 액세스할 수 있도록 제한됩니다.

프라이빗 웹 서버 액세스가 가능한 환경을 만들 때는 모든 종속성을 Python 휠 아카이브(`.whl`)에서 패키징한 다음 `requirements.txt`에서 `.whl`을 참조해야 합니다. 휠을 사용하여 종속성을 패키징하고 설치하는 방법에 대한 지침은 Python 휠을 이용한 종속성 관리를 참조하십시오.

다음 이미지는 Amazon MWAA 콘솔에서 프라이빗 네트워크 옵션을 찾을 수 있는 위치를 보여줍니다.



사용 사례

Amazon MWAA 환경을 생성하기 전 또는 생성한 후에 이 튜토리얼을 활용할 수 있습니다. 사용자 환경과 동일한 Amazon VPC, VPC 보안 그룹 및 프라이빗 서브넷을 사용해야 합니다. Amazon MWAA 환경을 만든 후 이 튜토리얼을 사용하는 경우 단계를 완료한 후 Amazon MWAA 콘솔로 돌아가서 Apache Airflow 웹 서버 액세스 모드를 프라이빗 네트워크로 변경할 수 있습니다.

시작하기 전에

1. 사용자 권한을 확인합니다. AWS Identity and Access Management(IAM)의 사용자 계정에 VPC 리소스를 생성하고 관리할 수 있는 충분한 권한이 있는지 확인합니다.
2. Amazon MWAA VPC를 사용합니다. 이 튜토리얼에서는 Client VPN을 기존 VPC에 연결한다고 가정합니다. Amazon VPC는 Amazon MWAA 환경과 동일한 AWS 리전에 있어야 하며 두 개의 프라이빗 서브넷이 있어야 합니다. Amazon VPC를 생성하지 않은 경우 옵션 3: 인터넷에 액세스할 수 없는 Amazon VPC 네트워크 생성에서 AWS CloudFormation 템플릿을 사용합니다.

목표

이 튜토리얼에서는 다음을 수행합니다.

1. 기존 Amazon VPC용 AWS CloudFormation 템플릿을 사용하여 AWS Client VPN 엔드포인트를 생성합니다.
2. 서버와 클라이언트 인증서 및 키를 생성한 다음, 서버 인증서와 키를 Amazon MWAA 환경과 동일한 AWS 리전의 AWS Certificate Manager에 업로드합니다.
3. Client VPN용 Client VPN 엔드포인트 구성 파일을 다운로드 및 수정하고, 이 파일을 사용하여 데스크톱용 Client VPN을 이용해 연결할 VPN 프로필을 생성합니다.

(선택 사항) 1단계: VPC, CIDR 규칙 및 VPC 보안을 식별합니다.

다음 섹션에서는 Amazon VPC, VPC 보안 그룹의 ID를 찾는 방법과 이후 단계에서 Client VPN을 생성하는 데 필요한 CIDR 규칙을 식별하는 방법을 설명합니다.

CIDR 규칙 식별

다음 섹션에서는 Client VPN을 만드는 데 필요한 CIDR 규칙을 식별하는 방법을 보여줍니다.

Client VPN의 CIDR을 식별하려면

1. Amazon VPC 콘솔에서 내 Amazon VPC 페이지를 엽니다.
2. 내비게이션 바의 리전 선택기를 사용하여 Amazon MWAA 환경과 동일한 AWS 리전을 선택합니다.
3. Amazon VPC를 선택합니다.
4. 프라이빗 서브넷의 CIDR이 다음과 같다고 가정합니다.

   - 프라이빗 서브넷 1: 10.192.10.0/24
   - 프라이빗 서브넷 2: 10.192.11.0/24

   Amazon VPC의 CIDR이 10.192.0.0/16인 경우, Client VPN에 지정한 Client IPv4 CIDR은 10.192.0.0/22이(가) 됩니다.
5. 다음 단계를 위해 이 CIDR 값과 VPC ID의 값을 저장합니다.

VPC 및 보안 그룹 식별

다음 섹션에서는 Client VPN을 생성하는 데 필요한 Amazon VPC 및 보안 그룹의 ID를 찾는 방법을 보여줍니다.

> **Note**
>
> 보안 그룹을 두 개 이상 사용하고 있을 수 있습니다. 이후 단계에서 사용자 VPC의 보안 그룹을 모두 지정해야 합니다.

보안 그룹을 식별하려면

1. Amazon VPC 콘솔에서 보안 그룹 페이지를 엽니다.
2. 내비게이션 바에서 리전 선택기를 사용하여 AWS 리전을 선택합니다.
3. VPC ID에서 Amazon VPC를 찾아 VPC와 관련된 보안 그룹을 식별합니다.
4. 후속 단계를 위해 보안 그룹 및 VPC의 ID를 저장합니다.

2단계: 서버 및 클라이언트 인증서 생성

Client VPN 엔드포인트는 1024비트 및 2048비트 RSA 키 크기만 지원합니다. 다음 섹션에서는 OpenVPN easy-rsa를 사용하여 서버 및 클라이언트 인증서와 키를 생성한 다음 AWS Command Line Interface(AWS CLI)을(를) 사용하여 인증서를 ACM에 업로드하는 방법을 보여줍니다.

클라이언트 인증서를 생성하려면

1. 다음의 빠른 단계에 따라 클라이언트 인증 및 권한 부여: 상호 인증의 AWS CLI을(를) 통해 인증서를 생성하고 ACM에 업로드합니다.
2. 이 단계에서는 서버 및 클라이언트 인증서를 업로드할 때 AWS CLI 명령에 Amazon MWAA 환경과 동일한 AWS 리전을 반드시 지정해야 합니다. 다음은 이러한 명령에서 리전을 지정하는 방법을 몇 가지 예시로 보여줍니다.

   a. Example 서버 인증서 리전

   ```
   aws acm import-certificate --certificate fileb://server.crt --private-key
    fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
   ```

   b. Example 클라이언트 인증서 리전

   ```
   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. 이 단계를 거친 후 서버 인증서 및 클라이언트 인증서 ARN에 대한 AWS CLI 응답에 반환된 값을 저장해야 합니다. AWS CloudFormation 템플릿에서 이러한 ARN을 지정하여 Client VPN을 생성합니다.
3. 이 단계에서는 클라이언트 인증서와 프라이빗 키가 컴퓨터에 저장됩니다. 다음은 이러한 보안 인증을 찾을 수 있는 방법을 보여주는 예시입니다.

   a. Example macOS에서

   macOS에서는 콘텐츠가 `/Users/youruser/custom_folder`에 저장됩니다. 이 디렉터리의 모든 (`ls -a`) 콘텐츠를 나열하면 다음과 비슷한 내용이 표시됩니다.

   ```
   .
   ..
   ca.crt
   client1.domain.tld.crt
   ```

```
client1.domain.tld.key
server.crt
server.key
```

b. 이 단계를 수행한 후 내용을 저장하거나 `client1.domain.tld.crt`의 클라이언트 인증서와 `client1.domain.tld.key`의 프라이빗 키 위치를 메모해 둡니다. 이 값을 Client VPN의 구성 파일에 추가할 것입니다.

3단계: AWS CloudFormation 템플릿을 로컬에 저장

다음 섹션에는 Client VPN을 생성하기 위한 AWS CloudFormation 템플릿이 포함되어 있습니다. Amazon MWAA 환경과 동일한 Amazon VPC, VPC 보안 그룹 및 프라이빗 서브넷을 지정해야 합니다.

- 다음 템플릿의 내용을 복사하고 로컬에 `mwaa_vpn_client.yaml`로 저장합니다. 템플릿을 다운로드할 수도 있습니다.

  다음 값을 대체합니다.

  - **`YOUR_CLIENT_ROOT_CERTIFICATE_ARN`** – `ClientRootCertificateChainArn`의 client1.domain.tld 인증서에 대한 ARN.
  - **`YOUR_SERVER_CERTIFICATE_ARN`** – `ServerCertificateArn`의 server 인증서에 대한 ARN.
  - `ClientCidrBlock`의 Client IPv4 CIDR 규칙. `10.192.0.0/22`의 CIDR 규칙이 제공되었습니다.
  - `VpcId`의 사용자 Amazon VPC ID. `vpc-010101010101`의 VPC가 제공되었습니다.
  - `SecurityGroupIds`의 사용자 VPC 보안 그룹 ID. `sg-0101010101`의 보안 그룹이 제공되었습니다.

```
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

환경에 두 가지 이상의 보안 그룹을 사용하는 경우 다음 형식으로 여러 보안 그룹을 지정할 수 있습니다.

```
SecurityGroupIds:
      - sg-0112233445566778b
      - sg-0223344556677889f
```

4단계: Client VPN AWS CloudFormation 스택 생성

AWS Client VPN를 생성하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. 템플릿이 준비됨을 선택하고 템플릿 파일 업로드를 선택합니다.
3. 파일 선택을 선택하고 `mwaa_vpn_client.yaml` 파일을 선택합니다.
4.

5. 다음, 다음을 선택합니다.
6. 승인을 선택한 다음 스택 생성을 선택합니다.

5단계: Client VPN에 서브넷을 연결

프라이빗 서브넷을 AWS Client VPN에 연결하려면

1. Amazon VPC 콘솔을 엽니다.
2. Client VPN 엔드포인트 페이지를 선택합니다.
3. Client VPN을 선택한 다음 연결 탭에서 연결을 선택합니다.
4. 드롭다운 목록에서 다음을 선택합니다.
   - VPC의 사용자 Amazon VPC.
   - 연결할 서브넷 선택에 있는 프라이빗 서브넷 중 하나.
5. 연결을 선택합니다.

> **Note**
>
> VPC와 서브넷을 Client VPN에 연결하는 데 몇 분 정도 걸립니다.

6단계: Client VPN에 인증 수신 규칙 추가

VPC의 CIDR 규칙을 사용하는 인증 수신 규칙을 Client VPN에 추가해야 합니다. Active Directory 그룹 또는 SAML 기반 ID 제공업체(idP)의 특정 사용자 또는 그룹에 권한을 부여하려면 Client VPN 가이드의 권한 부여 규칙을 참조하십시오.

AWS Client VPN에 CIDR을 추가하려면

1. Amazon VPC 콘솔을 엽니다.
2. Client VPN 엔드포인트 페이지를 선택합니다.
3. Client VPN을 선택한 다음 권한 부여 탭의 수신 승인을 선택합니다.
4. 다음을 지정합니다.
   - 대상 네트워크 활성화의 사용자 Amazon VPC CIDR 규칙. 예:

```
10.192.0.0/16
```

- 액세스 권한 부여에서 모든 사용자에게 액세스 허용을 선택합니다.
- 설명에 설명 이름을 입력합니다.

5. 권한 부여 규칙 추가를 선택합니다.

> **Note**
>
> Amazon VPC의 네트워킹 구성 요소에 따라 네트워크 액세스 제어 목록(NACL)에 이 인증 수신 규칙이 필요할 수도 있습니다.

7단계: Client VPN 엔드포인트 구성 파일 다운로드

구성 파일을 다운로드하려면

1. Client VPN 엔드포인트 구성 파일 다운로드에서 다음 빠른 단계에 따라 Client VPN 구성 파일을 다운로드합니다.
2. 이 단계에서는 Client VPN 엔드포인트 DNS 이름 앞에 문자열을 추가하라는 메시지가 표시됩니다. 다음은 그 예입니다:

   - Example 엔드포인트 DNS 이름

     Client VPN 엔드포인트 DNS 이름이 다음과 같은 경우:

     ```
     remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
     ```

     다음과 같이 Client VPN 엔드포인트를 식별하는 문자열을 추가할 수 있습니다.

     ```
     remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-
     west-1.amazonaws.com 443
     ```

3. 이 단계에서 새 <cert></cert> 태그 세트 사이에 클라이언트 인증서의 내용을 추가하고 새 <key></key> 태그 세트 사이에 프라이빗 키의 내용을 추가하라는 메시지가 표시됩니다. 다음은 그 예입니다:

a. 명령 프롬프트를 열고 디렉터리를 클라이언트 인증서 및 프라이빗 키가 있는 위치로 변경합니다.

b. Example macOS client1.domain.tld.crt

macOS에서 `client1.domain.tld.crt` 파일 내용을 표시하려면 `cat client1.domain.tld.crt`을(를) 사용할 수 있습니다.

터미널에서 값을 복사하고 `downloaded-client-config.ovpn`을(를) 다음과 같이 붙여넣습니다.

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

c. Example macOS client1.domain.tld.key

`client1.domain.tld.key`의 내용을 표시하려면 `cat client1.domain.tld.key`을(를) 사용할 수 있습니다.

터미널에서 값을 복사하고 `downloaded-client-config.ovpn`을(를) 다음과 같이 붙여넣습니다.

```
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>
```

8단계: AWS Client VPN에 연결

AWS Client VPN 클라이언트는 무료로 제공됩니다. 컴퓨터를 AWS Client VPN에 직접 연결하여 종단 간 VPN 환경을 이용할 수 있습니다.

Client VPN에 연결하려면

1. 데스크톱용 AWS Client VPN 다운로드 및 설치
2. AWS Client VPN을 엽니다.
3. VPN 클라이언트 메뉴에서 파일, 관리 프로필을 선택합니다.
4. 프로필 추가를 선택한 다음 `downloaded-client-config.ovpn`을(를) 선택합니다.
5. 이름 표시에 설명이 포함된 이름을 입력합니다.
6. 프로필 추가, 완료를 선택합니다.
7. 연결(을 선택합니다.

Client VPN에 연결한 후 Amazon VPC의 리소스를 보려면 다른 VPN과의 연결을 끊어야 합니다.

> **Note**
>
> 클라이언트를 종료하고 다시 시작해야 연결할 수 있을 것입니다.

다음 단계

- Amazon Managed Workflows for Apache Airflow 시작하기에서 Amazon MWAA 환경을 만드는 방법을 알아봅니다. Client VPN과 동일한 AWS 리전에 환경을 만들고 Client VPN과 동일한 VPC, 프라이빗 서브넷 및 보안 그룹을 사용해야 합니다.

튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성

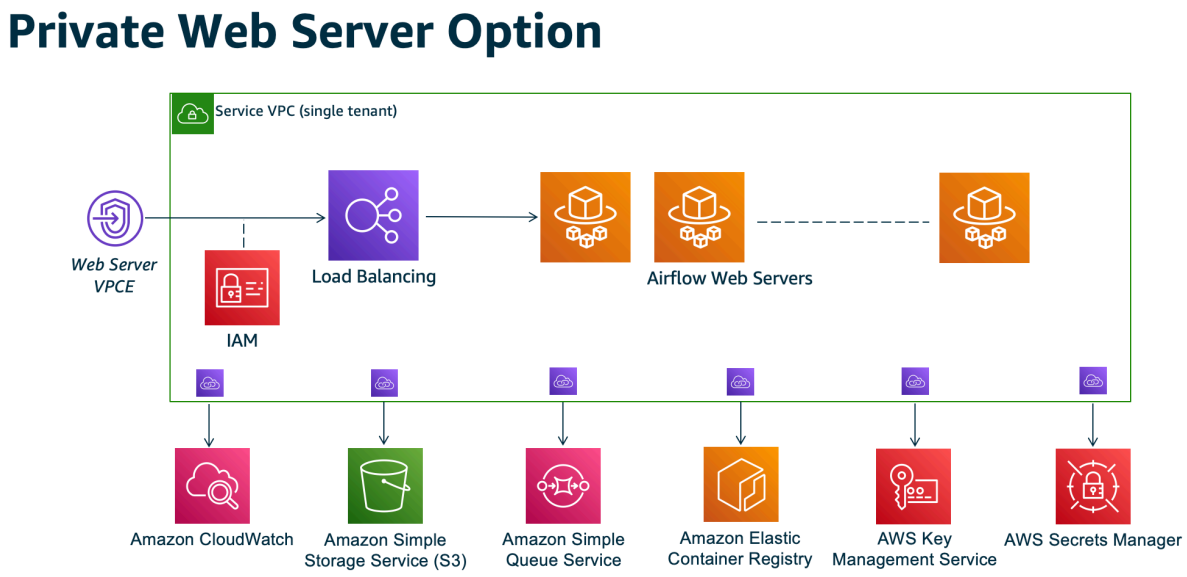
이 튜토리얼에서는 Amazon Managed Workflows for Apache Airflow 환경을 위해 컴퓨터에서 Apache Airflow 웹 서버까지 연결되는 SSH 터널을 생성하는 단계를 안내합니다. 여기서는 Amazon MWAA 환경을 이미 만들었다고 가정합니다. 일단 설정이 완료되면 Linux Bastion Host가 점프 서버 역할을 하여

컴퓨터와 VPC의 리소스를 안전하게 연결합니다. 그런 다음 SOCKS 프록시 관리 애드온을 사용하여 브라우저의 프록시 설정을 제어하여 Apache Airflow UI에 액세스할 수 있습니다.

Sections

프라이빗 네트워크

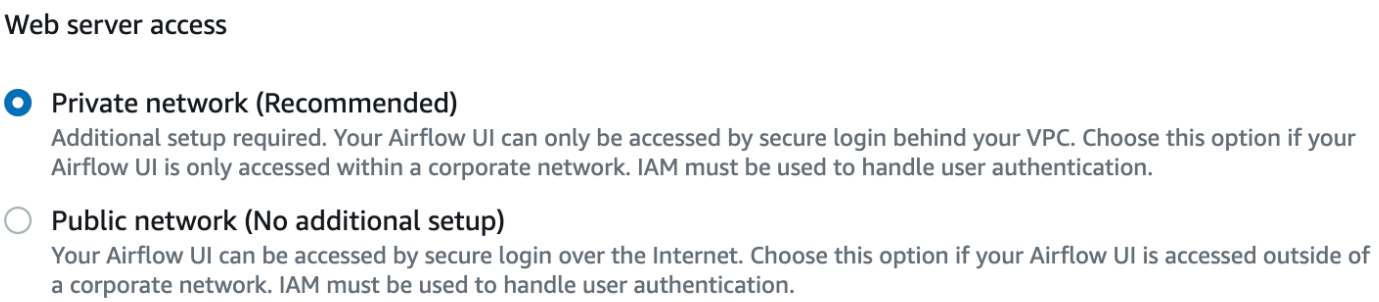
이 튜토리얼에서는 Apache Airflow 웹 서버의 프라이빗 네트워크 액세스 모드를 선택했다고 가정합니다.



프라이빗 네트워크 액세스 모드를 사용하면 Amazon VPC 내에서 사용자 환경의 IAM 정책에 대한 액세스 권한을 부여 받은 사용자만 Apache Airflow UI에 액세스할 수 있도록 제한됩니다.

프라이빗 웹 서버 액세스가 가능한 환경을 만들 때는 모든 종속성을 Python 휠 아카이브(`.whl`)에서 패키징한 다음 `requirements.txt`에서 `.whl`을 참조해야 합니다. 휠을 사용하여 종속성을 패키징하고 설치하는 방법에 대한 지침은 Python 휠을 이용한 종속성 관리를 참조하십시오.

다음 이미지는 Amazon MWAA 콘솔에서 프라이빗 네트워크 옵션을 찾을 수 있는 위치를 보여줍니다.



사용 사례

Amazon MWAA 환경을 만든 후에 이 튜토리얼을 사용할 수 있습니다. 반드시 사용자 환경과 동일한 Amazon VPC, VPC 보안 그룹 및 퍼블릭 서브넷을 사용해야 합니다.

시작하기 전 준비 사항

1. 사용자 권한을 확인합니다. AWS Identity and Access Management (IAM) 의 계정에 VPC 리소스를 생성하고 관리할 수 있는 충분한 권한이 있는지 확인하십시오.
2. Amazon MWAA VPC를 사용합니다. 이 튜토리얼은 Bastion Host를 기존 VPC에 연결한다고 가정합니다. Amazon VPC는 Amazon MWAA 환경과 동일한 리전에 있어야 하며 VPC 네트워크 생성에서 정의한 대로 두 개의 프라이빗 서브넷이 있어야 합니다.
3. SSH 키를 생성합니다, 가상 서버에 연결하려면 Amazon MWAA 환경과 동일한 리전에 Amazon EC2 SSH 키(.pem)를 생성해야 합니다. SSH 키가 없는 경우 Amazon EC2 사용 설명서의 키 페어 생성 또는 가져오기를 참조하십시오.

목표

이 튜토리얼에서는 다음을 수행합니다.

1. 기존 VPC용AWS CloudFormation 템플릿을 사용하여 Linux Bastion Host 인스턴스를 생성합니다.

2. 포트 22에 대한 수신 규칙을 사용하여 Bastion 인스턴스의 보안 그룹에 대한 인바운드 트래픽을 승인합니다.
3. Amazon MWAA 환경의 보안 그룹에서 Bastion 인스턴스의 보안 그룹에 대한 인바운드 트래픽을 승인합니다.
4. Bastion 인스턴스로 연결되는 SSH 터널을 생성합니다.
5. Apache Airflow UI를 볼 수 있도록 Firefox 브라우저용 FoxyProxy 애드온을 설치 및 구성하십시오.

1단계: Bastion 인스턴스 생성

다음 섹션에서는 콘솔의 기존 VPC용AWS CloudFormation 템플릿을 사용하여 linux bastion 인스턴스를 만드는 단계를 설명합니다. AWS CloudFormation

Linux Bastion Host를 만들려면

1. 콘솔에서 퀵 스타트 배포 페이지를 엽니다. AWS CloudFormation
2. 탐색 표시줄의 지역 선택기를 사용하여 Amazon MWAA 환경과 동일한 AWS 지역을 선택합니다.
3. 다음을 선택합니다.
4. 스택 이름 텍스트 필드에 `mwaa-linux-bastion`와(과) 같이 이름을 입력합니다.
5. 파라미터, 네트워크 구성 창에서 다음 옵션을 선택합니다.

    a. Amazon MWAA 환경의 VPC ID를 선택합니다.

    b. Amazon MWAA 환경의 퍼블릭 서브넷 1 ID를 선택합니다.

    c. Amazon MWAA 환경의 퍼블릭 서브넷 2 ID를 선택합니다.

    d. 허용된 Bastion 외부 액세스 CIDR에 가능한 가장 좁은 주소 범위(예: 내부 CIDR 범위)를 입력합니다.

    > **Note**
    >
    > 범위를 식별하는 가장 간단한 방법은 퍼블릭 서브넷과 동일한 CIDR 범위를 사용하는 것입니다. 예를 들어, 페이지에 있는 AWS CloudFormation 템플릿의 퍼블릭 서브넷은 및 입니다. VPC 네트워크 생성 `10.192.10.0/24` `10.192.11.0/24`

6. Amazon EC2 구성 창에서 다음을 선택합니다.

    a. 키 쌍 이름의 드롭다운 목록에서 SSH 키를 선택합니다.

    b. Bastion Host 이름에 이름을 입력합니다.

c. TCP 전달에서 true를 선택합니다.

> **Warning**
> 이 단계에서는 TCP 전달을 true로 설정해야 합니다. 그렇지 않으면 다음 단계에서 SSH 터널을 생성할 수 없습니다.

7. 다음, 다음을 선택합니다.
8. 승인을 선택한 다음 스택 생성을 선택합니다.

Linux Bastion 호스트의 아키텍처에 대해 자세히 알아보려면 AWS 클라우드의 Linux Bastion 호스트: 아키텍처를 참조하십시오.

2단계: SSH 터널 생성

다음 단계에서는 Linux Bastion에 대한 SSH 터널을 생성하는 방법을 설명합니다. SSH 터널은 로컬 IP 주소로부터 Linux Bastion으로의 요청을 수신합니다. 이것이 이전 단계에서 Linux Bastion에 대한 TCP 전달이 `true`(으)로 설정된 이유입니다.

macOS/Linux

명령줄을 통해 터널을 만들려면

1. Amazon EC2 콘솔에서 인스턴스 페이지를 엽니다.
2. 인스턴스를 선택합니다.
3. Public IPv4 DNS에서 주소를 복사합니다. 예를 들어 `ec2-4-82-142-1.compute-1.amazonaws.com`입니다.
4. 명령 프롬프트에서 SSH 키가 저장된 디렉터리로 이동합니다.
5. 다음 명령을 실행하여 SSH를 사용하여 bastion 인스턴스에 연결합니다. 샘플 값을 `mykeypair.pem`의 SSH 키 이름으로 대체합니다.

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

PuTTY를 사용하여 터널을 생성하려면

1. Amazon EC2 콘솔에서 인스턴스 페이지를 엽니다.
2. 인스턴스를 선택합니다.
3. Public IPv4 DNS에서 주소를 복사합니다. 예를 들어 `ec2-4-82-142-1.compute-1.amazonaws.com`입니다.
4. PuTTY를 열고 세션을 선택합니다.
5. 호스트 이름에 호스트 이름을 ec2-user@*YOUR_PUBLIC_IPV4_DNS*로 입력하고 포트는 22(으)로 입력합니다.
6. SSH 탭을 확장하고 인증을 선택합니다. 인증을 위한 프라이빗 키 파일에서 로컬 'ppk' 파일을 선택합니다.
7. SSH에서 터널 탭을 선택한 다음 동적 및 자동 옵션을 선택합니다.
8. 소스 포트에서 8157 포트(또는 사용하지 않은 다른 포트)를 추가한 다음 대상 포트를 비워 둡니다. 추가를 선택합니다.
9. 세션 탭을 선택하고 세션 이름을 입력합니다. 예를 들어 `SSH Tunnel`입니다.
10. 저장, 열기 를 선택합니다.

> **Note**
>
> 퍼블릭 키의 암호 문구를 입력해야 할 수도 있습니다.

> **Note**
>
> `Permission denied (publickey)`오류가 발생하는 경우 AWSSupport-TroubleoTSSH 도구를 사용하고 이 자동화 실행 (콘솔) 을 선택하여 SSH 설정 문제를 해결하는 것이 좋습니다.

3단계: Bastion 보안 그룹을 인바운드 규칙으로 구성

서버에 대한 액세스 및 서버에서의 정기적인 인터넷 액세스는 해당 서버에 연결된 특수 유지 관리 보안 그룹을 통해 허용됩니다. 다음 단계는 Bastion 보안 그룹을 환경의 VPC 보안 그룹에 대한 트래픽의 인바운드 소스로 구성하는 방법을 설명합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 네트워킹 패널에서 VPC 보안 그룹을 선택합니다.
4. 인바운드 규칙 편집을 선택합니다.
5. 다른 규칙 추가를 선택합니다.
6. 소스 드롭다운 목록에서 VPC 보안 그룹 ID를 선택합니다.
7. 나머지 옵션은 비워 두거나 기본값으로 설정합니다.
8. 규칙 저장을 선택합니다.

4단계: Apache Airflow URL 복사

다음 단계는 Amazon MWAA 콘솔을 열고 URL을 Apache Airflow UI에 복사하는 방법을 설명합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 후속 단계를 위해 Airflow UI의 URL을 복사합니다.

5단계: 프록시 설정 구성

동적 포트 전달과 함께 SSH 터널을 사용하는 경우 SOCKS 프록시 관리 추가 기능을 사용하여 브라우저의 프록시 설정을 제어해야 합니다. 예를 들어 Chromium의 `--proxy-server` 기능을 사용하여 브라우저 세션을 시작하거나 Mozilla 브라우저에서 확장 프로그램을 사용할 수 있습니다. FoxyProxy FireFox

옵션 1: 로컬 포트 전달을 사용하여 SSH 터널 설정

SOCKS 프록시를 사용하지 않을 경우 로컬 포트 전달을 사용하여 SSH 터널을 설정할 수 있습니다. 다음 예제 명령은 로컬 포트 8157을 통해 트래픽을 전달하여 Amazon ResourceManagerEC2 웹 인터페이스에 액세스합니다.

1. 새 명령 프롬프트 창을 엽니다.
2. 다음 명령을 입력하여 SSH 터널을 엽니다.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-
vpce.YOUR_REGION.airflow.amazonaws.com:443
 ubuntu@YOUR_PUBLIC_IPV4_DNS.YOUR_REGION.compute.amazonaws.com
```

-L은 로컬 포트 전송의 사용을 나타내며, 이를 통해 노드의 로컬 웹 서버에서 식별된 원격 포트에 데이터를 전송하는 데 사용되는 로컬 포트를 지정할 수 있습니다.

3. 브라우저에 `http://localhost:8157/`을 입력합니다.

> **Note**
>
> `https://localhost:8157/`을 사용해야 할 수도 있습니다.

옵션 2: 명령줄을 통한 프록시

대부분의 웹 브라우저에서 명령줄 또는 구성 파라미터를 통해 프록시를 구성할 수 있습니다. 예를 들어 Chromium을 사용하면 다음 명령을 사용하여 브라우저를 시작할 수 있습니다.

```
chromium --proxy-server="socks5://localhost:8157"
```

그러면 이전 단계에서 만든 SSH 터널을 사용하여 요청을 프록시하는 브라우저 세션이 시작됩니다. 다음과 같이 프라이빗 Amazon MWAA 환경 URL(https:// 사용)을 열 수 있습니다.

```
https://YOUR_VPC_ENDPOINT_ID-vpce.YOUR_REGION.airflow.amazonaws.com/home.
```

옵션 3: 모질라 파이어폭스용 프록시 FoxyProxy

다음 예제는 모질라 파이어폭스의 FoxyProxy 표준 (버전 7.5.1) 구성을 보여줍니다. FoxyProxy 프록시 관리 도구 세트를 제공합니다. 이를 통해 Apache Airflow UI에서 사용하는 도메인에 해당하는 패턴과 일치하는 URL용 프록시 서버를 사용할 수 있습니다.

1. Firefox에서 FoxyProxy 표준 확장 페이지를 엽니다.
2. Firefox에 추가를 선택합니다.
3. 추가를 선택합니다.
4. 브라우저 툴바에서 FoxyProxy 아이콘을 선택하고 옵션을 선택합니다.
5. 다음 코드를 복사하고 로컬에 `mwaa-proxy.json`(으)로 저장합니다. *YOUR_HOST_NAME*의 샘플 값을 Apache Airflow URL로 대체합니다.

```
{
  "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. FoxyProxy 6.0 이상에서 설정 가져오기 패널에서 설정 가져오기를 선택하고 파일을 선택합니다. `mwaa-proxy.json`
7. 확인을 선택합니다.

6단계: Apache Airflow UI 열기

다음 단계에서는 Apache Airflow UI를 여는 방법을 설명합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. Airflow UI 열기를 선택합니다.

다음 단계

- Apache Airflow CLI 명령 참조의 Bastion Host에 대한 SSH 터널에서 Airflow CLI 명령을 실행하는 방법을 알아봅니다.
- DAG 추가 또는 업데이트에서 Amazon S3 버킷에 DAG 코드를 업로드하는 방법을 알아봅니다.

튜토리얼: DAG의 하위 집합에 대한 Amazon MWAA 사용자 액세스 제한

Amazon MWAA는 IAM 보안 주체를 하나 이상의 Apache Airflow 기본 역할에 매핑하여 환경에 대한 액세스를 관리합니다. 다음 튜토리얼에서는 개별 Amazon MWAA 사용자가 특정 DAG 또는 DAG 세트만 보고 상호 작용하도록 제한하는 방법을 보여줍니다.

> **Note**
>
> IAM 역할을 가정할 수 있는 한, 이 튜토리얼의 단계는 페더레이션 액세스를 사용하여 완료할 수 있습니다.

주제

필수 조건

이 자습서의 단계를 완료하려면 다음이 필요합니다.

- 여러 DAG가 있는 Amazon MWAA 환경
- AdministratorAccess권한이 `Admin` 있는 IAM 보안 주체와 DAG 액세스를 제한할 수 있는 보안 주체인 IAM 사용자 `MWAAUser` 관리자 역할에 대한 자세한 내용은 IAM 사용 설명서의 관리자 직무 기능을 참조하십시오.

> **Note**
>
> 권한 정책을 IAM 사용자에게 직접 연결하지 마십시오. 사용자가 Amazon MWAA 리소스에 임시로 액세스할 수 있도록 위임할 수 있는 IAM 역할을 설정하시기 바랍니다.

- AWS Command Line Interface 버전 2가 설치되었습니다.

1단계: 기본 **`Public`** Apache Airflow 역할을 사용하여 Amazon MWAA 웹 서버에 IAM 보안 주체에 대한 액세스 권한을 제공합니다.

를 사용하여 권한을 부여하려면 AWS Management Console

1. `Admin`역할로 AWS 계정에 로그인하고 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 사용자를 선택한 후 사용자 테이블에서 Amazon MWAA IAM 사용자를 선택합니다.
3. 사용자 세부 정보 페이지의 요약에서 권한 탭을 선택한 다음 권한 정책을 선택하여 카드를 확장하고 권한 추가를 선택합니다.

4. 권한 설정 섹션에서 기존 정책 직접 연결을 선택한 후 정책 생성을 선택하여 사용자 지정 권한 정책을 생성하고 연결합니다.
5. 정책 생성 페이지에서 JSON을 선택한 후, 정책 편집기에 다음 JSON 권한 정책을 복사하여 붙여 넣습니다. 이 정책은 기본 `Public` Apache Airflow 역할을 가진 사용자에게 웹 서버 액세스 권한을 부여합니다.

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

 "arn:aws:airflow:YOUR_REGION:YOUR_ACCOUNT_ID:role/YOUR_ENVIRONMENT_NAME/Public"
            ]
        }
    ]
 }
```

2단계: 새 Apache Airflow 사용자 지정 역할 생성

Apache Airflow UI를 사용하여 새 역할을 만들려면

1. 관리자 IAM 역할을 사용하여 Amazon MWAA 콘솔을 열고 사용자 환경의 Apache Airflow UI를 시작합니다.
2. 상단의 탐색 창에서 보안에 마우스오버하여 드롭다운 목록을 연 다음 역할 목록을 선택하여 기본 Apache Airflow 역할을 확인합니다.
3. 역할 목록에서 사용자를 선택한 다음 페이지 상단에서 작업을 선택하여 드롭다운을 엽니다. 역할 복사를 선택하고 확인을 클릭합니다

> **Note**
>
> Ops 또는 Viewer 역할을 복사하여 각각 액세스 권한을 늘리거나 줄일 수 있습니다.

4. 테이블에서 생성한 새 역할을 찾아 레코드 편집을 선택합니다.
5. 역할 전환 페이지에서 다음을 수행합니다.

- 이름의 텍스트 필드에 역할의 새 이름을 입력합니다. 예를 들어 **Restricted**입니다.
- 권한 목록에서 액세스를 제공하려는 DAG 집합의 읽기 및 쓰기 권한을 제거한 `can read on DAGs` 다음 추가하십시오. `can edit on DAGs` 예를 들어 DAG `example_dag.py`의 경우 **can read on DAG:*example_dag*** 및 **can edit on DAG:*example_dag***을(를) 추가합니다.

저장을 선택합니다. 이제 Amazon MWAA 환경에서 사용할 수 있는 DAG의 하위 집합에 대한 액세스를 제한하는 새 역할이 생겼을 것입니다. 이제 모든 기존 Apache Airflow 사용자에게 이 역할을 할당할 수 있습니다.

3단계: 생성한 역할을 Amazon MWAA 사용자에게 할당합니다

새 역할을 할당하려면

1. `MWAAUser`에 대한 액세스 보안 인증을 사용하여 다음 CLI 명령을 실행하고 환경의 웹 서버 URL을 검색합니다.

```
$ aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq
 '.Environment.WebserverUrl'
```

성공하면 다음과 같은 결과가 출력됩니다.

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. 에 `MWAAUser` AWS Management Console로그인한 후 새 브라우저 창을 열고 다음 URL에 액세스합니다. `Webserver-URL`를 자신의 정보로 대체합니다.

```
https://<Webserver-URL>/home
```

성공하면 `MWAAUser`에게 아직 Apache Airflow UI에 대한 액세스 권한이 부여되지 않았으므로 `Forbidden` 오류 페이지가 표시됩니다.

3. 에 `Admin` 로그인한 후 AWS Management Console Amazon MWAA 콘솔을 다시 열고 사용자 환경의 Apache 에어플로우 UI를 시작합니다.
4. UI 대시보드에서 보안 드롭다운을 확장하고 이번에는 사용자 목록을 선택합니다.

5. 사용자 테이블에서 새 Apache Airflow 사용자를 찾아 레코드 편집을 선택합니다. 사용자의 이름은 `user/mwaa-user`의 패턴으로 IAM 사용자 이름과 일치합니다.
6. 사용자 편집 페이지의 역할 섹션에서 생성한 새 사용자 지정 역할을 추가한 다음 저장을 선택합니다.

> **Note**
>
> 성 필드는 필수이지만 스페이스를 입력해도 충분합니다.

IAM `Public` 보안 주체는 `MWAAUser`이(가) Apache Airflow UI에 액세스할 수 있는 권한을 부여하는 반면, 새 역할은 DAG를 보는 데 필요한 추가 권한을 제공합니다.

> **Important**
>
> Apache Airflow UI를 사용하여 추가한 IAM에서 승인하지 않은 5가지 기본 역할(예:`Admin`)은 다음 사용자 로그인 시 제거됩니다.

다음 단계

- Amazon MWAA 환경 액세스 관리에 대해 자세히 알아보고 환경 사용자를 위해 사용할 수 있는 샘플 JSON IAM 정책을 보려면 the section called “Amazon MWAA 환경 액세스” 단원을 참조하십시오.

관련 리소스

- 액세스 제어 (Apache Airflow 설명서) — Apache Airflow 설명서 웹사이트에서 기본 Apache Airflow 역할에 대해 자세히 알아봅니다.

자습서: Amazon MWAA에서 자체 환경 엔드포인트를 자동으로 관리합니다.

를 AWS Organizations사용하여 리소스를 공유하는 여러 AWS 계정을 관리하는 경우 Amazon MWAA를 사용하면 자체 Amazon VPC 엔드포인트를 생성하고 관리할 수 있습니다. 즉, 환경에 필요한 리소스에만 액세스를 허용하는 더 엄격한 보안 정책을 사용할 수 있습니다.

공유 Amazon VPC에서 환경을 만들 때 기본 Amazon VPC를 소유한 계정 (소유자) 은 Amazon MWAA에 필요한 두 개의 프라이빗 서브넷을 동일한 조직에 속한 다른 계정 (참가자) 과 공유합니다. 그런 다음 해당 서브넷을 공유하는 참가자 계정은 공유 VPC에서 환경을 보고, 만들고, 수정하고, 삭제할 수 있습니다.

공유 또는 정책이 제한된 Amazon VPC에서 환경을 생성하면 Amazon MWAA가 먼저 서비스 VPC 리소스를 생성한 다음 최대 72시간 동안 상태를 입력합니다. `PENDING`

환경 상태가 에서 `CREATING` 로 `PENDING` 변경되면 Amazon MWAA는 상태 변경에 대한 Amazon에 EventBridge 알림을 보냅니다. 이를 통해 소유자 계정은 Amazon MWAA 콘솔 또는 API의 엔드포인트 서비스 정보를 기반으로 참여자를 대신하여 또는 프로그래밍 방식으로 필요한 엔드포인트를 생성할 수 있습니다. 다음에서는 Lambda 함수와 EventBridge Amazon MWAA 상태 변경 알림을 수신하는 규칙을 사용하여 새 Amazon VPC 엔드포인트를 생성합니다.

여기서는 환경과 동일한 Amazon VPC에 새 엔드포인트를 생성합니다. 공유 Amazon VPC를 설정하려면 소유자 계정에서 EventBridge 규칙 및 Lambda 함수를 생성하고 참가자 계정에서 Amazon MWAA 환경을 생성하십시오.

주제

사전 조건

이 자습서의 단계를 완료하려면 다음이 필요합니다.

- ...

아마존 VPC 생성

다음 AWS CloudFormation 템플릿과 AWS CLI 명령을 사용하여 새 Amazon VPC를 생성합니다. 템플릿은 Amazon VPC 리소스를 설정하고 특정 대기열에 대한 액세스를 제한하도록 엔드포인트 정책을 수정합니다.

1. AWS CloudFormation템플릿을 다운로드한 다음 파일의 압축을 풉니다. `.yml`
2. 새 명령 프롬프트 창에서 템플릿을 저장한 폴더로 이동한 다음 `create-stack`사용하여 스택을 생성합니다. `--template-body`플래그는 템플릿의 경로를 지정합니다.

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

다음 섹션에서는 Lambda 함수를 생성해 보겠습니다.

Lambda 함수 생성

다음 Python 코드와 IAM JSON 정책을 사용하여 새 Lambda 함수 및 실행 역할을 생성합니다. 이 함수는 프라이빗 아파치 에어플로우 웹 서버와 Amazon SQS 대기열을 위한 Amazon VPC 엔드포인트를 생성합니다. Amazon MWAA는 환경을 확장할 때 Amazon SQS를 사용하여 여러 작업자 사이에서 Celery를 사용하여 작업을 대기열에 추가합니다.

1. Python 함수 코드를 다운로드하십시오.
2. IAM 권한 정책을 다운로드한 다음 파일의 압축을 풉니다.
3. 명령 프롬프트를 연 다음 JSON 권한 정책을 저장한 폴더로 이동합니다. IAM `create-role`명령어를 사용하여 새 역할을 생성합니다.

```
$ aws iam create-role --role-name function-role \
--assume-role-policy-document file://lambda-mwaa-vpce-policy.json
```

응답의 역할 ARN을 기록해 AWS CLI 둡니다. 다음 단계에서는 ARN을 사용하여 이 새 역할을 함수의 실행 역할로 지정합니다.

4. 함수 코드를 저장한 폴더로 이동한 다음 `create-function`명령을 사용하여 새 함수를 생성합니다.

```
$ 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
```

응답의 함수 ARN을 기록해 AWS CLI 둡니다. 다음 단계에서는 ARN을 지정하여 함수를 새 EventBridge 규칙의 대상으로 구성합니다.

다음 섹션에서는 환경이 상태에 들어갈 때 이 함수를 호출하는 EventBridge 규칙을 만들 것입니다. `PENDING`

규칙 만들기 EventBridge

Amazon MWAA 알림을 수신하고 새 Lambda 함수를 대상으로 하는 새 규칙을 생성하려면 다음을 수행하십시오.

1. EventBridge `put-rule`명령을 사용하여 새 규칙을 생성합니다. EventBridge

   ```
   $ aws events put-rule --name "mwaa-lambda-rule" \
   --event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment
    Status Change\"]}"
   ```

   이벤트 패턴은 환경 상태가 변경될 때마다 Amazon MWAA가 보내는 알림을 수신합니다.

   ```
   {
     "source": ["aws.airflow"],
     "detail-type": ["MWAA Environment Status Change"]
   }
   ```

2. `put-targets`명령을 사용하여 Lambda 함수를 새 규칙의 대상으로 추가합니다.

   ```
   $ aws events put-targets --rule "mwaa-lambda-rule" \
   --targets "Id"="1","Arn"="arn:aws::lambda:region:123456789012:function:mwaa-vpce-
   lambda"
   ```

이제 고객 관리형 Amazon VPC 엔드포인트를 사용하여 새로운 Amazon MWAA 환경을 만들 준비가 되었습니다.

아마존 MWAA 환경 만들기

Amazon MWAA 콘솔을 사용하여 고객 관리형 Amazon VPC 엔드포인트가 있는 새 환경을 만들 수 있습니다.

1. Amazon MWAA 콘솔을 열고 환경 생성을 선택합니다.
2. 이름에 고유한 이름을 입력합니다.
3. Airflow 버전의 경우 최신 버전을 선택하십시오.

4. Amazon S3 버킷과 DAGs 폴더 (예: 환경과 함께 사용할) `dags/` 를 선택한 후 [Next] 를 선택합니다.
5. 고급 설정 구성 페이지에서 다음을 수행하십시오.
   a. 가상 사설 클라우드의 경우 이전 단계에서 생성한 Amazon VPC를 선택합니다.
   b. 웹 서버 액세스의 경우 공용 네트워크 (인터넷 액세스 가능) 를 선택합니다.
   c. 보안 그룹의 경우 생성할 때 사용한 보안 그룹을 선택합니다AWS CloudFormation. 이전 단계의 AWS PrivateLink 엔드포인트에 대한 보안 그룹은 자체 참조이므로 환경에 맞는 동일한 보안 그룹을 선택해야 합니다.
   d. 엔드포인트 관리의 경우 고객 관리형 엔드포인트를 선택합니다.
6. 나머지 기본 설정을 유지하고 다음을 선택합니다.
7. 선택 항목을 검토한 다음 환경 만들기를 선택합니다.

> **Tip**
>
> 새 환경 설정에 대한 자세한 내용은 Amazon MWAA 시작하기를 참조하십시오.

환경인 경우 Amazon MWAA는 규칙에 설정한 이벤트 패턴과 일치하는 알림을 보냅니다. `PENDING` 규칙은 Lambda 함수를 호출합니다. 함수는 알림 이벤트를 파싱하여 웹 서버 및 Amazon SQS 대기열에 필요한 엔드포인트 정보를 가져옵니다. 그런 다음 Amazon VPC에 엔드포인트를 생성합니다.

엔드포인트를 사용할 수 있게 되면 Amazon MWAA가 환경 생성을 재개합니다. 준비가 되면 환경 상태가 로 `AVAILABLE` 변경되고 Amazon MWAA 콘솔을 사용하여 Apache Airflow 웹 서버에 액세스할 수 있습니다.

Amazon Managed Workflows for Apache Airflow용 코드 예제

이 가이드에는 Amazon Managed Workflows for Apache Airflow 환경에서 사용할 수 있는 DAG 및 사용자 지정 플러그인을 비롯한 코드 샘플이 포함되어 있습니다. AWS 서비스와 함께 Apache Airflow를 사용하는 예제를 더 보려면 Apache Airflow 저장소의 `dags`디렉토리를 참조하십시오. GitHub

샘플

- DAG를 사용하여 CLI에서 변수 가져오기
- SSHOperator을(를) 사용하여 SSH 연결 생성
- Apache Airflow Snowflake 연결에 AWS Secrets Manager의 암호 키 사용
- CloudWatch에서 DAG를 사용하여 사용자 지정 지표 작성
- Amazon MWAA 환경에서 Aurora PostgreSQL 데이터베이스 정리
- Amazon S3의 CSV 파일로 환경 메타데이터 내보내기
- Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용
- Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용
- Oracle을 사용하여 사용자 지정 플러그인 생성
- 런타임 환경 변수를 생성하는 사용자 지정 플러그인 생성
- Amazon MWAA에서 DAG 시간대 변경
- CodeArtifact 토큰 새로고침
- Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인 생성
- Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성
- Lambda 함수를 사용한 DAG 호출
- 여러 Amazon MWAA 환경에서 DAG 호출
- Amazon RDS for Microsoft SQL Server와 함께 Amazon MWAA 사용
- Amazon EMR과 함께 Amazon MWAA 사용
- Amazon EKS에서 Amazon MWAA 사용
- ECSOperator를 사용하여 Amazon ECS에 연결
- Amazon MWAA에서 dbt 사용
- AWS 블로그 및 튜토리얼

DAG를 사용하여 CLI에서 변수 가져오기

다음 샘플 코드는 Amazon Managed Workflows for Apache Airflow의 CLI를 사용하여 변수를 가져옵니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

권한

AWS계정이 `AmazonMWAAAirflowCliAccess` 정책에 액세스할 수 있어야 합니다. 자세한 내용은 아파치 에어플로우 CLI 정책: 아마존MWAA 액세스 AirflowCli 단원을 참조하십시오.

종속성

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 샘플

다음 샘플 코드는 Amazon MWAA 환경 이름(`mwaa_env`), 환경의 AWS 리전(`var_file`), 가져오려는 변수가 포함된 로컬 파일(`aws_region`) 등 세 가지 입력을 사용합니다.

```
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)
```

다음 단계

- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 dags 폴더에 업로드하는 방법을 알아봅니다.

SSHOperator을(를) 사용하여 SSH 연결 생성

다음 예제는 DAG(유방향 비순환 그래프)에서 SSHOperator을(를) 사용하여 Amazon Managed Workflows for Apache Airflow 환경에서 원격 Amazon EC2 인스턴스에 연결하는 방법을 설명합니다. 비슷한 접근 방식을 사용하여 SSH 액세스가 있는 모든 원격 인스턴스에 연결할 수 있습니다.

다음 예제에서는 Amazon S3에 있는 사용자 환경의 dags 디렉터리에 SSH 암호 키(.pem)를 업로드합니다. 그런 다음 requirements.txt을(를) 사용하여 필요한 종속성을 설치하고 UI에 새 Apache Airflow 연결을 생성합니다. 마지막으로 원격 인스턴스에 대한 SSH 연결을 생성하는 DAG를 작성합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.
- SSH 암호 키. 코드 샘플은 Amazon MWAA 환경과 동일한 리전에 Amazon EC2 인스턴스와 `.pem`이(가) 있다고 가정합니다. 키가 없는 경우 Amazon EC2 사용 설명서의 키 페어 생성 또는 가져오기를 참조하십시오.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

요구 사항

웹 서버에 `apache-airflow-providers-ssh` 패키지를 설치하려면 다음 파라미터를 `requirements.txt`에 추가합니다. 환경이 업데이트되고 Amazon MWAA가 종속성을 성공적으로 설치하면 UI에 새 SSH 연결 유형이 표시됩니다.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
```

```
apache-airflow-providers-ssh
```

> **Note**
>
> `-c`이(가) `requirements.txt`의 제약 조건 URL을 정의합니다. 이렇게 하면 Amazon MWAA가 사용자 환경에 맞는 올바른 패키지 버전을 설치할 수 있습니다.

암호 키를 Amazon S3에 복사

다음 AWS Command Line Interface 명령을 사용하여 Amazon S3의 환경 `dags` 디렉터리에 `.pem` 키를 복사합니다.

```
$ aws s3 cp your-secret-key.pem s3://your-bucket/dags/
```

Amazon MWAA는 `.pem` 키를 포함해 `dags`의 콘텐츠를 로컬 `/usr/local/airflow/dags/` 디렉터리에 복사합니다. 이렇게 하면 Apache Airflow가 키에 액세스할 수 있습니다.

새 Apache Airflow 연결 생성

Apache Airflow UI를 사용하여 새 SSH 연결을 생성하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경 목록에서 사용자 환경에 맞는 Open Airflow UI를 선택합니다.
3. Apache Airflow UI 페이지의 상단 내비게이션 바에서 관리자를 선택하여 드롭다운 목록을 확장한 다음 연결을 선택합니다.
4. 연결 목록 페이지에서 +를 선택하거나 새 레코드 추가 버튼을 선택하여 새 연결을 추가합니다.
5. 연결 추가 페이지에서 다음 정보를 추가합니다.
   a. 연결 ID에 **ssh_new**를 입력합니다.
   b. 연결 유형의 경우 드롭다운 목록에서 SSH를 선택합니다.

> **Note**
>
> 목록에 SSH 연결 유형이 없는 경우 Amazon MWAA가 필요한 `apache-airflow-providers-ssh` 패키지를 설치하지 않은 것입니다. 이 패키지를 포함하도록 `requirements.txt` 파일을 업데이트한 다음 다시 시도하십시오.

c. Host에 연결하려는 Amazon EC2 인스턴스의 IP 주소를 입력합니다. 예를 들어 **12.345.67.89**입니다.

d. Amazon EC2 인스턴스에 연결 중인 경우 사용자 이름에 **ec2-user**을(를) 입력합니다. Apache Airflow를 연결하려는 원격 인스턴스의 유형에 따라 사용자 이름이 달라질 수 있습니다.

e. Extra에는 다음 키-값 쌍을 JSON 형식으로 입력합니다.

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

이 키-값 쌍은 Apache Airflow가 로컬 /dags 디렉터리에서 암호 키를 찾도록 지시합니다.

코드 샘플

다음 DAG는 SSHOperator을(를) 사용하여 대상 Amazon EC2 인스턴스에 연결한 다음 hostname Linux 명령을 실행하여 인스턴스 이름을 인쇄합니다. 원격 인스턴스에서 모든 명령 또는 스크립트를 실행하도록 DAG를 수정할 수 있습니다.

1. 터미널을 열고 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 ssh.py로 저장합니다.

```
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. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 성공하면 ssh_operator_example DAG의 ssh_task에 대한 작업 로그에 다음과 비슷한 출력이 표시됩니다.

```
[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
```

Apache Airflow Snowflake 연결에 AWS Secrets Manager의 암호 키 사용

다음 샘플은 Amazon Managed Workflows for Apache Airflow에서 Apache Airflow Snowflake 연결의 암호 키를 얻기 위해 AWS Secrets Manager을(를) 호출합니다. 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager의 단계를 완료했다고 가정합니다.

주제

- 버전

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Apache Airflow 구성 옵션인 Secrets Manager 백엔드.
- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager의 Apache Airflow 연결 문자열.

권한

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager 권한.

요구 사항

이 페이지의 샘플 코드를 사용하려면 다음 종속성을 사용자 `requirements.txt`에 추가합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

```
apache-airflow-providers-snowflake==1.3.0
```

코드 샘플

다음 단계는 Secrets Manager를 호출하여 암호를 가져오는 DAG 코드를 만드는 방법을 설명합니다.

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 snowflake_connection.py로 저장합니다.

```
"""
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",
    )
```

다음 단계

- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 dags 폴더에 업로드하는 방법을 알아봅니다.

CloudWatch에서 DAG를 사용하여 사용자 지정 지표 작성

다음 코드 예제를 사용하여 PythonOperator를 실행하는 방향성 비순환 그래프(DAG)를 작성하여 Amazon MWAA 환경에 대한 OS 수준 지표를 검색할 수 있습니다. 그런 다음 DAG는 데이터를 Amazon CloudWatch에 사용자 지정 지표로 게시합니다.

사용자 지정 OS 수준 지표를 통해 환경 작업자가 가상 메모리 및 CPU와 같은 리소스를 어떻게 활용하고 있는지 추가로 파악할 수 있습니다. 이 정보를 사용하여 워크로드에 가장 적합한 환경 클래스를 선택할 수 있습니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지에서 코드 예제를 사용하려면 다음이 필요합니다.

- Amazon MWAA 환경.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

종속성

- 이 페이지의 코드 예제를 사용하는 데 추가 종속성이 필요하지 않습니다.

코드 예제

1. 명령 프롬프트에서 DAG 코드가 저장된 폴더로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 예제의 콘텐츠를 복사하고 로컬에서 dag-custom-metrics.py로 저장합니다. 환경 이름을 MWAA-ENV-NAME 로 변경합니다.

```
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. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. DAG가 성공적으로 실행되면 Apache Airflow 로그에 다음과 비슷한 콘텐츠가 표시됩니다.

```
[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
```

Amazon MWAA 환경에서 Aurora PostgreSQL 데이터베이스 정리

아파치 에어플로우용 Amazon 관리형 워크플로는 Aurora PostgreSQL 데이터베이스를 아파치 에어플로우 메타데이터 데이터베이스로 사용합니다. 이 데이터베이스에는 DAG 실행 및 작업 인스턴스가 저장됩니다. 다음 샘플 코드는 Amazon MWAA 환경의 전용 Aurora PostgreSQL 데이터베이스에서 항목을 정기적으로 정리합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

의존성

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 예제

다음 DAG는 에 지정된 테이블의 메타데이터 데이터베이스를 정리합니다. TABLES_TO_CLEAN 이 예제에서는 지정된 테이블에서 지난 7일 동안의 데이터를 삭제합니다. 항목이 삭제되는 기간을 MAX_AGE_IN_DAYS 조정하려면 다른 값으로 설정합니다.

Apache Airflow v2

```
from airflow import settings
from airflow.utils.dates import days_ago
from airflow.models import DagTag, DagModel, DagRun, ImportError, Log, SlaMiss,
 RenderedTaskInstanceFields, TaskInstance, TaskReschedule, XCom
from airflow.decorators import dag, task
from airflow.utils.dates import days_ago
from time import sleep

from airflow.version import version
major_version, minor_version = int(version.split('.')[0]), int(version.split('.')
[1])
if major_version >= 2 and minor_version >= 6:
    from airflow.jobs.job import Job
else:
    # The BaseJob class was renamed as of Apache Airflow v2.6
    from airflow.jobs.base_job import BaseJob as Job

# Delete entries for the past seven days. Adjust MAX_AGE_IN_DAYS to set how far back
 this DAG cleans the database.
MAX_AGE_IN_DAYS = 7
MIN_AGE_IN_DAYS = 0
DECREMENT = -7

# This is a list of (table, time) tuples.
# table = the table to clean in the metadata database
# time  = the column in the table associated to the timestamp of an entry
#         or None if not applicable.
TABLES_TO_CLEAN = [[Job, Job.latest_heartbeat],
    [TaskInstance, TaskInstance.execution_date],
    [TaskReschedule, TaskReschedule.execution_date],
    [DagTag, None],
    [DagModel, DagModel.last_parsed_time],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
```

```
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [XCom, XCom.execution_date],
]

@task()
def cleanup_db_fn(x):
    session = settings.Session()

    if x[1]:
        for oldest_days_ago in range(MAX_AGE_IN_DAYS, MIN_AGE_IN_DAYS, DECREMENT):
            earliest_days_ago = max(oldest_days_ago + DECREMENT, MIN_AGE_IN_DAYS)
            print(f"deleting {str(x[0])} entries between {earliest_days_ago} and
 {oldest_days_ago} days old...")
            earliest_date = days_ago(earliest_days_ago)
            oldest_date = days_ago(oldest_days_ago)
            query = session.query(x[0]).filter(x[1] >= oldest_date).filter(x[1] <=
 earliest_date)
            query.delete(synchronize_session= False)
            session.commit()
            sleep(5)
    else:
        # No time column specified for the table. Delete all entries
        print("deleting", str(x[0]), "...")
        query = session.query(x[0])
        query.delete(synchronize_session= False)
        session.commit()

    session.close()


@dag(
    dag_id="cleanup_db",
    schedule_interval="@weekly",
    start_date=days_ago(7),
    catchup=False,
    is_paused_upon_creation=False
)

def clean_db_dag_fn():
    t_last=None
    for x in TABLES_TO_CLEAN:
        t=cleanup_db_fn(x)
        if t_last:
```

```
            t_last >> t
        t_last = t

clean_db_dag = clean_db_dag_fn()
```

Amazon S3의 CSV 파일로 환경 메타데이터 내보내기

다음 코드 예제는 다양한 DAG 실행 정보에 대해 데이터베이스를 쿼리하고 Amazon S3에 저장된 `.csv` 파일에 데이터를 쓰는 DAG(방향성 비순환 그래프)를 생성하는 방법을 보여줍니다.

데이터를 로컬에서 검사하거나, 이를 객체 스토리지에 보관하거나, Amazon S3에서 Amazon Redshift 운영자 및 데이터베이스 정리와 같은 도구와 결합하기 위해, Amazon MWAA 메타데이터를 환경 외부로 이동하고 향후 분석을 위해 보존하기 위해 사용자 환경의 Aurora PostgreSQL 데이터베이스에서 정보를 내보내고자 할 수 있습니다.

Apache Airflow 모델에 나열된 모든 객체에 대해 데이터베이스를 쿼리할 수 있습니다. 이 코드 샘플은 DAG 실행과 관련된 정보를 제공하는 세 가지 모델인 `DagRun`, `TaskFail` 및 `TaskInstance`을 사용합니다.

주제

- 버전
- 필수 조건
- 권한
- 요구 사항
- 코드 예제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.
- 메타데이터 정보를 내보내고자 하는 새 Amazon S3 버킷.

권한

Amazon MWAA에는 쿼리된 메타데이터 정보를 Amazon S3에 쓰기 위한 Amazon S3 작업s3:PutObject에 대해 권한이 필요합니다. 다음 정책 문을 사용자 환경의 실행 역할에 추가합니다.

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::your-new-export-bucket"
}
```

이 정책은 *your-new-export-bucket*에 대해서만 쓰기 액세스를 제한합니다.

요구 사항

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 예제

다음 단계는 Aurora PostgreSQL을 쿼리하고 새 Amazon S3 버킷에 결과를 쓰는 DAG를 생성하는 방법을 설명합니다.

1. 터미널에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

   ```
   cd dags
   ```

2. 다음 코드 예제의 콘텐츠를 복사하고 로컬에서 metadata_to_csv.py로 저장합니다. 사용자 DAG가 메타데이터 데이터베이스에서 쿼리하는 가장 오래된 기록의 보존 기간을 제어하기 위해 MAX_AGE_IN_DAYS에 할당된 값을 변경할 수 있습니다.

   ```
   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. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 성공적으로 실행되면 export_db 작업의 작업 로그에 다음과 유사한 출력이 표시됩니다.

```
[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
```

이제 새 Amazon S3 버킷에서 내보낸 .csv 파일에 액세스하여 /files/export/에 다운로드할 수 있습니다.

Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용

다음 샘플은 Amazon Managed Workflows for Apache Airflow에서 Apache Airflow 변수에 대한 암호 키를 가져오기 위한 AWS Secrets Manager 호출입니다. 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager의 단계를 완료했다고 가정합니다.

주제

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.
- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Apache Airflow 구성 옵션인 Secrets Manager 백엔드.
- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager의 Apache Airflow 변수 문자열.

권한

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager 권한.

요구 사항

- 이 코드 예제를 Apache Airflow v1과 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v1 기본 설치를 사용합니다.

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 샘플

다음 단계는 Secrets Manager를 호출하여 암호를 가져오는 DAG 코드를 만드는 방법을 설명합니다.

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 `secrets-manager-var.py`로 저장합니다.

```
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
    )
```

다음 단계

- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 dags 폴더에 업로드하는 방법을 알아봅니다.

Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용

다음은 Amazon Managed Workflows for Apache Airflow에서 Apache Airflow 연결을 위한 암호 키를 가져오기 위한 샘플 호출 AWS Secrets Manager입니다. 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager의 단계를 완료했다고 가정합니다.

주제

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Apache Airflow 구성 옵션인 Secrets Manager 백엔드.
- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager의 Apache Airflow 연결 문자열.

권한

- 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 표시된 바와 같은 Secrets Manager 권한.

요구 사항

- 이 코드 예제를 Apache Airflow v1과 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v1 기본 설치를 사용합니다.

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 샘플

다음 단계는 Secrets Manager를 호출하여 암호를 가져오는 DAG 코드를 만드는 방법을 설명합니다.

Apache Airflow v2

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 콘텐츠를 복사하고 로컬에서 `secrets-manager.py`로 저장합니다.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
```

```
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Apache Airflow v1

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 콘텐츠를 복사하고 로컬에서 `secrets-manager.py`로 저장합니다.

```
from airflow import DAG, settings, secrets
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from airflow.contrib.hooks.aws_hook import AwsHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
```

```
    ### set up Secrets Manager
    hook = AwsHook()
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

다음 단계

- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 dags 폴더에 업로드하는 방법을 알아봅니다.

Oracle을 사용하여 사용자 지정 플러그인 생성

다음 샘플은 Amazon MWAA용 Oracle을 사용하여 사용자 지정 플러그인을 생성하는 단계를 안내하며, plugins.zip 파일에서 다른 사용자 지정 플러그인 및 바이너리와 결합할 수 있습니다.

목차

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.
- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.
- 사용자 환경의 모든 로그 수준, `CRITICAL` 또는 그 이상에서 작업자 로깅을 사용할 수 있습니다. Amazon MWAA 로그 유형 및 로그 그룹 관리 방법에 대한 자세한 내용은 the section called “Airflow 로그 확인” 단원을 참조하십시오.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

요구 사항

이 페이지의 샘플 코드를 사용하려면 다음 종속성을 사용자 `requirements.txt`에 추가합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

Apache Airflow v1

```
cx_Oracle==8.1.0
apache-airflow[oracle]==1.10.12
```

코드 샘플

다음 단계에서는 사용자 지정 플러그인을 테스트할 DAG 코드를 생성하는 방법을 설명합니다.

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 `oracle.py`로 저장합니다.

```
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
    )
```

사용자 지정 플러그인 만들기

이 섹션에서는 종속성을 다운로드하고, 사용자 지정 플러그인과 plugins.zip 파일을 만드는 방법을 설명합니다.

종속성 다운로드

Amazon MWAA는 plugins.zip 콘텐츠를 각 Amazon MWAA 스케줄러 및 작업자 컨테이너에 있는 `/usr/local/airflow/plugins`로 추출합니다. 이는 환경에 바이너리를 추가하는 데 사용됩니다. 다음 단계에서는 사용자 지정 플러그인에 필요한 파일을 조합하는 방법을 설명합니다.

Amazon Linux 컨테이너 이미지 가져오기

1. 명령 프롬프트에서 Amazon Linux 컨테이너 이미지를 가져와서 로컬에서 컨테이너를 실행합니다. 예:

   ```
   docker pull amazonlinux
   docker run -it amazonlinux:latest /bin/bash
   ```

   명령 프롬프트에서 bash 명령줄을 호출해야 합니다. 예:

   ```
   bash-4.2#
   ```

2. Linux 네이티브 비동기 I/O 기능(libaio)을 설치합니다.

   ```
   yum -y install libaio
   ```

3. 후속 단계를 위해 이 창을 열어 둡니다. `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1` 파일을 로컬로 복사할 예정입니다.

클라이언트 폴더 다운로드

1. unzip 패키지를 로컬에 설치합니다. 예:

   ```
   sudo yum install unzip
   ```

2. `oracle_plugin` 디렉터리를 생성합니다. 예:

```
mkdir oracle_plugin
cd oracle_plugin
```

3. 다음 curl 명령을 사용하여 Linux x86-64용 Oracle Instant 클라이언트 다운로드(64비트)에서 instantclient-basic-linux.x64-18.5.0.0.0dbru.zip을 다운로드합니다.

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. `client.zip` 파일의 압축을 풉니다. 예:

```
unzip *.zip
```

Docker에서 파일을 추출합니다.

1. 새 명령 프롬프트에서 Docker 컨테이너 ID를 표시하고 기록해 둡니다. 예:

```
docker container ls
```

명령 프롬프트는 모든 컨테이너와 해당 ID를 반환해야 합니다. 예:

```
debc16fd6970
```

2. `oracle_plugin` 디렉터리에서 `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1` 파일을 로컬 `instantclient_18_5` 폴더로 추출합니다. 예:

```
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/
```

사용자 지정 플러그인

Apache Airflow는 스타트업 시 플러그인 폴더에 있는 Python 파일의 콘텐츠를 실행합니다. 이는 환경 변수를 설정하고 수정하는 데 사용됩니다. 다음 단계에서는 사용자 지정 플러그인의 샘플 코드를 설명합니다.

- 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 env_var_plugin_oracle.py로 저장합니다.

```
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

다음 단계에서는 plugins.zip을 생성하는 방법을 보여줍니다. 이 예제의 내용은 다른 플러그인 및 바이너리와 결합하여 단일 plugins.zip 파일로 만들 수 있습니다.

플러그인 디렉터리의 콘텐츠를 압축합니다.

1. 명령 프롬프트에서 oracle_plugin 디렉터리로 이동합니다. 예:

```
cd oracle_plugin
```

2. instantclient_18_5 디렉터리를 plugins.zip으로 압축합니다. 예:

```
zip -r ../plugins.zip ./
```

3. 명령 프롬프트에 다음이 표시되어야 합니다.

```
oracle_plugin$ ls
client.zip  instantclient_18_5
```

4. client.zip 파일을 제거합니다. 예:

```
rm client.zip
```

env_var_plugin_oracle.py 파일을 압축합니다.

1. plugins.zip. 파일의 루트에 env_var_plugin_oracle.py 파일을 추가합니다. 예:

```
zip plugins.zip env_var_plugin_oracle.py
```

2. 이제 plugins.zip 파일에 다음 정보가 포함되어야 합니다.

```
env_var_plugin_oracle.py
instantclient_18_5/
```

Airflow 구성 옵션

Apache Airflow v2를 사용하는 경우 Apache Airflow 구성 옵션으로 `core.lazy_load_plugins : False`을 추가합니다. 자세한 내용은 2에서 구성 옵션을 사용하여 플러그인 로드를 참조하십시오.

다음 단계

- 이 예제의 `requirements.txt` 파일을 Python 종속성 설치의 Amazon S3 버킷에 업로드하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 `dags` 폴더에 업로드하는 방법을 알아봅니다.
- 이 예제의 `plugins.zip` 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

런타임 환경 변수를 생성하는 사용자 지정 플러그인 생성

다음 샘플은 Amazon Managed Workflows for Apache Airflow 환경에서 런타임 시 환경 변수를 생성하는 사용자 지정 플러그인을 생성하는 단계를 안내합니다.

주제

- 버전
- 필수 조건
- 권한
- 요구 사항
- 사용자 지정 플러그인
- Plugins.zip
- Airflow 구성 옵션

- 다음 단계

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

요구 사항

- 이 코드 예제를 Apache Airflow v1과 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v1 기본 설치를 사용합니다.

사용자 지정 플러그인

Apache Airflow는 스타트업 시 플러그인 폴더에 있는 Python 파일의 콘텐츠를 실행합니다. 이는 환경 변수를 설정하고 수정하는 데 사용됩니다. 다음 단계에서는 사용자 지정 플러그인의 샘플 코드를 설명합니다.

1. 명령 프롬프트에서 플러그인이 저장된 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. 다음 코드 샘플의 내용을 복사하여 위 폴더에 `env_var_plugin.py`로 로컬로 저장합니다.

```
from airflow.plugins_manager import AirflowPlugin
import os
```

```
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/lib/python3.7/
site-packages"
os.environ["JAVA_HOME"]="/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"

class EnvVarPlugin(AirflowPlugin):
     name = 'env_var_plugin'
```

Plugins.zip

다음 단계에서는 `plugins.zip`을 생성하는 방법을 보여줍니다. 이 예제의 내용은 다른 플러그인 및 바이너리와 결합하여 단일 `plugins.zip` 파일로 만들 수 있습니다.

1. 명령 프롬프트에서 이전 단계의 `hive_plugin` 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. `plugins` 폴더 내 콘텐츠를 압축합니다.

```
zip -r ../plugins.zip ./
```

Airflow 구성 옵션

Apache Airflow v2를 사용하는 경우 Apache Airflow 구성 옵션으로 `core.lazy_load_plugins : False`을 추가합니다. 자세한 내용은 2에서 구성 옵션을 사용하여 플러그인 로드를 참조하십시오.

다음 단계

- 이 예제의 `requirements.txt` 파일을 Python 종속성 설치의 Amazon S3 버킷에 업로드하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 `dags` 폴더에 업로드하는 방법을 알아봅니다.
- 이 예제의 `plugins.zip` 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

Amazon MWAA에서 DAG 시간대 변경

Apache Airflow는 기본적으로 방향성 비순환 그래프(DAG)를 UTC+0으로 스케줄링합니다. 다음 단계는 Amazon MWAA가 Pendulum을 사용하여 DAG를 실행하는 시간대를 변경하는 방법을 보여줍니다. 선택적으로 이 주제에서는 사용자 지정 플러그인을 생성하여 사용자 환경의 Apache Airflow 로그의 시간대를 변경하는 방법을 보여줍니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

Airflow 로그의 시간대를 변경하는 플러그인을 생성합니다.

Apache Airflow는 시작시 `plugins` 디렉터리의 Python 파일을 실행합니다. 다음 플러그인을 사용하면 실행자의 시간대를 재정의하여 Apache Airflow가 로그를 기록하는 시간대를 수정할 수 있습니다.

1. 사용자 지정 플러그인에 대해 지정된 이름 plugins 디렉터리를 생성하고 디렉터리로 이동합니다. 예:

```
$ mkdir plugins
$ cd plugins
```

2. 다음 코드 샘플의 콘텐츠를 복사하고 로컬의 plugins 폴더에 dag-timezone-plugin.py로 저장합니다.

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. plugins 디렉터리에서 __init__.py라는 빈 Python 파일을 만듭니다. plugins 디렉터리는 다음과 비슷해야 합니다.

```
plugins/
|-- __init__.py
|-- dag-timezone-plugin.py
```

plugins.zip 생성

다음 단계에서는 plugins.zip을 생성하는 방법을 보여줍니다. 이 예제의 콘텐츠를 다른 플러그인 및 바이너리와 결합하여 단일 plugins.zip 파일로 만들 수 있습니다.

1. 명령 프롬프트에서 이전 단계의 plugins 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. plugins 디렉터리 내의 콘텐츠를 압축합니다.

```
zip -r ../plugins.zip ./
```

3. S3 버킷에 plugins.zip를 업로드합니다.

```
$ aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

코드 샘플

DAG가 실행되는 기본 시간대(UTC+0)를 변경하려면 시간대를 인식하는 날짜/시간을 처리하는 Python 라이브러리인 Pendulum이라는 라이브러리를 사용합니다.

1. 명령 프롬프트에서 DAG가 저장된 디렉터리로 이동합니다. 예:

```
$ cd dags
```

2. 다음 예제의 콘텐츠를 복사하고 `tz-aware-dag.py`로 저장합니다.

```
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. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 성공하면 `tz_test` DAG의 `tz_aware_task`에 대한 작업 로그에 다음과 비슷한 결과가 출력됩니다.

```
[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
```

다음 단계

- 이 예제의 `plugins.zip` 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

CodeArtifact 토큰 새로고침

CodeArtifact를 사용하여 Python 종속성을 설치하는 경우 Amazon MWAA에는 활성 토큰이 필요합니다. Amazon MWAA가 런타임의 CodeArtifact 리포지토리에 액세스할 수 있도록 하려면 시작 스크립트를 사용하고 토큰과 `PIP_EXTRA_INDEX_URL`을(를) 설정하면 됩니다.

다음 주제에서는 `get_authorization_token` CodeArtifact API 작업을 사용하여 환경이 시작되거나 업데이트될 때마다 새 토큰을 검색하는 시작 스크립트를 만드는 방법을 설명합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.
- 환경에 대한 종속성을 저장하는 CodeArtifact 리포지토리입니다.

권한

CodeArtifact 토큰을 새로 고치고 결과를 Amazon S3에 쓰려면 Amazon MWAA의 실행 역할에 다음과 같은 권한이 있어야 합니다.

- 이 `codeartifact:GetAuthorizationToken` 작업을 통해 Amazon MWAA는 CodeArtifact에서 새 토큰을 검색할 수 있습니다. 다음 정책은 사용자가 생성하는 모든 CodeArtifact 도메인에 권한을 부여합니다. 명령문의 리소스 값을 수정하고 사용자 환경에서 액세스할 도메인만 지정하여 도메인에 대한 액세스를 추가로 제한할 수 있습니다.

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- CodeArtifact `GetAuthorizationToken` API 오퍼레이션을 호출하려면 `sts:GetServiceBearerToken` 액션이 필요합니다. 이 작업은 `pip`와(과) CodeArtifact 같은 패키지 관리자를 사용할 때 사용해야 하는 토큰을 반환합니다. CodeArtifact 리포지토리와 함께 패키지 관리자를 사용하려면 사용자 환경의 실행 역할 역할이 다음 정책 설명에 표시된 대로 `sts:GetServiceBearerToken`을(를) 허용해야 합니다.

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
```

```
}
```

코드 예제

다음 단계는 CodeArtifact 토큰을 업데이트하는 시작 스크립트를 만드는 방법을 설명합니다.

1. 다음 코드 샘플의 내용을 복사하고 로컬의 동일한 디렉터리에 `code_artifact_startup_script.sh`로 저장합니다.

```
#!/bin/sh

# Startup script for MWAA, see https://docs.aws.amazon.com/mwaa/latest/userguide/
using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"
echo "CodeArtifact startup setup complete"
```

2. 스크립트를 저장한 폴더로 이동합니다. 새 프롬프트 창에서 cp를 사용하여 스크립트를 버킷에 업로드합니다. *your-s3-bucket*을 자신의 정보로 바꿉니다.

```
$ aws s3 cp code_artifact_startup_script.sh s3://your-s3-bucket/
code_artifact_startup_script.sh
```

   성공하면 Amazon S3가 객체에 대한 URL 경로를 출력합니다.

```
upload: ./code_artifact_startup_script.sh to s3://your-s3-bucket/
code_artifact_startup_script.sh
```

스크립트를 업로드하면 시작 시 환경이 업데이트되고 스크립트가 실행됩니다.

다음 단계

- 시작 스크립트를 사용하여 the section called “시작 스크립트 사용”에서 환경을 사용자 지정하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 `dags` 폴더에 업로드하는 방법을 알아봅니다.
- 이 예제의 `plugins.zip` 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인 생성

Amazon MWAA는 `plugins.zip`의 콘텐츠를 `/usr/local/airflow/plugins`로 추출합니다. 이를 사용하여 컨테이너에 바이너리를 추가할 수 있습니다. 또한 Apache Airflow는 스타트업 시 `plugins` 폴더에 있는 Python 파일의 콘텐츠를 실행하므로 환경 변수를 설정하고 수정할 수 있습니다. 다음 샘플은 Amazon Managed Workflows for Apache Airflow 환경에서 Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인을 생성하고 다른 사용자 지정 플러그인 및 바이너리와 결합할 수 있는 단계를 안내합니다.

주제

- 버전
- 필수 조건
- 권한
- 요구 사항
- 종속성 다운로드
- 사용자 지정 플러그인
- Plugins.zip
- 코드 샘플
- Airflow 구성 옵션

- 다음 단계

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.
- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

요구 사항

이 페이지의 샘플 코드를 사용하려면 다음 종속성을 사용자 `requirements.txt`에 추가합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Apache Airflow v1

```
apache-airflow[hive]==1.10.12
```

종속성 다운로드

Amazon MWAA는 plugins.zip 콘텐츠를 각 Amazon MWAA 스케줄러 및 작업자 컨테이너에 있는 /usr/local/airflow/plugins로 추출합니다. 이는 환경에 바이너리를 추가하는 데 사용됩니다. 다음 단계에서는 사용자 지정 플러그인에 필요한 파일을 조합하는 방법을 설명합니다.

1. 명령 프롬프트에서 플러그인을 만들려는 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. 미러에서 Hadoop을 다운로드합니다. 예를 들면 다음과 같습니다.

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

3. 미러에서 Hive를 다운로드합니다. 예를 들면 다음과 같습니다.

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

4. 디렉터리를 생성합니다. 예:

```
mkdir hive_plugin
```

5. Hadoop을 추출합니다.

```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. Hive를 추출.

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

사용자 지정 플러그인

Apache Airflow는 스타트업 시 플러그인 폴더에 있는 Python 파일의 콘텐츠를 실행합니다. 이는 환경 변수를 설정하고 수정하는 데 사용됩니다. 다음 단계에서는 사용자 지정 플러그인의 샘플 코드를 설명합니다.

1. 명령 프롬프트에서 hive_plugin 디렉터리로 이동합니다. 예:

```
cd hive_plugin
```

2. 다음 코드 샘플의 콘텐츠를 복사하여 로컬의 hive_plugin 디렉터리에 hive_plugin.py로 저장합니다.

```
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. 다음 텍스트의 콘텐츠를 대응하여 로컬의 hive_plugin 디렉터리에 .airflowignore로 저장합니다.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

다음 단계에서는 plugins.zip을 생성하는 방법을 보여줍니다. 이 예제의 내용은 다른 플러그인 및 바이너리와 결합하여 단일 plugins.zip 파일로 만들 수 있습니다.

1. 명령 프롬프트에서 이전 단계의 hive_plugin 디렉터리로 이동합니다. 예:

```
cd hive_plugin
```

2. plugins 폴더 내 콘텐츠를 압축합니다.

```
zip -r ../hive_plugin.zip ./
```

코드 샘플

다음 단계에서는 사용자 지정 플러그인을 테스트할 DAG 코드를 생성하는 방법을 설명합니다.

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 콘텐츠를 복사하고 로컬에서 hive.py로 저장합니다.

```
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'
    )
```

Airflow 구성 옵션

Apache Airflow v2를 사용하는 경우 Apache Airflow 구성 옵션으로 core.lazy_load_plugins : False을 추가합니다. 자세한 내용은 2에서 구성 옵션을 사용하여 플러그인 로드를 참조하십시오.

다음 단계

- 이 예제의 requirements.txt 파일을 Python 종속성 설치의 Amazon S3 버킷에 업로드하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 dags 폴더에 업로드하는 방법을 알아봅니다.
- 이 예제의 plugins.zip 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성

다음 샘플은 Amazon Managed Workflows for Apache Airflow에서 사용자 지정 플러그인을 사용하여 Apache Airflow PythonVirtualenvOperator를 패치하는 방법을 보여줍니다.

주제

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.
- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

권한

- 이 페이지의 코드 예제를 사용하는 데 추가 권한이 필요하지 않습니다.

요구 사항

이 페이지의 샘플 코드를 사용하려면 다음 종속성을 사용자 `requirements.txt`에 추가합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

```
virtualenv
```

사용자 지정 플러그인 샘플 코드

Apache Airflow는 스타트업 시 플러그인 폴더에 있는 Python 파일의 콘텐츠를 실행합니다. 이 플러그인은 시작 프로세스 중에 내장 `PythonVirtualenvOperator`을 패치하여 Amazon MWAA와 호환되도록 합니다. 다음 단계는 사용자 지정 플러그인의 샘플 코드를 보여줍니다.

Apache Airflow v2

1. 명령 프롬프트에서 위의 `plugins` 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 `virtual_python_plugin.py`로 저장합니다.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str,
 system_site_packages: bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd
```

```
airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v1

1. 명령 프롬프트에서 위의 `plugins` 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 `virtual_python_plugin.py`로 저장합니다.

```
from airflow.plugins_manager import AirflowPlugin
from airflow.operators.python_operator import PythonVirtualenvOperator

def _generate_virtualenv_cmd(self, tmp_dir):
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if self.system_site_packages:
        cmd.append('--system-site-packages')
    if self.python_version is not None:
        cmd.append('--python=python{}'.format(self.python_version))
    return cmd
PythonVirtualenvOperator._generate_virtualenv_cmd=_generate_virtualenv_cmd

class EnvVarPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Plugins.zip

다음 단계에서는 `plugins.zip`을 생성하는 방법을 보여줍니다.

1. 명령 프롬프트에서 위의 `virtual_python_plugin.py`이 포함된 디렉터리로 이동합니다. 예:

```
cd plugins
```

2. `plugins` 폴더 내 콘텐츠를 압축합니다.

```
zip plugins.zip virtual_python_plugin.py
```

코드 샘플

다음 단계에서는 사용자 지정 플러그인의 DAG 코드를 생성하는 방법을 설명합니다.

Apache Airflow v2

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 `virtualenv_test.py`로 저장합니다.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from airflow.operators.python import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)
```

```
with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Apache Airflow v1

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 `virtualenv_test.py`로 저장합니다.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from airflow.operators.python_operator import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"
```

```
def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Airflow 구성 옵션

Apache Airflow v2를 사용하는 경우 Apache Airflow 구성 옵션으로 `core.lazy_load_plugins : False`을 추가합니다. 자세한 내용은 2에서 구성 옵션을 사용하여 플러그인 로드를 참조하십시오.

다음 단계

- 이 예제의 `requirements.txt` 파일을 Python 종속성 설치의 Amazon S3 버킷에 업로드하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 `dags` 폴더에 업로드하는 방법을 알아봅니다.
- 이 예제의 `plugins.zip` 파일을 사용자 지정 플러그인 설치의 Amazon S3 버킷에 업로드하는 방법에 대해 자세히 알아봅니다.

Lambda 함수를 사용한 DAG 호출

다음 코드 예제는 Amazon MWAA 환경에서 AWS Lambda 함수를 사용하여 Apache Airflow CLI 토큰을 가져오고 방향성 비순환 그래프(DAG)를 호출합니다.

주제

- 버전
- 필수 조건
- 권한

- 의존성
- 코드 예제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 코드 예제를 활용하려면 다음과 같이 해야 합니다.

- Amazon MWAA 환경에서는 퍼블릭 네트워크 액세스 모드를 사용합니다.
- 최신 Python 런타임을 사용하는 Lambda 함수를 보유합니다.

> **Note**
>
> Lambda 함수와 Amazon MWAA 환경이 동일한 VPC에 있는 경우, 프라이빗 네트워크에서 이 코드를 사용할 수 있습니다. 이 구성의 경우 Lambda 함수의 실행 역할에 Amazon Elastic Compute Cloud(Amazon EC2) CreateNetworkInterface API 작업을 호출할 수 있는 권한이 필요합니다. `AWSLambdaVPCAccessExecutionRole` AWS 관리형 정책을 사용하여 이 권한을 제공할 수 있습니다.

권한

이 페이지의 코드 예제를 사용하려면 Amazon MWAA 환경의 실행 역할에 `airflow:CreateCliToken` 작업을 수행할 수 있는 액세스 권한이 필요합니다. `AmazonMWAAAirflowCliAccess` AWS 관리형 정책을 사용하여 이 권한을 제공할 수 있습니다.

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

```
            "Resource": "*"
        }
    ]
}
```

자세한 정보는 아파치 에어플로우 CLI 정책: 아마존MWAA 액세스 AirflowCli을 참조하세요.

의존성

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 예제

1. https://console.aws.amazon.com/lambda/ 에서 AWS Lambda 콘솔을 엽니다.
2. 함수 목록에서 Lambda 함수를 선택합니다.
3. 함수 페이지에서 다음 코드를 복사하고 다음을 리소스 이름으로 바꿉니다.
   - YOUR_ENVIRONMENT_NAME – Amazon MWAA 환경의 이름입니다.
   - YOUR_DAG_NAME – 호출하려는 DAG의 이름입니다.

```
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. 배포를 선택합니다.
5. Lambda 콘솔을 사용하여 함수를 호출하려면 테스트를 선택합니다.
6. Lambda가 DAG를 성공적으로 호출했는지 확인하려면 Amazon MWAA를 사용하여 환경의 Apache Airflow UI로 이동한 후 다음을 수행합니다.
   a. DAG 페이지의 DAG 목록에서 새 대상 DAG를 찾습니다.
   b. 마지막 실행에서 최신 DAG 실행의 타임스탬프를 확인합니다. 이 타임스탬프는 사용자의 다른 환경에서 `invoke_dag`에 대한 최신 타임스탬프와 거의 일치해야 합니다.
   c. 최근 작업에서 마지막 실행이 성공했는지 확인합니다.

여러 Amazon MWAA 환경에서 DAG 호출

다음 코드 예제는 Apache Airflow CLI 토큰을 생성합니다. 그런 다음 코드는 한 Amazon MWAA 환경에서 방향성 비순환 그래프(DAG)를 사용하여 다른 Amazon MWAA 환경에서 DAG를 호출합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지에서 코드 예제를 사용하려면 다음이 필요합니다.

- 퍼블릭 네트워크 웹 서버에 액세스할 수 있는 두 개의 Amazon MWAA 환경(현재 환경 포함).
- 대상 환경의 Amazon Simple Storage Service(S3) 버킷에 업로드된 샘플 DAG.

권한

이 페이지의 코드 예제를 사용하려면 사용자 환경의 실행 역할에 Apache Airflow CLI 토큰을 생성할 권한이 있어야 합니다. AWS관리형 정책을 연결하여 이 권한을 `AmazonMWAAAirflowCliAccess` 부여할 수 있습니다.

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

자세한 내용은 아파치 에어플로우 CLI 정책: 아마존MWAA 액세스 AirflowCli 단원을 참조하십시오.

종속성

- 이 코드 예제를 Apache Airflow v2와 함께 사용하려면 추가 종속성이 필요하지 않습니다. 코드는 사용자 환경에 설치된 Apache Airflow v2 기본 설치를 사용합니다.

코드 예제

다음 코드 예제는 현재 환경에서 DAG를 사용하여 다른 환경에서 DAG를 호출한다고 가정합니다.

1. 터미널에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 예제의 내용을 복사하고 로컬에서 invoke_dag.py로 저장합니다. 다음 값을 사용자의 정보로 교체합니다.

   - your-new-environment-name— DAG를 호출하려는 다른 환경의 이름.
   - your-target-dag-id— 호출하려는 다른 환경에서 DAG의 ID.

```
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. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. DAG가 성공적으로 실행되면 invoke_dag_task의 작업 로그에 다음과 유사한 출력이 표시됩니다.

```
[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
```

DAG가 성공적으로 호출되었는지 확인하려면 새 환경에 대한 Apache Airflow UI로 이동한 다음 다음을 수행합니다.

a. DAG 페이지의 DAG 목록에서 새 대상 DAG를 찾습니다.

b. 마지막 실행에서 최신 DAG 실행의 타임스탬프를 확인합니다. 이 타임스탬프는 사용자의 다른 환경에서 invoke_dag에 대한 최신 타임스탬프와 거의 일치해야 합니다.

c. 최근 작업에서 마지막 실행이 성공했는지 확인합니다.

Amazon RDS for Microsoft SQL Server와 함께 Amazon MWAA 사용

Amazon Managed Workflows for Apache Airflow를 사용하여 RDS for SQL Server에 연결할 수 있습니다. 다음 샘플 코드는 Amazon Managed Workflows for Apache Airflow 환경에서 DAG를 사용하여 Amazon RDS for Microsoft SQL Server 서버에 연결하고 쿼리를 실행합니다.

주제

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.
- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.
- Amazon MWAA와 RDS for SQL Server 는 동일한 Amazon VPC에서 실행/
- Amazon MWAA 및 서버의 VPC 보안 그룹은 다음과 같은 연결로 구성됩니다.
  - Amazon MWAA의 보안 그룹에 있는 Amazon RDS에 대해 포트 1433 오픈에 대한 인바운드 규칙
  - 또는 Amazon MWAA에서 RDS로의 1433 오픈 포트에 대한 아웃바운드 규칙
- RDS for SQL Server에 대한 Apache Airflow 연결에는 이전 프로세스에서 만든 Amazon RDS SQL 서버 데이터베이스의 호스트 이름, 포트, 사용자 이름 및 암호가 반영됩니다.

종속성

이 섹션의 샘플 코드를 사용하려면 다음 종속성을 사용자 `requirements.txt`에 추가합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

Apache Airflow v2

```
apache-airflow-providers-microsoft-mssql==1.0.1
apache-airflow-providers-odbc==1.0.1
pymssql==2.2.1
```

Apache Airflow v1

```
apache-airflow[mssql]==1.10.12
```

Apache Airflow v2 연결

Apache Airflow v2에서 연결을 사용하는 경우 Airflow 연결 객체에 다음과 같은 키-값 쌍이 포함되어 있는지 확인합니다.

1. 연결 ID: mssql_default
2. 연결 유형: Amazon Web Services
3. 호스트: YOUR_DB_HOST
4. 스키마:
5. 로그인: 관리자
6. 암호:
7. 포트: 1433
8. 추가:

코드 샘플

1. 명령 프롬프트에서 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 콘텐츠를 복사하고 로컬에서 sql-server.py로 저장합니다.

```
"""
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
```

다음 단계

- 이 예제의 `requirements.txt` 파일을 Python 종속성 설치의 Amazon S3 버킷에 업로드하는 방법을 알아봅니다.
- 이 예제의 DAG 코드를 DAG 추가 또는 업데이트에서 Amazon S3 버킷의 `dags` 폴더에 업로드하는 방법을 알아봅니다.
- 예제 스크립트와 기타 pymssql 모듈 예제를 살펴봅니다.

- Apache Airflow 참조 가이드에서 mssql_operator를 사용하여 특정 Microsoft SQL 데이터베이스에서 SQL 코드를 실행하는 방법에 대해 자세히 알아봅니다.

Amazon EMR과 함께 Amazon MWAA 사용

다음 코드 샘플은 Amazon EMR 및 Amazon Managed Workflows for Apache Airflow를 사용하여 통합을 활성화하는 방법을 보여줍니다.

주제

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.

코드 샘플

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

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

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
    FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
    COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
    IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
    """
    from airflow import DAG

    from airflow.contrib.operators.emr_add_steps_operator import EmrAddStepsOperator
    from airflow.contrib.operators.emr_create_job_flow_operator import
EmrCreateJobFlowOperator
```

```
from airflow.contrib.sensors.emr_step_sensor import EmrStepSensor

from airflow.utils.dates import days_ago
from datetime import timedelta
import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}

SPARK_STEPS = [
    {
        'Name': 'calculate_pi',
        'ActionOnFailure': 'CONTINUE',
        'HadoopJarStep': {
            'Jar': 'command-runner.jar',
            'Args': ['/usr/lib/spark/bin/run-example', 'SparkPi', '10'],
        },
    }
]

JOB_FLOW_OVERRIDES = {
    'Name': 'my-demo-cluster',
    'ReleaseLabel': 'emr-5.30.1',
    'Applications': [
        {
            'Name': 'Spark'
        },
    ],
    'Instances': {
        'InstanceGroups': [
            {
                'Name': "Master nodes",
                'Market': 'ON_DEMAND',
                'InstanceRole': 'MASTER',
                'InstanceType': 'm5.xlarge',
                'InstanceCount': 1,
            },
```

```
                {
                    'Name': "Slave nodes",
                    'Market': 'ON_DEMAND',
                    'InstanceRole': 'CORE',
                    'InstanceType': 'm5.xlarge',
                    'InstanceCount': 2,
                }
            ],
            'KeepJobFlowAliveWhenNoSteps': False,
            'TerminationProtected': False,
            'Ec2KeyName': 'mykeypair',
        },
        'VisibleToAllUsers': True,
        'JobFlowRole': 'EMR_EC2_DefaultRole',
        'ServiceRole': 'EMR_DefaultRole'
    }

    with DAG(
        dag_id=DAG_ID,
        default_args=DEFAULT_ARGS,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval='@once',
        tags=['emr'],
    ) as dag:

        cluster_creator = EmrCreateJobFlowOperator(
            task_id='create_job_flow',
            job_flow_overrides=JOB_FLOW_OVERRIDES
        )

        step_adder = EmrAddStepsOperator(
            task_id='add_steps',
            job_flow_id="{{ task_instance.xcom_pull(task_ids='create_job_flow',
key='return_value') }}",
            aws_conn_id='aws_default',
            steps=SPARK_STEPS,
        )

        step_checker = EmrStepSensor(
            task_id='watch_step',
            job_flow_id="{{ task_instance.xcom_pull('create_job_flow',
key='return_value') }}",
```

```
            step_id="{{ task_instance.xcom_pull(task_ids='add_steps',
key='return_value')[0] }}",
            aws_conn_id='aws_default',
        )

        cluster_creator >> step_adder >> step_checker
```

Amazon EKS에서 Amazon MWAA 사용

다음 샘플은 Amazon EKS와 함께 Amazon Managed Workflows for Apache Airflow를 사용하는 방법을 보여줍니다.

주제

- 버전
- 필수 조건
- Amazon EC2용 퍼블릭 키 생성
- 클러스터 생성
- mwaa 네임스페이스 생성
- mwaa 네임스페이스에 대한 역할 생성
- Amazon EKS 클러스터에 대한 IAM 역할 생성 및 연결
- requirements.txt 파일 생성
- Amazon EKS에 대한 자격 증명 매핑 생성
- kubeconfig 생성
- DAG 생성
- Amazon S3 버킷에 DAG 및 kube_config.yaml 추가
- 예제 활성화 및 트리거

버전

- 이 페이지의 샘플 코드는 Python 3.7의 Apache Airflow v1과 함께 사용할 수 있습니다.

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 항목의 예제를 사용하려면 다음이 필요합니다.

- Amazon MWAA 환경.
- eksctl. 자세한 내용은 eksctl 설치를 참조하십시오.
- kubectl. 자세한 내용은 kubectl 설치 및 설정을 참조하십시오. 경우에 따라 eksctl과 함께 설치되기도 합니다.
- Amazon MWAA 환경을 생성하는 리전의 EC2 키 쌍입니다. 자세히 알아보려면 키 쌍 생성 또는 가져오기를 참조하십시오.

> **Note**
>
> `eksctl` 명령을 사용할 때 `--profile`를 포함하여 기본값 이외의 프로필을 지정할 수 있습니다.

Amazon EC2용 퍼블릭 키 생성

다음 명령을 사용하여 프라이빗 키 쌍에서 퍼블릭 키를 생성합니다.

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

자세한 내용은 키 쌍에 대한 퍼블릭 키 검색을 참조하십시오.

클러스터 생성

다음 명령을 사용하여 클러스터를 생성합니다. 클러스터에 사용자 지정 이름을 지정하거나 다른 리전에 생성하려면 이름 및 리전 값을 바꿉니다. Amazon MWAA 환경을 생성하는 동일한 리전에서 클러스터를 생성해야 합니다. Amazon MWAA에 사용하는 Amazon VPC 네트워크의 서브넷과 일치하도록 서브넷 값을 바꿉니다. `ssh-public-key`의 값을 사용하는 키와 일치하도록 바꿉니다. 동일한 리전에 있는 Amazon EC2의 기존 키를 사용하거나 Amazon MWAA 환경을 생성한 동일한 리전에서 새 키를 생성할 수 있습니다.

```
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-1111111111111111, subnet-2222222222222222" \
--vpc-private-subnets "subnet-33333333333333333, subnet-44444444444444444"
```

클러스터 생성을 완료하는 데 시간이 걸립니다. 완료되면 다음 명령을 사용하여 클러스터가 성공적으로 생성되었고 IAM OIDC 공급자가 구성되어 있는지 확인할 수 있습니다.

```
eksctl utils associate-iam-oidc-provider \
--region us-west-2 \
--cluster mwaa-eks \
--approve
```

mwaa 네임스페이스 생성

클러스터가 성공적으로 생성되었는지 확인한 후 다음 명령을 사용하여 포드에 대한 네임스페이스를 생성합니다.

```
kubectl create namespace mwaa
```

mwaa 네임스페이스에 대한 역할 생성

네임스페이스를 생성한 후, MWAA 네임스페이스에서 포드를 실행할 수 있는 EKS의 Amazon MWAA 사용자에 대한 역할 및 역할 바인딩을 생성합니다. 네임스페이스에 다른 이름을 사용한 경우 `-n` *mwaa*의 mwaa를 사용한 이름으로 바꿉니다.

```
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
```

다음 명령을 실행하여 새 역할이 Amazon EKS 클러스터에 액세스할 수 있는지 확인합니다. *mwaa*를 사용하지 않았다면 반드시 올바른 이름을 사용하십시오.

```
kubectl get pods -n mwaa --as mwaa-service
```

또한 다음과 같은 메시지가 표시됩니다.

```
No resources found in mwaa namespace.
```

Amazon EKS 클러스터에 대한 IAM 역할 생성 및 연결

IAM 역할을 생성한 다음 Amazon EKS(k8s) 클러스터에 바인딩해야 IAM을 통한 인증에 사용할 수 있습니다. 역할은 클러스터에 로그인하는 데만 사용되며 콘솔 또는 API 호출에 대한 권한은 없습니다.

Amazon MWAA 실행 역할의 단계를 사용하여 Amazon MWAA 환경을 위한 새 역할을 생성합니다. 하지만 해당 항목에 설명된 정책을 생성하고 첨부하는 대신 다음 정책을 첨부하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:${MWAA_REGION}:${ACCOUNT_NUMBER}:environment/
${MWAA_ENV_NAME}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
```

```
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:${MWAA_REGION}:${ACCOUNT_NUMBER}:log-group:airflow-
${MWAA_ENV_NAME}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:${MWAA_REGION}:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:${ACCOUNT_NUMBER}:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "sqs.${MWAA_REGION}.amazonaws.com"
                    ]
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:${MWAA_REGION}:${ACCOUNT_NUMBER}:cluster/
${EKS_CLUSTER_NAME}"
        }
    ]
}
```

역할을 생성한 후 Amazon MWAA 환경을 편집하여 생성한 역할을 환경의 실행 역할로 사용합니다. 역할을 변경하려면 사용할 환경을 편집합니다. 권한에서 실행 역할을 선택합니다.

알려진 문제:

- Amazon EKS로 인증할 수 없는 하위 경로가 있는 역할 ARN에 알려진 문제가 있습니다. 이에 대한 해결 방법은 Amazon MWAA에서 자체적으로 생성한 서비스 역할을 사용하는 대신 서비스 역할을 수동으로 생성하는 것입니다. 자세히 알아보려면 aws-auth configmap의 ARN에 경로가 포함된 경우 경로가 있는 역할이 작동하지 않음을 참조하십시오.
- IAM에서 Amazon MWAA 서비스 목록을 사용할 수 없는 경우 Amazon EC2와 같은 대체 서비스 정책을 선택한 다음, 역할의 신뢰 정책을 다음과 일치하도록 업데이트해야 합니다.

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

자세한 내용은 IAM 역할에 신뢰 정책을 사용하는 방법을 참조하십시오.

requirements.txt 파일 생성

이 섹션의 샘플 코드를 사용하려면 다음 데이터베이스 옵션 중 하나를 `requirements.txt`에 추가했는지 확인합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.

Apache Airflow v2

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Apache Airflow v1

```
awscli
kubernetes==12.0.1
```

Amazon EKS에 대한 자격 증명 매핑 생성

다음 명령에서 생성한 역할의 ARN을 사용하여 Amazon EKS에 대한 자격 증명 매핑을 생성합니다. 리전 *your-region*을 사용자가 환경을 생성한 리전으로 바꿉니다. 역할의 ARN을 바꾸고, 마지막으로 *mwaa-execution-role*을 사용자 환경의 실행 역할로 바꿉니다.

```
eksctl create iamidentitymapping \
--region your-region \
--cluster mwaa-eks \
--arn arn:aws:iam::111222333444:role/mwaa-execution-role \
--username mwaa-service
```

kubeconfig 생성

다음 명령을 실행해 kubeconfig을 생성합니다.

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

update-kubeconfig 실행 시 특정 프로필을 사용한 경우 kube_config.yaml 파일에 추가된 env: 섹션을 제거해야 Amazon MWAA에서 제대로 작동할 수 있습니다. 이렇게 하려면 파일에서 다음을 삭제한 다음 저장합니다.

```
env:
- name: AWS_PROFILE
  value: profile_name
```

DAG 생성

다음 코드 예제를 사용하여 DAG에 대한 mwaa_pod_example.py와 같은 Python 파일을 만듭니다.

Apache Airflow v2

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
 KubernetesPodOperator

default_args = {
   'owner': 'aws',
   'depends_on_past': False,
   'start_date': datetime(2019, 2, 20),
   'provide_context': True
}

dag = DAG(
   'kubernetes_pod_example', default_args=default_args, schedule_interval=None)
```

```
#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                        namespace="mwaa",
                        image="ubuntu:18.04",
                        cmds=["bash"],
                        arguments=["-c", "ls"],
                        labels={"foo": "bar"},
                        name="mwaa-pod-test",
                        task_id="pod-task",
                        get_logs=True,
                        dag=dag,
                        is_delete_operator_pod=False,
                        config_file=kube_config_path,
                        in_cluster=False,
                        cluster_context='aws'
                        )
```

Apache Airflow v1

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from datetime import datetime
from airflow.contrib.operators.kubernetes_pod_operator import KubernetesPodOperator

default_args = {
   'owner': 'aws',
   'depends_on_past': False,
```

```
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                        namespace="mwaa",
                        image="ubuntu:18.04",
                        cmds=["bash"],
                        arguments=["-c", "ls"],
                        labels={"foo": "bar"},
                        name="mwaa-pod-test",
                        task_id="pod-task",
                        get_logs=True,
                        dag=dag,
                        is_delete_operator_pod=False,
                        config_file=kube_config_path,
                        in_cluster=False,
                        cluster_context='aws'
                        )
```

Amazon S3 버킷에 DAG 및 **kube_config.yaml** 추가

생성한 DAG와 kube_config.yaml 파일을 Amazon MWAA 환경용 Amazon S3 버킷에 넣습니다. Amazon S3 콘솔 또는 AWS Command Line Interface를 사용하여 파일을 버킷에 넣을 수 있습니다.

예제 활성화 및 트리거

Apache Airflow에서 예제를 활성화한 다음 트리거합니다.

성공적으로 실행되고 완료되면 다음 명령을 사용하여 포드를 확인합니다.

```
kubectl get pods -n mwaa
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
NAME READY STATUS RESTARTS AGE
```

```
mwaa-pod-test-aa11bb22cc3344445555666677778888 0/1 Completed 0 2m23s
```

이후 다음 명령을 사용하여 포드의 출력을 확인할 수 있습니다. 이름 값을 이전 명령에서 반환된 값으로 바꿉니다.

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc3344445555666677778888
```

ECSOperator를 사용하여 Amazon ECS에 연결

이 항목에서는 ECSOperator를 사용하여 Amazon MWAA에서 Amazon Elastic Container Service(Amazon ECS) 컨테이너에 연결하기 위한 샘플 코드를 설명합니다. 다음 단계에서는 환경의 실행 역할에 필요한 권한을 추가하고, AWS CloudFormation 템플릿을 사용하여 Amazon ECS Fargate 클러스터를 생성하고, 마지막으로 새 클러스터에 연결되는 DAG를 생성 및 업로드합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

이 페이지의 이 샘플 코드를 사용하려면 다음 항목이 필요합니다.

- Amazon MWAA 환경.

권한

- 환경의 실행 역할에는 Amazon ECS에서 작업을 실행할 수 있는 권한이 필요합니다. AWSAmazonECS_FullAccess 관리형 정책을 실행 역할에 연결하거나 다음 정책을 생성하여 실행 역할에 연결할 수 있습니다.

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

- Amazon ECS에서 작업을 실행하는 데 필요한 권한을 추가하는 것 외에도 다음과 같이 Amazon MWAA 실행 역할에서 CloudWatch Logs 정책 설명을 수정하여 Amazon ECS 작업 로그 그룹에 대한 액세스를 허용해야 합니다. Amazon ECS 로그 그룹은 the section called “Amazon ECS 클러스터 생성”의 AWS CloudFormation 템플릿에 의해 생성됩니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
```

```
            "logs:CreateLogGroup",
            "logs:PutLogEvents",
            "logs:GetLogEvents",
            "logs:GetLogRecord",
            "logs:GetLogGroupFields",
            "logs:GetQueryResults"
        ],
        "Resource": [
            "arn:aws:logs:region:account-id:log-group:airflow-environment-name-*",
            "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
        ]
    }
```

Amazon MWAA 실행 역할 및 정책 연결 방법에 대한 자세한 내용은 실행 역할 단원을 참조하십시오.

Amazon ECS 클러스터 생성

다음 AWS CloudFormation 템플릿을 사용하여 Amazon MWAA 워크플로우에 사용할 Amazon ECS Fargate 클러스터를 구축합니다. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 작업 정의 생성을 참조하십시오.

1. 다음 코드로 JSON 파일을 생성하고 `ecs-mwaa-cfn.json`으로 저장합니다.

```
{
    "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. 명령 프롬프트에서 다음 AWS CLI 명령을 사용하여 새 스택을 생성합니다. SecurityGroups 값과 SubnetIds 값을 Amazon MWAA 환경의 보안 그룹 및 서브넷의 값으로 바꿔야 합니다.

```
$ 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
```

또는 다음 쉘 스크립트를 사용할 수 있습니다. 스크립트는 get-environment AWS CLI 명령을 사용하여 환경의 보안 그룹 및 서브넷에 필요한 값을 검색한 다음 그에 따라 스택을 생성합니다. 스크립트를 실행하려면 다음을 수행합니다.

a. 스크립트를 복사하여 AWS CloudFormation 템플릿과 동일한 디렉터리에 `ecs-stack-helper.sh`로 저장합니다.

```
#!/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. 다음 명령을 사용하여 스크립트를 실행합니다. `environment-name` 및 `stack-name`을(를) 자신의 정보로 대체합니다.

```
$ chmod +x ecs-stack-helper.sh
$ ./ecs-stack-helper.bash environment-name stack-name
```

성공하면 새 AWS CloudFormation 스택 ID를 표시하는 다음 출력이 표시됩니다.

```
{
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

AWS CloudFormation 스택이 완성되고 AWS가 Amazon ECS 리소스를 프로비저닝한 후에는 DAG를 생성하고 업로드할 준비가 된 것입니다.

코드 샘플

1. 명령 프롬프트를 열고 DAG 코드가 저장된 디렉터리로 이동합니다. 예:

```
cd dags
```

2. 다음 코드 샘플의 내용을 복사하여 로컬에 `mwaa-ecs-operator.py`로 저장한 다음 새 DAG를 Amazon S3에 업로드합니다.

```
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**
>
> 예제 DAG에서는 awslogs_group의 경우 Amazon ECS 작업 로그 그룹의 이름을 사용하여 로그 그룹을 수정해야 할 수 있습니다. 예제에서는 mwaa-ecs-zero이라는 이름의 로그 그룹을 가정합니다. awslogs_stream_prefix의 경우 Amazon ECS 작업 로그 스트림 접두사를 사용합니다. 이 예제에서는 로그 스트림 접두사, ecs를 가정합니다.

3. 다음 AWS CLI 명령을 실행하여 DAG를 환경 버킷에 복사한 다음 Apache Airflow UI를 사용하여 DAG를 트리거합니다.

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 성공하면 ecs_fargate_dag DAG의 ecs_operator_task에 대한 작업 로그에 다음과 비슷한 출력이 표시됩니다.

```
[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
```

Amazon MWAA에서 dbt 사용

이 주제에서는 Amazon MWAA으로 dbt 및 Postgres를 사용하는 방법을 보여줍니다. 다음 단계에서는 필수 종속성을 `requirements.txt`에 추가하고 샘플 dbt 프로젝트를 환경의 Amazon S3 버킷에 업로드합니다. 그런 다음 샘플 DAG를 사용하여 Amazon MWAA가 종속성을 설치했는지 확인하고 마지막으로 `BashOperator`를 사용하여 dbt 프로젝트를 실행합니다.

주제

버전

- 이 페이지의 코드 예제는 Python 3.10의 Apache Airflow v2 이상에서 사용할 수 있습니다.

필수 조건

다음 단계를 완료하려면 먼저 다음을 수행해야 합니다.

- Apache Airflow v2.2.2를 사용하는 Amazon MWAA 환경. 이 샘플은 v2.2.2로 작성 및 테스트되었습니다. 다른 Apache Airflow 버전과 함께 사용하려면 샘플을 수정해야 할 수 있습니다.
- 샘플 dbt 프로젝트. Amazon MWAA에서 dbt 사용을 시작하려면 포크를 생성하고 dbt-labs 리포지토리에서 dbt 스타터 프로젝트를 복제하면 됩니다. GitHub

의존성

dbt와 함께 Amazon MWAA를 사용하려면 다음 시작 스크립트를 환경에 추가하십시오. 자세한 내용은 Amazon MWAA에서 시작 스크립트 사용을 참조하십시오.

```
#!/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
```

다음 섹션에서는 dbt 프로젝트 디렉터리를 Amazon S3에 업로드하고 Amazon MWAA가 필수 dbt 종속성을 성공적으로 설치했는지 여부를 검증하는 DAG를 실행합니다.

Amazon S3에 dbt 프로젝트를 업로드합니다.

Amazon MWAA 환경에서 dbt 프로젝트를 사용할 수 있으려면 전체 프로젝트 디렉터리를 환경 dags 폴더에 업로드하면 됩니다. 환경이 업데이트되면 Amazon MWAA는 dbt 디렉터리를 로컬 `usr/local/airflow/dags/` 폴더에 다운로드합니다.

Amazon S3에 dbt 프로젝트를 업로드하려면

1. dbt 스타터 프로젝트를 복제한 디렉터리로 이동합니다.
2. 다음 Amazon S3 AWS CLI 명령을 실행하여 `--recursive` 파라미터를 사용하여 프로젝트의 콘텐츠를 환경 dags 폴더에 재귀적으로 복사합니다. 이 명령은 모든 dbt 프로젝트에 사용할 수 있는 dbt라고 하는 하위 디렉터리를 생성합니다. 하위 디렉터리가 이미 있는 경우 프로젝트 파일은 기존 디렉터리에 복사되며 새 디렉터리는 생성되지 않습니다. 또한 이 명령은 이 특정 스타터 프로젝트의 dbt 디렉터리 내에 하위 디렉터리를 생성합니다.

   ```
   $ aws s3 cp dbt-starter-project s3://mwaa-bucket/dags/dbt/dbt-starter-project --
   recursive
   ```

   프로젝트 하위 디렉터리에 다른 이름을 사용하여 상위 dbt 디렉터리 내에 여러 dbt 프로젝트를 구성할 수 있습니다.

DAG를 사용하여 dbt 종속성 설치를 확인합니다.

다음 DAG는 BashOperator 및 bash 명령을 사용하여 Amazon MWAA가 requirements.txt에 지정된 dbt 종속성을 성공적으로 설치했는지 확인합니다.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="/usr/local/airflow/.local/bin/dbt --version"
    )
```

작업 로그를 보고 dbt 및 해당 종속성이 설치되었는지 확인하려면 다음을 수행합니다.

1. Amazon MWAA 콘솔로 이동한 다음 사용 가능한 환경 목록에서 Open Airflow UI를 선택합니다.
2. Apache Airflow UI의 목록에서 dbt-installation-test DAG를 찾은 다음 Last Run 열 아래에서 날짜를 선택하여 마지막으로 성공한 작업을 엽니다.
3. 그래프 보기를 사용하여 bash_command 작업을 선택하여 작업 인스턴스 세부 정보를 엽니다.
4. 로그를 선택하여 작업 로그를 연 다음, 로그에 requirements.txt 지정된 dbt 버전이 성공적으로 나열되는지 확인합니다.

DAG를 사용하여 dbt 프로젝트를 실행합니다.

다음 DAG는 BashOperator를 사용하여 Amazon S3에 업로드한 dbt 프로젝트를 로컬 usr/local/airflow/dags/ 디렉터리에서 쓰기 액세스 가능한 /tmp 디렉터리로 복사한 다음 dbt 프로젝트를 실행합니다. bash 명령은 제목이 dbt-starter-project인 스타터 dbt 프로젝트를 가정합니다. 프로젝트 디렉터리 이름에 따라 디렉터리 이름을 수정합니다.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

import os

DAG_ID = os.path.basename(__file__).replace(".py", "")
```

```
# assumes all files are in a subfolder of DAGs called dbt

with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False, start_date=days_ago(1))
 as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="source /usr/local/airflow/python3-virtualenv/dbt-env/bin/
activate;\
cp -R /usr/local/airflow/dags/dbt /tmp;\
echo 'listing project files:';\
ls -R /tmp;\
cd /tmp/dbt/mwaa_dbt_test_project;\
/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt run --project-dir /tmp/dbt/
mwaa_dbt_test_project --profiles-dir ..;\
cat /tmp/dbt_logs/dbt.log;\
rm -rf /tmp/dbt/mwaa_dbt_test_project"
    )
```

AWS 블로그 및 튜토리얼

- Apache Airflow v2.x용 Amazon EKS 및 Amazon MWAA 작업

Amazon Managed Workflows for Apache Airflow 모범 사례

이 가이드에서는 Amazon Managed Workflows for Apache Airflow를 사용할 때 권장하는 모범 사례를 설명합니다.

주제

Amazon MWAA의 Apache Airflow 성능 튜닝

이 페이지에서는 Amazon MWAA에서 Apache Airflow 구성 옵션 사용를 사용하여 Amazon Managed Workflows for Apache Airflow 환경의 성능을 조정할 때 권장하는 모범 사례를 설명합니다.

목차

Apache Airflow 구성 옵션 추가

다음 절차에서는 Airflow 구성 옵션을 환경에 추가하는 단계를 안내합니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.

4. 다음을 선택합니다.
5. Airflow 구성 옵션 창에서 사용자 지정 구성 추가를 선택합니다.
6. 드롭다운 목록에서 구성을 선택하고 값을 입력하거나, 사용자 정의 구성을 입력하고 값을 입력합니다.
7. 추가하려는 각 구성에 대해 사용자 정의 구성 추가를 선택합니다.
8. 저장을 선택합니다.

자세한 내용은 Amazon MWAA에서 Apache Airflow 구성 옵션 사용 단원을 참조하십시오.

Apache Airflow 스케줄러

Apache Airflow 스케줄러는 Apache Airflow의 핵심 구성 요소입니다. 스케줄러에 문제가 있으면 DAG가 파싱되지 않고 작업이 예약되지 않을 수 있습니다. Apache Airflow 스케줄러 조정에 대한 자세한 내용은 Apache Airflow 문서 웹사이트의 스케줄러 성능 미세 조정을 참조하십시오.

파라미터

이 섹션에서는 Apache Airflow 스케줄러에 사용할 수 있는 구성 옵션과 해당 사용 사례에 대해 설명합니다.

Apache Airflow v2

버전	구성 옵션	기본값	설명	사용 사례
v2	celery.sync_parallelism	1	Celery 실행기가 작업 상태를 동기화하는 데 사용하는 프로세스 수입니다.	이 옵션을 사용하면 셀러리 실행기가 사용하는 프로세스를 제한하여 대기열 충돌을 방지할 수 있습니다. 기본적으로 작업 로그를 1 Logs로 전송할 때 오류가 발생하지 않도록 값이 설정됩니다.

버전	구성 옵션	기본값	설명	사용 사례
				CloudWatch 값을 0으로 설정하는 것은 최대 프로세스 수를 사용하는 것을 의미하지만, 작업 로그 전달 시 오류가 발생할 수 있습니다.

버전	구성 옵션	기본값	설명	사용 사례
v2	scheduler.processor_poll_interval	1	스케줄러 '루프'에서 연속적인 DAG 파일 프로세싱 사이에 대기해야 하는 초 단위 시간입니다.	이 옵션을 사용하면 실행기에서 DAG 파싱 결과 검색, 작업 찾기, 대기열에 저장, 저장된 작업 실행을 완료한 후, 스케줄러가 대기 상태로 유지되는 시간을 늘려 스케줄러의 CPU 사용량을 줄일 수 있습니다. 이 값을 늘리면 Apache Airflow v2용 `scheduler.parsing_processes` 와 Apache Airflow v1용 `scheduler.max_threads` 환경에서 실행되는 스케줄러 스레드 수가 소모됩니다. 이렇게 하면 스케줄러의 DAG 파싱 용량이 줄어들어 DAG가 웹 서버에 표시되는 데 걸리는 시간

버전	구성 옵션	기본값	설명	사용 사례
				이 늘어날 수 있습니다.
v2	scheduler.max_dagruns_to_create_per_loop	10	스케줄러 “루프”당 생성할 DagRunsDAG의 최대 수입니다.	이 옵션을 사용하면 스케줄러 “루프”의 최대 수를 줄임으로써 작업 스케줄링에 필요한 리소스를 확보할 수 있습니다. DagRuns
v2	scheduler.parsing_processes	2	스케줄러가 DAG를 스케줄링하기 위해 병렬로 실행할 수 있는 스레드 수입니다.	이 옵션을 사용하면 스케줄러가 DAG를 파싱하기 위해 병렬로 실행하는 프로세스 수를 줄여 리소스를 확보할 수 있습니다. DAG 파싱이 작업 스케줄링에 영향을 미치는 경우, 이 수치를 낮게 유지하는 것이 좋습니다. 사용자 환경의 vCPU 수보다 적은 값을 지정해야 합니다. 자세한 내용은 제한을 참조하십시오.

Limits

이 섹션에서는 스케줄러의 기본 파라미터를 조정할 때 고려해야 하는 제한에 대해 설명합니다.

scheduler.parsing_processes, scheduler.max_threads

환경 클래스의 경우 vCPU당 스레드 2개가 허용됩니다. 환경 클래스의 스케줄러용으로 하나 이상의 스레드를 예약해야 합니다. 작업을 예약하는 데 지연이 발생하는 경우, 환경 클래스를 늘려야 할 수 있습니다. 예를 들어, 대규모 환경에는 스케줄러용으로 4개의 vCPU Fargate 컨테이너 인스턴스가 있습니다. 즉, 최대 7개의 스레드를 다른 프로세스에 사용할 수 있습니다. 즉, 2개의 스레드 곱하기 4개의 vCPU에서 스케줄러 자체용으로 하나를 뺀 것입니다. `scheduler.max_threads` 및 `scheduler.parsing_processes`에서 지정하는 값은 환경 클래스에 사용할 수 있는 스레드 수를 초과해서는 안 됩니다(아래 그림 참조).

- mw1.small - 다른 프로세스의 경우 1 스레드를 초과해서는 안 됩니다. 나머지 스레드는 스케줄러용으로 예약되어 있습니다.
- mw1.medium - 다른 프로세스의 경우 3 스레드를 초과해서는 안 됩니다. 나머지 스레드는 스케줄러용으로 예약되어 있습니다.
- mw1.large - 다른 프로세스의 경우 7 스레드를 초과해서는 안 됩니다. 나머지 스레드는 스케줄러용으로 예약되어 있습니다.

DAG 폴더

Apache Airflow 스케줄러는 사용자 환경의 DAG 폴더를 지속적으로 스캔합니다. 포함된 모든 `plugins.zip` 파일, 또는 'airflow' 가져오기 명령문이 포함된 Python(`.py`) 파일 그런 다음 모든 Python DAG 객체를 해당 파일에 배치하여 스케줄러가 해당 파일을 처리하여 스케줄링해야 하는 작업(있는 경우) 을 결정합니다. DagBag Dag 파일 구문 분석은 파일에 실행 가능한 DAG 객체가 포함되어 있는지 여부에 관계없이 수행됩니다.

파라미터

이 섹션에서는 DAG 폴더에 사용할 수 있는 구성 옵션과 해당 사용 사례에 대해 설명합니다.

Apache Airflow v2

버전	구성 옵션	기본값	설명	사용 사례
v2	scheduler.dag_dir_list_interval	300초	DAG 폴더에서 새 파일을 검색해야 하는 초 단위 시간입니다.	이 옵션을 사용하면 DAG 폴더를 파싱하는 데 걸리는 초 단위 시간을 늘려 리소스를 확보할 수 있습니다. DAG 폴더에 파일이 너무 많아 `total_parse_time` metrics의 파싱 시간이 오래 걸리면 이 값을 늘리는 것이 좋습니다.
v2	scheduler.min_file_process_interval	30초	스케줄러가 DAG를 파싱하고 DAG에 대한 업데이트가 반영되기까지 걸리는 초 단위 시간입니다.	이 옵션을 사용하면 스케줄러가 DAG를 파싱하기 전에 대기하는 초 단위 시간을 늘려 리소스를 확보할 수 있습니다. 예를 들면, `30`의 값을 지정하면 30초마다 DAG 파일 파싱이 이뤄집니다. 사용자 환경의 CPU 사용량을 줄이려면 이 수치를 높게 유

버전	구성 옵션	기본값	설명	사용 사례
				지하는 것이 좋습니다.

DAG 파일

Apache Airflow 스케줄러 루프의 일부로 개별 DAG 파일을 구문 분석하여 DAG Python 객체를 추출합니다. Apache Airflow v2 이상에서는 스케줄러가 동시에 최대 수의 파싱 프로세스를 파싱합니다. 동일한 파일을 다시 파싱하려면 `scheduler.min_file_process_interval`에 지정된 시간(초)이 경과해야 합니다.

파라미터

이 섹션에서는 Apache Airflow DAG 파일에 사용할 수 있는 구성 옵션과 해당 사용 사례에 대해 설명합니다.

Apache Airflow v2

버전	구성 옵션	기본값	설명	사용 사례
v2	core.dag_file_processor_timeout	50초	DagFile프로세서의 DAG 파일 처리 시간이 초과되기까지의 시간 (초).	이 옵션을 사용하면 DagFile프로세서 제한 시간이 초과될 때까지 걸리는 시간을 늘려 리소스를 확보할 수 있습니다. DAG 프로세싱 로그에 시간 초과가 발생하여 실행 가능한 DAG가 로드되지 않는 경우, 이 값을 늘리는 것이 좋습니다.

버전	구성 옵션	기본값	설명	사용 사례
v2	core.dagbag_import_timeout	30초	Python 파일을 가져오기하는 데 걸리는 시간이 초과되기 까지의 초 단위 시간입니다.	이 옵션을 사용하면 Python 파일을 가져와서 DAG 객체를 추출하는 동안 스케줄러의 시간이 초과될 때까지 걸리는 시간을 늘려 리소스를 확보할 수 있습니다. 이 옵션은 스케줄러 '루프'의 일부로 처리되며 `core.dag_file_processor_timeout` 에 지정된 값보다 작은 값을 포함해야 합니다.

버전	구성 옵션	기본값	설명	사용 사례
v2	core.min_serialized_dag_update_interval	30	데이터베이스의 직렬화된 DAG가 업데이트되기까지 걸리는 초 단위 최소 시간입니다.	이 옵션을 사용하면 데이터베이스의 직렬화된 DAG가 업데이트기까지 걸리는 초 단위 시간을 늘려 리소스를 확보할 수 있습니다. DAG가 많거나 DAG가 복잡한 경우, 이 값을 늘리는 것이 좋습니다. 이 값을 늘리면 DAG가 직렬화될 때 스케줄러와 데이터베이스의 부하가 줄어듭니다.

버전	구성 옵션	기본값	설명	사용 사례
v2	core.min_serialized_dag_fetch_interval	10	에 이미 로드되어 있을 때 직렬화된 DAG를 데이터베이스에서 다시 페치하는 시간 (초) 입니다. DagBag	이 옵션을 사용하면 직렬화된 DAG를 다시 페치하는 초 단위 시간을 늘려 리소스를 확보할 수 있습니다. 데이터베이스 '쓰기' 속도를 줄이려면 이 값이 `core.min_serialized_dag_update_interval` 에 지정된 값보다 커야 합니다. 이 값을 늘리면 DAG가 직렬화될 때 웹 서버와 데이터베이스의 부하가 줄어듭니다.

Tasks

Apache Airflow 스케줄러와 작업자는 모두 대기열 생성 및 대기열 제거 작업에 관여합니다. 스케줄러는 파싱된 작업을 없음 상태에서 예약됨 상태로 전환하여 스케줄링할 수 있습니다. 실행기는 Fargate의 스케줄러 컨테이너에서도 실행되며, 해당 작업을 대기열에 넣고 상태를 대기 상태로 설정합니다. 작업자가 작업 용량을 확보하면, 대기열에서 작업을 가져와 상태를 실행 중으로 설정합니다. 그러면 작업의 성공 또는 실패 여부에 따라 상태가 성공 또는 실패로 변경됩니다.

파라미터

이 섹션에서는 Apache Airflow 작업에 사용할 수 있는 구성 옵션과 해당 사용 사례에 대해 설명합니다.

Amazon MWAA에서 재정의하는 기본 구성 옵션은 ###으로 표시됩니다.

Apache Airflow v2

버전	구성 옵션	기본값	설명	사용 사례
v2	core.parallelism	10000	'실행 중' 상태를 가질 수 있는 작업 인스턴스의 최대 수입니다.	이 옵션을 사용하면 동시에 실행할 수 있는 작업 인스턴스 수를 늘려 리소스를 확보할 수 있습니다. 지정된 값은 가용한 작업자 수에 작업자의 작업 밀도를 '곱한' 값이어야 합니다. 이 값은 '실행 중' 또는 '대기 중' 상태에서 멈춘 작업이 많은 경우에만 변경하는 것이 좋습니다.
v2	core.dag_concurrency	10000	각 DAG에서 동시에 실행할 수 있는 작업 인스턴스의 수입니다.	이 옵션을 사용하면 동시에 실행할 수 있는 작업 인스턴스 수를 늘려 리소스를 확보할 수 있습니다. 예를 들어, 10개의 병렬 작업이 있는 100개의 DAG가 있고, 모든 DAG를 동시에 실행하

버전	구성 옵션	기본값	설명	사용 사례
				고자 하는 경우, 가용 작업자 수에 `celery.worker_concurrency` 에 있는 작업자의 작업 밀도를 '곱한' 값을 DAG 수(예: 100)로 나누어 최대 병렬도를 계산할 수 있습니다.
v2	core.execute_tasks_new_python_interpreter	`True`	Apache Airflow가 상위 프로세스를 분기하여 작업을 실행하는지 아니면 새 Python 프로세스를 생성하여 작업을 실행하는지 결정합니다.	`True`로 설정하면, Apache Airflow는 플러그인에 대한 변경 사항을 작업을 실행하기 위해 생성된 새로운 Python 프로세스로 인식합니다.
v2	celery.worker_concurrency	N/A	Amazon MWAA는 이 옵션에 대한 Airflow 기본 설치를 재정의하여 자동 확장 구성 요소의 일부로서 작업자를 크기를 조정합니다.	`# ### ### ## ## #####.`

버전	구성 옵션	기본값	설명	사용 사례
v2	celery.worker_autoscale	mw1.small - 5,0 mw1.medium - 10,0 mw1.Large - 20,0 mw1.xlarge - 40,0 mw1.2xlarge - 80,0	작업자 타스트의 동시성	이 옵션을 사용하면 작업자의`minimum`, `maximum` 작업 동시성을 줄여 리소스를 확보할 수 있습니다. 작업자는 충분한 리소스가 있는지 여부에 관계없이 구성된 최대 `maximum`개의 동시 작업을 수락합니다. 리소스가 충분하지 않은 상태에서 작업을 예약하면 작업이 즉시 실패합니다. 리소스를 많이 사용하는 작업의 경우, 이 값을 기본값보다 작게 줄여 작업당 용량을 늘릴 수 있도록 변경하는 것이 좋습니다.

requirements.txt에서의 Python 종속성 관리

이 페이지에서는 Amazon Managed Workflows for Apache Airflow 환경을 위해 `requirements.txt` 파일에 Python 종속성을 설치 및 관리하기 위해 권장하는 모범 사례를 설명합니다.

목차

Amazon MWAA CLI 유틸리티를 사용한 DAG 테스트

- 명령줄 인터페이스(CLI) 유틸리티는 Amazon Managed Workflows for Apache Airflow 환경을 로컬로 복제합니다.
- CLI는 Amazon MWAA 프로덕션 이미지와 유사한 Docker 컨테이너 이미지를 로컬로 구축합니다. 이를 통해 Amazon MWAA에 배포하기 전에 로컬 Apache Airflow 환경을 실행하여 DAG, 사용자 지정 플러그인 및 종속성을 개발하고 테스트할 수 있습니다.
- CLI를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub

PyPi.org 요구 사항 파일 형식을 사용하여 Python 종속성 설치

다음 PyPi 섹션은.org 요구 사항 파일 형식에 따라 Python 종속성을 설치하는 다양한 방법을 설명합니다.

옵션 1: Python 패키지 인덱스의 Python 종속성

다음 섹션에서는 `requirements.txt` 파일의 Python 패키지 인덱스에서 Python 종속성을 지정하는 방법을 설명합니다.

Apache Airflow v2

1. 로컬에서 테스트. `requirements.txt` 파일을 생성하기 전에 라이브러리를 반복적으로 추가하여 패키지와 버전의 적절한 조합을 찾습니다. Amazon MWAA CLI 유틸리티를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub
2. Apache Airflow 패키지 엑스트라를 검토합니다. Amazon MWAA에서 아파치 에어플로우 v2용으로 설치된 패키지 목록을 보려면 웹 사이트의 Amazon MWAA 로컬 러너를 참조하십시오. `requirements.txt` GitHub
3. 제약 조건 문을 추가합니다. `requirements.txt` 파일 상단에 Apache Airflow v2 환경에 대한 제약 조건 파일을 추가합니다. Apache Airflow 제약 조건 파일은 Apache Airflow 릴리스 당시 사용할 수 있는 공급자 버전을 지정합니다.

   Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.

   다음 예제에서 *{environment-version} # ### ### ##* 번호로 바꾸고, *{Python-version}* 을 사용자 환경과 호환되는 Python 버전으로 바꾸십시오.

   Apache Airflow 환경과 호환되는 Python 버전에 대한 자세한 내용은 Apache Airflow 버전을 참조하십시오.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   제약 조건 파일에서 `xyz==1.0` 패키지가 사용자 환경의 다른 패키지와 호환되지 않는 것으로 확인되면 `pip3 install`은 호환되지 않는 라이브러리가 환경에 설치되는 것을 막을 수 없습니다. 패키지 설치가 실패하는 경우 로그의 해당 로그 스트림에서 각 Apache Airflow 구성 요소(스케줄러, 작업자, 웹 서버) 에 대한 오류 로그를 볼 수 있습니다. CloudWatch 로그 형식에 대한 자세한 내용은 the section called “Airflow 로그 확인” 단원을 참조하십시오.
4. Apache Airflow 패키지 패키지 엑스트라 및 버전(==)을 추가합니다. 이렇게 하면 이름은 같지만 버전이 다른 패키지가 사용자 환경에 설치되는 것을 방지할 수 있습니다.

   ```
   apache-airflow[package-extra]==2.5.1
   ```
5. Python 라이브러리. `requirements.txt` 파일에 패키지 이름과 버전(==)을 추가합니다. 이렇게 하면 PyPi.org의 향후 주요 업데이트가 자동으로 적용되는 것을 방지할 수 있습니다.

```
library == version
```

Example Boto3 및 psycopg2-binary

이 예제는 데모용으로 제공됩니다. boto 및 psycopg2-binary 라이브러리는 Apache Airflow v2 기본 설치에 포함되어 있으며 `requirements.txt` 파일에서 지정할 필요가 없습니다.

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

버전 없이 패키지를 지정한 경우 Amazon MWAA는.org에서 패키지의 최신 버전을 설치합니다. PyPi 이 버전은 `requirements.txt`에 있는 다른 패키지와 충돌할 수 있습니다.

Apache Airflow v1

1. 로컬에서 테스트. `requirements.txt` 파일을 생성하기 전에 라이브러리를 반복적으로 추가하여 패키지와 버전의 적절한 조합을 찾습니다. Amazon MWAA CLI 유틸리티를 실행하려면 on에서 aws-mwaa-local-runner를 참조하십시오. GitHub
2. Airflow 패키지 추가 내용을 검토합니다. https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt에서 Apache Airflow v1.10.12에 사용할 수 있는 패키지 목록을 검토합니다.
3. 제약조건 파일 추가 `requirements.txt` 파일 상단에 Apache Airflow v1.10.12의 제약 조건 파일을 추가합니다. 제약 조건 파일에서 `xyz==1.0` 패키지가 사용자 환경의 다른 패키지와 호환되지 않는 것으로 확인되면 `pip3 install`은 호환되지 않는 라이브러리가 환경에 설치되는 것을 막을 수 없습니다.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-1.10.12/constraints-3.7.txt"
   ```
4. Apache Airflow v1.10.12 패키지. Airflow 패키지 엑스트라 및 Apache Airflow v1.10.12 버전(==)을 추가합니다. 이렇게 하면 이름은 같지만 버전이 다른 패키지가 사용자 환경에 설치되는 것을 방지할 수 있습니다.

   ```
   apache-airflow[package]==1.10.12
   ```

Example Secure Shell(SSH)

다음 예제 `requirements.txt` 파일은 Apache Airflow v1.10.12용 SSH를 설치합니다.

```
apache-airflow[ssh]==1.10.12
```

5. Python 라이브러리. `requirements.txt` 파일에 패키지 이름과 버전(==)을 추가합니다. 이렇게 하면 PyPi.org의 향후 주요 업데이트가 자동으로 적용되는 것을 방지할 수 있습니다.

```
library == version
```

Example Boto3

다음 예제 `requirements.txt` 파일은 Apache Airflow v1.10.12용 Boto3 라이브러리를 설치합니다.

```
boto3 == 1.17.4
```

버전 없이 패키지를 지정한 경우 Amazon MWAA는.org에서 패키지의 최신 버전을 설치합니다. PyPi 이 버전은 `requirements.txt`에 있는 다른 패키지와 충돌할 수 있습니다.

옵션 2: Python 휠(.whl)

Python 휠은 컴파일된 아티팩트와 함께 라이브러리를 배포하도록 설계된 패키지 형식입니다. Amazon MWAA에 종속성을 설치하는 방법으로 휠 패키지를 사용하면 여러 가지 이점이 있습니다.

- 빠른 설치 - WHL 파일을 단일 ZIP으로 컨테이너에 복사하여 로컬에 설치하므로 각 파일을 다운로드할 필요가 없습니다.
- 충돌 감소 - 패키지의 버전 호환성을 미리 확인할 수 있습니다. 따라서, `pip`가 호환되는 버전을 재귀적으로 찾아낼 필요가 없습니다.
- 복원성 향상 - 외부에서 호스팅되는 라이브러리를 사용하면 다운스트림 요구 사항이 변경되어 Amazon MWAA 환경의 컨테이너 간 버전이 호환되지 않는 문제가 발생할 수 있습니다. 외부 소스의 종속성에 의존하지 않기 때문에 각 컨테이너가 예시된 시기에 관계없이 모든 컨테이너가 동일한 라이브러리를 갖게 됩니다.

Python 휠 아카이브(`.whl`)의 Python 종속성을 설치하려면 사용자의 `requirements.txt`에서 다음 메서드를 사용하는 것이 좋습니다.

메서드

- Amazon S3 버킷에서 plugins.zip 파일 사용
- URL에 호스팅된 WHL 파일 사용
- DAG에서의 WHL 파일 생성

Amazon S3 버킷에서 **plugins.zip** 파일 사용

Apache Airflow 스케줄러, 작업자 및 웹 서버 (Apache Airflow v2.2.2 이상용) 는 시작 중에 사용자 환경에 맞는 -managed AWS Fargate 컨테이너에서 사용자 지정 플러그인을 찾습니다. `/usr/local/airflow/plugins/*` 이 프로세스는 Python 종속성과 Apache Airflow 서비스 시작을 위한 Amazon MWAA의 `pip3 install -r requirements.txt` 이전에 시작됩니다. 환경 실행 중에 지속적으로 변경되지 않기를 바라는 파일, 또는 DAG를 작성하는 사용자에게 액세스 권한을 부여하고 싶지 않은 모든 파일에는 `plugins.zip` 파일을 사용합니다. 예를 들면, Python 라이브러리 휠 파일, 인증서 PEM 파일, 구성 YAML 파일이 있습니다.

다음 섹션에서는 Amazon S3 버킷의 `plugins.zip` 파일에 있는 휠을 설치하는 방법을 설명합니다.

1. 필요한 WHL 파일 다운로드 Amazon MWAA 로컬 러너 또는 다른 Amazon Linux 2 컨테이너에 있는 기존 `requirements.txt`와 함께 `pip download`를 이용하여 필요한 Python 휠 파일을 확인하고 다운로드할 수 있습니다.

   ```
   $ pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
   $ cd "$AIRFLOW_HOME/plugins"
   $ zip "$AIRFLOW_HOME/plugins.zip" *
   ```

2. **`requirements.txt`**에서의 경로 지정 다음 그림과 같이 `--find-links`를 사용하여 requirements.txt 상단에 플러그인 디렉터리를 지정하고, `--no-index`를 사용하여 `pip`이 다른 소스에서 설치하지 않도록 지시합니다.

   ```
   --find-links /usr/local/airflow/plugins
   --no-index
   ```

   Example requirements.txt의 휠

   다음 예제는 Amazon S3 버킷의 루트에 있는 `plugins.zip` 파일에 휠을 업로드했다고 가정합니다. 예:

   ```
   --find-links /usr/local/airflow/plugins
   ```

```
--no-index

numpy
```

Amazon MWAA가 `plugins` 폴더에서 `numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl` 휠을 가져와 사용자 환경에 설치합니다.

URL에 호스팅된 WHL 파일 사용

다음 섹션에서는 URL에서 호스팅되는 휠 설치 방법을 설명합니다. URL은 공개적으로 액세스할 수 있거나 Amazon MWAA 환경을 위해 명시한 사용자 지정 Amazon VPC 내에서 액세스할 수 있어야 합니다.

- URL 제공 `requirements.txt` 내 휠에 URL을 입력합니다.

  Example 퍼블릭 URL의 휠 아카이브

  다음 예에서는 퍼블릭 사이트에서 휠을 다운로드합니다.

  ```
  --find-links https://files.pythonhosted.org/packages/
  --no-index
  ```

  Amazon MWAA는 지정한 URL에서 휠을 가져와 사용자 환경에 설치합니다.

  > **Note**
  >
  > Amazon MWAA v2.2.2 이상의 요구 사항을 설치하는 프라이빗 웹 서버에서는 URL에 액세스할 수 없습니다.

DAG에서의 WHL 파일 생성

Apache Airflow v2.2.2 이상을 사용하는 사설 웹 서버가 있고 환경에서 외부 리포지토리에 액세스할 수 없어 요구 사항을 설치할 수 없는 경우, 다음 DAG를 사용하여 기존 Amazon MWAA 요구 사항을 가져와 Amazon S3에 패키징할 수 있습니다.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago
```

```
S3_BUCKET = 'my-s3-bucket'
S3_KEY = 'backup/plugins_whl.zip'

with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/
requirements/requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3
 cp /tmp/plugins.zip s3://{S3_BUCKET}/{S3_KEY}"
    )
```

DAG를 실행한 후, 옵션으로 이 새 파일을 Amazon MWAA `plugins.zip`으로 사용하고, 옵션으로 다른 플러그인과 함께 패키징할 수 있습니다. 그런 다음 이전 버전을 추가하지 않고 업데이트하십시오. `requirements.txt --find-links /usr/local/airflow/plugins --no-index --constraint`

이 방법을 사용하면 동일한 라이브러리를 오프라인에서 사용할 수 있습니다.

옵션 3: 프라이빗 PyPi /PEP-503 호환 리포지토리에서 호스팅되는 Python 종속성

다음 섹션에서는 인증을 통해 프라이빗 URL에 호스팅되는 Apache Airflow 엑스트라를 설치하는 방법을 설명합니다.

1. 사용자 이름과 암호를 Apache Airflow 구성 옵션으로 추가합니다. 예:
   - `foo.user` : *YOUR_USER_NAME*
   - `foo.pass` : *YOUR_PASSWORD*
2. `requirements.txt` 파일 생성 다음 예제의 자리 표시자를 프라이빗 URL과 Apache Airflow 구성 옵션으로 추가한 사용자 이름 및 암호로 대체합니다. 예:

   ```
   --index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
   ```

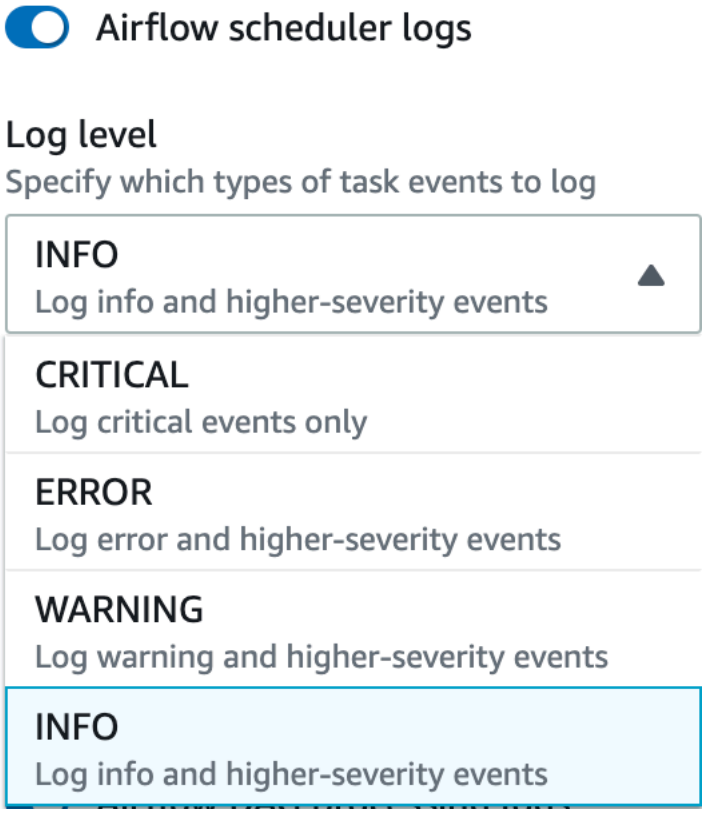
3. `requirements.txt` 파일에 라이브러리를 추가합니다. 예:

   ```
   --index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
   my-private-package==1.2.3
   ```

Amazon MWAA 콘솔에서 로그를 활성화합니다.

Amazon MWAA 환경의 실행 역할에는 로그를 로그로 전송할 수 있는 권한이 필요합니다. CloudWatch 실행 역할의 권한을 업데이트하려면 Amazon MWAA 실행 역할 단원을 참조하십시오.

INFO, WARNING, ERROR, 또는 CRITICAL 수준에서 Apache Airflow 로그를 활성화할 수 있습니다. 로그 수준을 선택하면 Amazon MWAA에서 해당 수준 및 더 높은 모든 심각도 수준에 대한 로그를 전송합니다. 예를 들어, INFO 수준에서 로그를 활성화하면 Amazon MWAA는 INFO 로그 및, WARNINGERROR, CRITICAL 로그 수준을 Logs로 CloudWatch 보냅니다. 스케줄러가 requirements.txt를 위해 수신한 로그를 볼 수 있도록 INFO 수준에서 Apache Airflow 로그를 활성화하는 것이 좋습니다.



로그 콘솔에서 CloudWatch 로그 보기

워크플로우를 예약하고 dags 폴더를 구문 분석하는 스케줄러에 대한 Apache Airflow 로그를 볼 수 있습니다. 다음 단계는 Amazon MWAA 콘솔에서 스케줄러의 로그 그룹을 열고 로그 콘솔에서 Apache Airflow 로그를 보는 방법을 설명합니다. CloudWatch

requirements.txt에 대한 로그를 보려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.

2. 환경을 선택합니다.
3. 모니터링 창에서 Airflow 스케줄러 로그 그룹을 선택합니다.
4. 로그 스트림에서 `requirements_install_ip` 로그를 선택합니다.
5. `/usr/local/airflow/.local/bin`에서 환경에 설치된 패키지 목록을 볼 수 있습니다. 예:

```
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. 패키지 목록을 검토하고 설치 중에 오류가 발생했는지 여부를 검토합니다. 문제가 발생한 경우, 다음과 비슷한 오류가 표시될 수 있습니다.

```
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))
```

Apache Airflow UI에서 오류 보기

Apache Airflow UI를 확인하여 오류가 다른 문제와 관련이 있는지 확인하는 것도 좋습니다. Amazon MWAA의 Apache Airflow에서 발생할 수 있는 가장 일반적인 오류는 다음과 같습니다.

```
Broken DAG: No module named x
```

Apache Airflow UI에 이 오류가 표시되면 `requirements.txt` 파일에 필수 종속성이 없는 것일 수 있습니다.

Apache Airflow에 로그인

Apache 에어플로우 UI를 보려면 AWS Identity and Access Management (IAM) 의 AWS 계정에 대한 아파치 에어플로우 UI 액세스 정책: 아마존 MWAA 액세스 WebServer 권한이 필요합니다.

Apache Airflow UI에 액세스하려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.

2. 환경을 선택합니다.
3. Airflow UI 열기를 선택합니다.

예제 **requirements.txt** 시나리오

requirements.txt에 다양한 형식을 믹스하여 매치할 수 있습니다. 다음 예제에서는 여러 가지 방법을 조합하여 엑스트라를 설치합니다.

Example PyPi.org의 추가 정보 및 공개 URL

사용자 지정 PEP 503 호환 리포지토리 URL과 같은 공개 URL의 패키지 PyPi 외에도.org에서 패키지를 지정할 때는 이 --index-url 옵션을 사용해야 합니다.

```
aws-batch == 0.6
phoenix-letter >= 0.3

--index-url http://dist.repoze.org/zope2/2.10/simple
    zopelib
```

Amazon Managed Workflows for Apache Airflow에 대한 모니터링 및 지표

모니터링은 Apache Airflow용 Amazon 관리형 워크플로 및 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. AWS 다중 지점 장애가 발생할 경우 더 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분에서 모니터링 데이터를 수집하는 것이 좋습니다. 이 주제에서는 Amazon MWAA 환경을 모니터링하고 잠재적 이벤트에 대응하기 위해 AWS 제공하는 리소스를 설명합니다.

> **Note**
>
> Apache Airflow 지표 및 로깅에는 표준 Amazon CloudWatch 요금이 적용됩니다.

Apache Airflow 모니터링에 대한 자세한 내용은 Apache Airflow 문서 웹 사이트의 로깅 및 모니터링을 참조하십시오.

Sections

Amazon MWAA의 모니터링 개요

이 페이지에서는 Apache Airflow용 Amazon 관리형 워크플로 환경을 모니터링하는 데 사용되는 AWS 서비스를 설명합니다.

목차

아마존 CloudWatch 개요

CloudWatch AWS 서비스에서 게시한 지표 및 차원을 기반으로 통계를 검색할 수 있는 서비스용 지표 리포지토리입니다. 이러한 지표를 사용하여 경보를 구성하고 통계를 계산한 다음 Amazon CloudWatch 콘솔에서 환경의 상태를 평가하는 데 도움이 되는 대시보드에 데이터를 표시할 수 있습니다.

Apache Airflow는 이미 Apache Airflow용 Amazon 관리형 워크플로우 환경에 대한 StatSD 지표를 Amazon에 전송하도록 설정되어 있습니다. CloudWatch

자세히 알아보려면 Amazon이란 무엇입니까 CloudWatch? 를 참조하십시오. .

AWS CloudTrail 개요

CloudTrail Amazon MWAA에서 사용자, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 감사 AWS 서비스입니다. 에서 수집한 정보를 사용하여 Amazon MWAA에 이루어진 요청 CloudTrail, 요청한 IP 주소, 요청한 사람, 요청 시기, 감사 로그에서 확인할 수 있는 추가 세부 정보를 확인할 수 있습니다.

자세한 내용은 무엇입니까를 참조하십시오. AWS CloudTrail.

감사 로그인 보기 AWS CloudTrail

AWS CloudTrail 계정을 만들면 AWS 계정에서 활성화됩니다. CloudTrail IAM 개체 또는 AWS 서비스(예: Apache Airflow용 Amazon Managed Workflows) 에서 수행한 활동을 기록하며, 이 활동은 이벤트로 기록됩니다. CloudTrail 콘솔에서 지난 90일간의 이벤트 기록을 보고, 검색하고, 다운로드할 수 있습니다. CloudTrail CloudTrail 아마존 MWAA 콘솔의 모든 이벤트와 아마존 MWAA API에 대한 모든 호출을 캡처합니다. `GetEnvironment` 또는 `PublishMetrics` 작업과 같은 읽기 전용 작업은 캡처하지 않습니다. 이 페이지에서는 Amazon MWAA의 이벤트를 모니터링하는 CloudTrail 데 사용하는 방법을 설명합니다.

목차

에서 트레일 생성 CloudTrail

Amazon MWAA 이벤트를 포함하여 AWS 계정에서 진행 중인 이벤트 기록을 보려면 트레일을 생성해야 합니다. 트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 트레일을 생성하지 않아도 CloudTrail 콘솔에서 사용 가능한 이벤트 기록을 볼 수 있습니다. 예를 들어 에서 수집한 CloudTrail 정보를 사용하여 Amazon MWAA에 요청한 내용, 요청한 IP 주소, 요청한 사람, 요청 시기, 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 계정 트레일 생성을 참조하십시오. AWS

이벤트 기록으로 CloudTrail 이벤트 보기

CloudTrail 콘솔에서 이벤트 기록을 확인하여 지난 90일 동안의 운영 및 보안 사고 문제를 해결할 수 있습니다. 예를 들어 AWS 계정의 리소스 (예: IAM 사용자 또는 기타 AWS 리소스) 의 생성, 수정 또는 삭제와 관련된 이벤트를 지역별로 볼 수 있습니다. 자세한 내용은 이벤트 기록을 통한 CloudTrail 이벤트 보기를 참조하십시오.

1. CloudTrail 콘솔을 엽니다.
2. 이벤트 기록을 선택합니다.
3. 보고자 하는 이벤트를 선택한 다음, 이벤트 세부 정보 비교를 선택합니다.

`CreateEnvironment`의 트레일 예시

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다.

CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며, 요청된 작업에 대한 정보 (예: 작업 날짜 및 시간, 요청 파라미터) 를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니며 특정 순서로 표시되지 않습니다. 다음 예는 권한 부족으로 인해 거부된 `CreateEnvironment` 작업에 대한 로그 항목입니다. `AirflowConfigurationOptions`의 값은 개인 정보 보호를 위해 수정된 것입니다.

```
{
    "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"
```

```
}
```

다음 단계

- CloudTrail 지원되는 서비스 및 통합의 CloudTrail 로그에서 수집된 이벤트 데이터에 대해 다른 AWS 서비스를 구성하는 방법을 알아보십시오.
- Amazon SNS 알림 구성에서 Amazon S3 버킷에 새 로그 파일을 CloudTrail 게시할 때 알림을 받는 방법을 알아보십시오. CloudTrail

Amazon에서 에어플로우 로그 보기 CloudWatch

Amazon MWAA는 아마존에 아파치 에어플로우 로그를 전송할 수 있습니다. CloudWatch 타사 도구를 사용하지 않고도 단일 위치에서 여러 환경의 로그를 확인하여 Apache Airflow 작업 지연 또는 워크플로우 오류를 쉽게 식별할 수 있습니다. Apache Airflow DAG 처리, 작업, 웹 서버, 작업자 로그인을 보려면 Apache Airflow용 Amazon Managed Workflow 콘솔에서 Apache Airflow 로그를 활성화해야 합니다. CloudWatch

목차

요금

- CloudWatch 표준 로그 요금이 적용됩니다. 자세한 내용은 CloudWatch 가격을 참조하십시오.

시작하기 전 준비 사항

- 로그인을 볼 수 있는 역할이 있어야 CloudWatch 합니다. 자세한 정보는 Amazon MWAA 환경 액세스을 참조하세요.

로그 유형

Amazon MWAA는 활성화한 각 Airflow 로깅 옵션에 대한 로그 그룹을 생성하고 해당 로그를 환경과 관련된 로그 그룹에 푸시합니다. CloudWatch 로그 그룹의 이름은 다음 형식으로 명명됩니다: `YourEnvironmentName-`**`LogType`**. 예를 들어, 환경 이름이 `Airflow-v202-Public`으로 명명되면 Apache Airflow 작업 로그가 `Airflow-v202-Public-`**`Task`**로 전송됩니다.

로그 유형	설명
`YourEnvironmentName-` **`DAGProcessing`**	DAG 프로세서 관리자(DAG 파일을 처리하는 스케줄러의 일부)의 로그.
`YourEnvironmentName-` **`Scheduler`**	Airflow 스케줄러가 생성하는 로그입니다.
`YourEnvironmentName-` **`Task`**	DAG가 생성하는 작업 로그입니다.
`YourEnvironmentName-` **`WebServer`**	Airflow 웹 인터페이스가 생성하는 로그입니다.
`YourEnvironmentName-` **`Worker`**	워크플로 및 DAG 실행의 일부로 생성된 로그입니다.

Apache Airflow 로그 활성화

`INFO`, `WARNING`, `ERROR`, 또는 `CRITICAL` 수준에서 Apache Airflow 로그를 활성화할 수 있습니다. 로그 수준을 선택하면 Amazon MWAA에서 해당 수준 및 더 높은 모든 심각도 수준에 대한 로그를 전송합니다. 예를 들어, `INFO` 수준에서 로그를 활성화하면 Amazon MWAA는 `INFO` 로그 및, `WARNINGERROR`, `CRITICAL` 로그 수준을 Logs로 CloudWatch 보냅니다.

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 편집을 선택합니다.
4. 다음을 선택합니다.
5. 다음 로깅 옵션 중 하나 이상을 선택합니다.
   a. 모니터링 창에서 Airflow 스케줄러 로그 그룹을 선택합니다.
   b. 모니터링 창에서 Airflow 웹 서버 로그 그룹을 선택합니다.
   c. 모니터링 창에서 Airflow 작업자 로그 그룹을 선택합니다.

d. 모니터링 창에서 Airflow DAG 프로세싱 로그 그룹을 선택합니다.

e. 모니터링 창에서 Airflow 작업 로그 그룹을 선택합니다.

f. 로그 수준에서 로깅 수준을 선택합니다.

6. 다음을 선택합니다.
7. 저장을 선택합니다.

Apache Airflow 로그 보기

다음 섹션에서는 콘솔에서 Apache Airflow 로그를 보는 방법을 설명합니다. CloudWatch

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 모니터링 창에서 로그 그룹을 선택합니다.
4. 로그 스트림을 선택합니다.

스케줄러 로그 예제

워크플로우를 예약하고 dags 폴더를 구문 분석하는 스케줄러에 대한 Apache Airflow 로그를 볼 수 있습니다. 다음 단계는 Amazon MWAA 콘솔에서 스케줄러의 로그 그룹을 열고 로그 콘솔에서 Apache Airflow 로그를 보는 방법을 설명합니다. CloudWatch

`requirements.txt`에 대한 로그를 보려면

1. Amazon MWAA 콘솔에서 환경 페이지를 엽니다.
2. 환경을 선택합니다.
3. 모니터링 창에서 Airflow 스케줄러 로그 그룹을 선택합니다.
4. 로그 스트림에서 `requirements_install_ip` 로그를 선택합니다.
5. `/usr/local/airflow/.local/bin`에서 환경에 설치된 패키지 목록을 볼 수 있습니다. 예:

```
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. 패키지 목록을 검토하고 설치 중에 오류가 발생했는지 여부를 검토합니다. 문제가 발생한 경우, 다음과 비슷한 오류가 표시될 수 있습니다.

```
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))
```

다음 단계

- Amazon CloudWatch CloudWatch 경보 사용에서 경보를 구성하는 방법을 알아보십시오.
- 대시보드 CloudWatch 사용에서 CloudWatch 대시보드를 만드는 방법을 알아보십시오.

Amazon MWAA의 대시보드 및 알람 모니터링

CloudWatch Amazon에서 사용자 지정 대시보드를 만들고 특정 지표에 대한 경보를 추가하여 Apache Airflow용 Amazon Managed Workflow 환경의 상태를 모니터링할 수 있습니다. 대시보드에 경보가 표시되면 ALARM 상태일 때 빨간색으로 표시되기 때문에 Amazon MWAA의 상태를 사전에 손쉽게 모니터링할 수 있습니다.

Apache Airflow는 DAG 프로세스 수, DAG 백 크기, 현재 실행 중인 작업, 작업 실패, 성공 등 여러 프로세스에 대한 지표를 공개합니다. 환경을 생성할 때 Airflow는 Amazon MWAA 환경에 대한 메트릭을 자동으로 전송하도록 구성됩니다. CloudWatch 이 페이지에서는 Amazon MWAA 환경에서 Airflow 지표에 대한 상태 대시보드를 생성하는 방법을 설명합니다. CloudWatch

목차

- 다음 단계

지표

Apache Airflow 버전에서 사용할 수 있는 모든 지표에 대해 사용자 지정 대시보드 및 경보를 생성할 수 있습니다. 각 지표는 Apache Airflow 핵심 성과 지표(KPI) 에 해당합니다. 지표 목록을 보려면 다음을 참조하십시오.

- 아파치 에어플로우 v2 환경 메트릭은 다음과 같습니다. CloudWatch

경보 상태 개요

지표 경보에는 다음과 같은 상태가 있을 수 있습니다.

- `OK` – 지표 또는 표현식이 정의된 임곗값 내에 있습니다.
- `ALARM` – 지표 또는 표현식이 정의된 임곗값을 벗어났습니다.
- `INSUFFICIENT_DATA` – 경보가 방금 시작되었거나 지표를 사용할 수 없거나 지표에서 경보 상태를 결정하는 데 사용할 수 있는 데이터가 충분하지 않습니다.

사용자 지정 대시보드 및 경보의 예

Amazon MWAA 환경에서 선택한 지표의 차트를 표시하는 사용자 지정 모니터링 대시보드를 구축할 수 있습니다.

이러한 지표에 대한 정보

다음 목록은 이 섹션의 튜토리얼 및 템플릿 정의를 통해 사용자 지정 대시보드에서 만든 각 지표를 설명합니다.

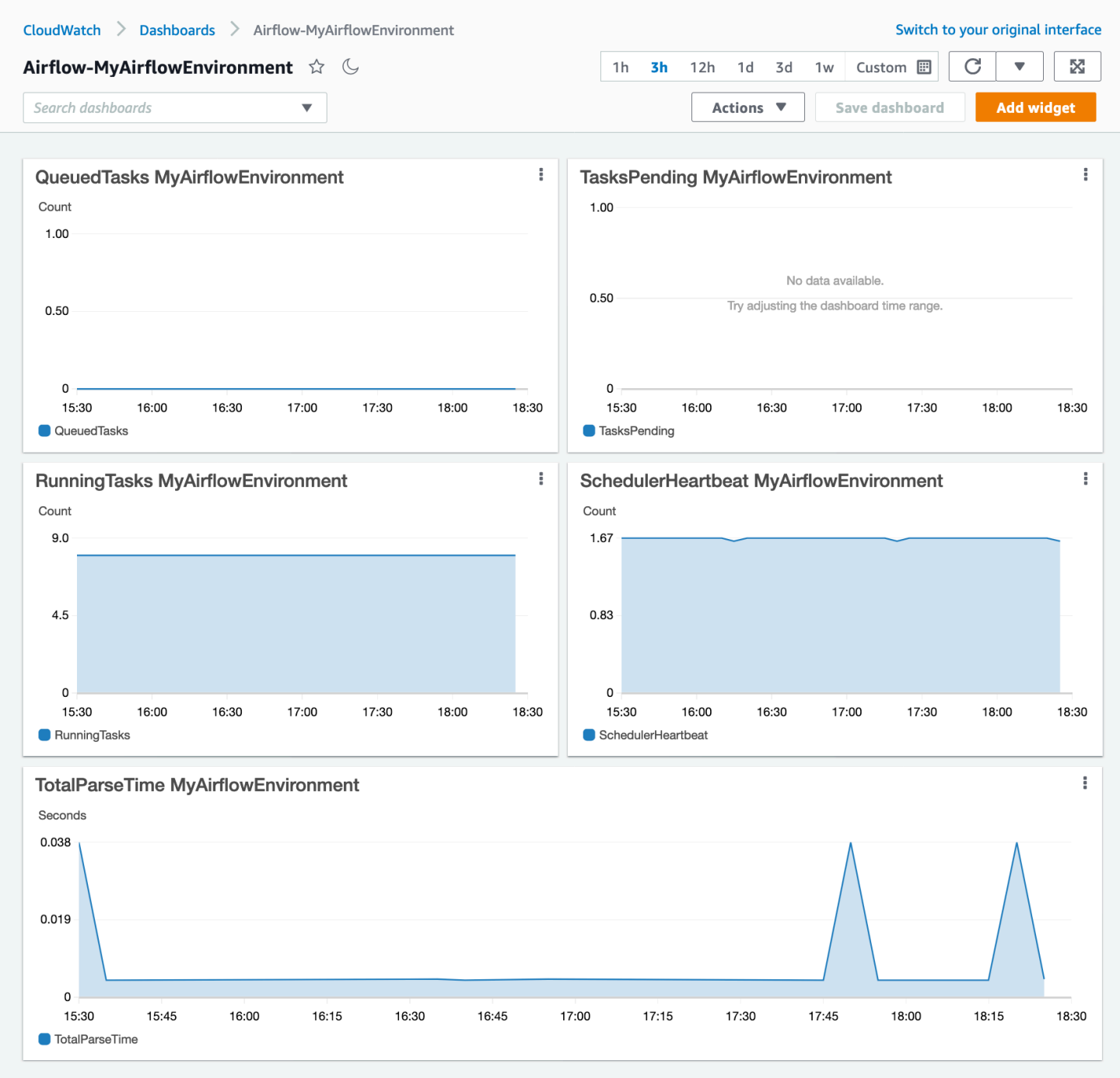
- QueuedTasks- 대기 상태인 작업 수. `executor.queued_tasks` Apache Airflow 지표에 해당합니다.
- TasksPending- 실행자에 보류 중인 작업 수 `scheduler.tasks.pending` Apache Airflow 지표에 해당합니다.

> **Note**
>
> Apache Airflow v2.2 이상에는 적용되지 않습니다.

- RunningTasks- 실행기에서 실행 중인 작업의 수 `executor.running_tasks` Apache Airflow 지표에 해당합니다.
- SchedulerHeartbeat- Apache Airflow가 스케줄러 작업에 대해 수행하는 체크인 수입니다. `scheduler_heartbeat` Apache Airflow 지표에 해당합니다.
- TotalParse시간 - 모든 DAG 파일을 한 번 스캔하고 가져오는 데 걸린 시간 (초) 입니다. `dag_processing.total_parse_time` Apache Airflow 지표에 해당합니다.

대시보드 정보

다음 이미지는 이 섹션의 튜토리얼 및 템플릿 정의로 생성된 모니터링 대시보드를 보여줍니다.



튜토리얼 사용 AWS

다음 AWS 자습서를 사용하여 현재 배포된 모든 Amazon MWAA 환경에 대한 상태 대시보드를 자동으로 생성할 수 있습니다. 또한 모든 Amazon MWAA 환경에서 건강하지 못한 작업자와 스케줄러 하트비트 장애에 대한 CloudWatch 경보를 생성합니다.

- CloudWatch Amazon MWAA를 위한 대시보드 자동화

사용 AWS CloudFormation

이 섹션의 AWS CloudFormation 템플릿 정의를 사용하여 에서 모니터링 대시보드를 만든 다음 CloudWatch 콘솔에 경보를 추가하여 지표가 특정 임계값을 초과할 경우 알림을 받을 수 있습니다. CloudWatch 이 템플릿 정의를 사용하여 스택을 생성하려면 콘솔에서 스택 생성을 참조하십시오. AWS CloudFormation 대시보드에 경보를 추가하려면 경보 사용을 참조하십시오.

```
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
                }
            }
        ]
    }'
```

지표 및 대시보드 삭제

Amazon MWAA 환경을 삭제하면 해당 대시보드도 삭제됩니다. CloudWatch 지표는 15개월 동안 저장되며 삭제할 수 없습니다. CloudWatch 콘솔은 Amazon MWAA 환경에서 가장 최신 인스턴스가 표시되도록 지표 검색을 지표가 마지막으로 수집된 후 2주로 제한합니다. 자세한 내용은 Amazon CloudWatch FAQ를 참조하십시오.

다음 단계

- 환경에 맞게 Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 쿼리하고 사용자 지정 지표를 게시하는 DAG를 만드는 방법을 알아봅니다. CloudWatch CloudWatch에서 DAG를 사용하여 사용자 지정 지표 작성

아파치 에어플로우 v2 환경 메트릭은 다음과 같습니다. CloudWatch

Apache Airflow v2는 이미 Apache Airflow용 Amazon 관리형 워크플로우 환경에 대한 StatSD 지표를 수집하여 Amazon으로 전송하도록 설정되어 있습니다. CloudWatch Apache Airflow가 전송하는 지표의 전체 목록은 Apache Airflow 참조 가이드의 지표 페이지에서 확인할 수 있습니다. 이 페이지에서는 에서 사용할 수 있는 Apache Airflow 지표와 콘솔에서 CloudWatch 지표에 액세스하는 방법을 설명합니다. CloudWatch

목차

용어

네임스페이스

네임스페이스는 서비스의 CloudWatch 메트릭을 담는 컨테이너입니다. AWS Amazon MWAA의 경우 네임스페이스는 AmazonMWAA입니다.

CloudWatch 메트릭

CloudWatch 지표는 특정한 시간에 따라 정렬된 데이터 요소 집합을 CloudWatch 나타냅니다.

Apache Airflow 지표

Apache Airflow 전용 지표

측정기준

차원은 지표의 보안 인증에 속하는 명칭/값 쌍입니다.

단위

통계에는 측정 단위가 포함되어 있습니다. Amazon MWAA의 경우, 단위에는 개수, 초, 밀리초가 포함됩니다. Amazon MWAA의 경우, 단위는 원래의 Airflow 지표의 단위를 기반으로 설정됩니다.

차원

이 섹션에서는 Apache Airflow 지표의 CloudWatch 차원 그룹화에 대해 설명합니다. CloudWatch

측정기준	설명
DAG	특정 Apache Airflow DAG 이름을 나타냅니다.
DAG 파일 이름	특정 Apache Airflow DAG 파일 이름을 나타냅니다.
함수	이 차원은 측정항목 그룹화를 개선하는 데 사용됩니다. CloudWatch
작업	스케줄러가 실행하는 Apache Airflow 작업을 나타냅니다. 항상 작업의 값을 갖습니다.
연산자	특정 Apache Airflow 연산자를 나타냅니다.
풀	특정 Apache Airflow 작업자 풀을 나타냅니다.
작업	특정 Apache Airflow 작업을 나타냅니다.
HostName	실행 중인 특정 Apache Airflow 프로세스의 호스트 이름을 나타냅니다.

콘솔에서 지표에 CloudWatch 액세스

이 섹션에서는 특정 DAG의 성능 CloudWatch 지표에 액세스하는 방법을 설명합니다.

차원에 대한 성능 지표를 보려면

1. CloudWatch 콘솔에서 지표 페이지를 엽니다.
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.

3. Amazon MWAA 네임스페이스를 선택합니다.
4. 모든 지표 탭에서 차원을 선택합니다. 예: DAG, 환경
5. 측정기준에 사용할 CloudWatch 측정항목을 선택합니다. TaskInstance성공 또는 TaskInstance지속 시간을 예로 들 수 있습니다. 모든 검색 결과를 그래프로 표시를 선택합니다.
6. 그래프 지표 탭을 선택하면 DAG, 환경, 작업과 같은 Apache Airflow 지표에 대한 성능 통계를 볼 수 있습니다.

Apache Airflow 지표는 다음에서 사용할 수 있습니다. CloudWatch

이 섹션에서는 전송 대상 Apache Airflow 지표 및 차원에 대해 설명합니다. CloudWatch

Apache Airflow 카운터

이 섹션의 Apache Airflow 지표에는 Apache Airflow 카운터에 대한 데이터가 포함되어 있습니다.

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
SLAMissed **Note** Apache Airflow v2.4.3 이상에서 사용할 수 있습니다.	sla_missed	개수	함수, 스케줄러	
FailedSLACallback **Note** Apache Airflow v2.4.3 이상에서 사용할 수 있습니다.	sla_callback_notification_failure	개수	함수, 스케줄러	
업데이트	dataset.updates	개수	함수, 스케줄러	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
Note Apache Airflow v2.6.3 이상에서 사용 가능.				
Orphaned Note Apache Airflow v2.6.3 이상에서 사용 가능.	dataset.orphaned	개수	함수, 스케줄러	
FailedCeleryTaskExecution Note Apache Airflow v2.4.3 이상에서 사용할 수 있습니다.	celery.execute_command.failure	개수	함수, Celery	
FilePathQueueUpdate개수 Note Apache Airflow v2.6.3 이상에서 사용 가능.	dag_processing.file_path_queue_update_count	개수	함수, 스케줄러	
CriticalSection바빠	scheduler.critical_section_busy	개수	함수, 스케줄러	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
DagBag사이즈	dagbag_size	개수	함수, DAG 프로세싱	
DagCallback예외	dag.callback_exceptions	개수	DAG, 모두	
실패한 SLA EmailAttempts	sla_email_notification_failure	개수	함수, 스케줄러	
TaskInstance완료되었습니다.	ti.finish.{dag_id}.{task_id}.{state}	개수	DAG, {dag_id} 작업, {task_id} 상태, {state}	
JobEnd	{job_name}_end	개수	작업, {job_name}	
JobHeartbeat실패	{job_name}_heartbeat_failure	개수	작업, {job_name}	
JobStart	{job_name}_start	개수	작업, {job_name}	
ManagerStalls	dag_processing.manager_stalls	개수	함수, DAG 프로세싱	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
OperatorFailures	operator_failures_{operator_name}	개수	연산자, {operator_name}	
OperatorSuccesses	operator_successes_{operator_name}	개수	연산자, {operator_name}	
OtherCallback카운트 Note Apache Airflow v2.6.3 이상에서 사용 가능.	dag_processing.other_callback_count	개수	함수, 스케줄러	
프로세스	dag_processing.processes	개수	함수, DAG 프로세싱	
SchedulerHeartbeat	scheduler_heartbeat	개수	함수, 스케줄러	
StartedTask인스턴스	ti.start.{dag_id}.{task_id}	개수	DAG, 모두 작업, 모두	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
SlaCallback개수	dag_processing.sla_callback_count Note Apache Airflow v2.6.3 이상에서 사용 가능.	개수	함수, 스케줄러	
TasksKilled외부적으로	scheduler.tasks.killed_externally	개수	함수, 스케줄러	
TaskTimeout오류	celery.task_timeout_error	개수	함수, Celery	
TaskInstanceCreatedUsing운영자	task_instance_created-{operator_name}	개수	연산자, {operator_name}	
TaskInstancePreviouslySucceeded	previously_succeeded	개수	DAG, 모두 작업, 모두	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
TaskInstance실패	ti_failures	개수	DAG, 모두 작업, 모두	
TaskInstance성공	ti_successes	개수	DAG, 모두 작업, 모두	
TaskRemoved프롬 데이	task_removed_from_dag.{dag_id}	개수	DAG, {dag_id}	
TaskRestored오늘	task_restored_to_dag.{dag_id}	개수	DAG, {dag_id}	
TriggersSucceeded Note Apache Airflow v2.7.2 이상에서 사용 가능.	triggers.succeeded	개수	함수, 트리거	
TriggersFailed Note Apache Airflow v2.7.2 이상에서 사용 가능.	triggers.failed	개수	함수, 트리거	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
TriggersBlockedMainThread Note Apache Airflow v2.7.2 이상에서 사용 가능.	triggers.blocked_main_thread	개수	함수, 트리거	
TriggerHeartbeat Note 아파치 에어플로우 v2.8.1 이상에서 사용 가능합니다.	트리거_하트비트	개수	함수, 트리거	
TaskInstanceCreatedUsing오퍼레이터	airflow.task_instance_created_{operator_name} Note Apache Airflow v2.7.2 이상에서 사용 가능.	개수	연산자, {operator_name}	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
ZombiesKilled	zombies_killed	개수	DAG, 모두 작업, 모두	

Apache Airflow 게이지

이 섹션의 Apache Airflow 지표에는 Apache Airflow 게이지에 대한 데이터가 포함되어 있습니다.

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
DAG 오류 FileRefresh	dag_file_refresh_error	개수	함수, DAG 프로세싱	
ImportErrors	dag_processing.import_errors	개수	함수, DAG 프로세싱	
Exception Failures	smart_sensor_operator.exception_failures	개수	함수, 스마트 센서 연산자	
ExecutedTasks	smart_sensor_operator.executed_tasks	개수	함수, 스마트 센서 연산자	
InfraFailures	smart_sensor_operator.infra_failures	개수	함수, 스마트 센서 연산자	
LoadedTasks	smart_sensor_operator.loaded_tasks	개수	함수, 스마트 센서 연산자	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
TotalParse시간	dag_processing.total_parse_time	초	함수, DAG 프로세싱	
TriggeredDag실행 Note Apache Airflow v2.6.3 이상에서 사용 가능.	dataset.triggered_dagruns	개수	함수, 스케줄러	
TriggersRunning Note Apache Airflow v2.7.2 이상에서 사용 가능.	triggers.running.*{hostname}*	개수	함수, 트리거 HostName, *{#####}*	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
PoolDeferred슬롯 Note Apache Airflow v2.7.2 이상에서 사용 가능.	pool.deferred_slots.{pool_name}	개수	풀, {pool_name}	
DAG FileProcessing LastRun SecondsAgo	dag_processing.last_run.seconds_ago.{dag_filename}	초	DAG 파일 이름, {dag_filename}	
OpenSlots	executor.open_slots	개수	함수, 실행기	
OrphanedTasks 입양	scheduler.orphaned_tasks.adopted	개수	함수, 스케줄러	
OrphanedTasks 클리어	scheduler.orphaned_tasks.cleared	개수	함수, 스케줄러	
PokedExceptions	smart_sensor_operator.poked_exception	개수	함수, 스마트 센서 연산자	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
PokedSuccess	smart_sensor_operator.poked_success	개수	함수, 스마트 센서 연산자	
PokedTasks	smart_sensor_operator.poked_tasks	개수	함수, 스마트 센서 연산자	
PoolFailures	pool.open_slots.{pool_name}	개수	풀, {pool_name}	
PoolStarving태스크	pool.starving_tasks.{pool_name}	개수	풀, {pool_name}	
PoolOpen슬롯	pool.open_slots.{pool_name}	개수	풀, {pool_name}	
PoolQueued슬롯	pool.queued_slots.{pool_name}	개수	풀, {pool_name}	
PoolRunning슬롯	pool.running_slots.{pool_name}	개수	풀, {pool_name}	
Processor Timeouts	dag_processing.processor_timeouts	개수	함수, DAG 프로세싱	
QueuedTasks	executor.queued_tasks	개수	함수, 실행기	
RunningTasks	executor.running_tasks	개수	함수, 실행기	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
TasksExecutable	scheduler.tasks.executable	개수	함수, 스케줄러	
TasksPending **Note** Apache Airflow v2.2 이상에는 적용되지 않습니다.	scheduler.tasks.pending	개수	함수, 스케줄러	
TasksRunning	scheduler.tasks.running	개수	함수, 스케줄러	
TasksStarving	scheduler.tasks.starving	개수	함수, 스케줄러	
TasksWithoutDagRun	scheduler.tasks.without_dagrun	개수	함수, 스케줄러	

Apache Airflow 타이머

이 섹션의 Apache Airflow 지표에는 Apache Airflow 타이머에 대한 데이터가 포함되어 있습니다.

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
CollectDBDags	collect_db_dags	밀리초	함수, DAG 프로세싱	
CriticalSection소요 시간	scheduler.critical_section_duration	밀리초	함수, 스케줄러	
CriticalSectionQueryDuration **Note** Apache Airflow v2.5.1 이상에서 사용 가능.	scheduler.critical_section_query_duration	밀리초	함수, 스케줄러	
DAG DependencyCheck	dagrun.dependency-check.{dag_id}	밀리초	DAG, {dag_id}	
DAG DurationFailed	dagrun.duration.failed.{dag_id}	밀리초	DAG, {dag_id}	
DAG DurationSuccess	dagrun.duration.success.{dag_id}	밀리초	DAG, {dag_id}	
DAG FileProcessing LastDuration	dag_processing.las	초	DAG 파일 이름, {dag_filename}	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
	t_duration.{dag_filename}			
DAG ScheduleDelay	dagrun.schedule_delay.{dag_id}	밀리초	DAG, {dag_id}	
FirstTaskSchedulingDelay	dagrun.{dag_id}.first_task_scheduling_delay	밀리초	DAG, {dag_id}	
SchedulerLoop 소요 시간 Note Apache Airflow v2.5.1 이상에서 사용 가능.	scheduler.scheduler_loop_duration	밀리초	함수, 스케줄러	
TaskInstance소요 시간	dag.{dag_id}.{task_id}.duration	밀리초	DAG, {dag_id} 작업, {task_id}	

CloudWatch 메트릭	Apache Airflow 지표	단위	측정기준	
TaskInstanceQueuedDuration	dag.{dag_id}.{t .queued_duration Note Apache Airflow v2.7.2 이상에서 사용 가능.	밀리초	DAG, {dag_id} 작업, {task_id}	
TaskInstanceScheduledDuration Note Apache Airflow v2.7.2 이상에서 사용 가능.	dag.{dag_id}.{t .scheduled_duration	밀리초	DAG, {dag_id} 작업, {task_id}	

보고할 지표 선택

다음 Amazon MWAA 구성 옵션을 사용하여 Apache Airflow로 CloudWatch 내보내거나 Apache Airflow에서 차단할 Apache Airflow 지표를 선택할 수 있습니다.

- **`metrics.metrics_allow_list`**— 환경에서 내보낼 지표를 선택하는 데 사용할 수 있는 쉼표로 구분된 접두사 목록입니다. CloudWatch Apache Airflow가 가용한 모든 지표를 전

송하지 않고 대신 요소의 하위 집합을 선택하도록 하려면 이 옵션을 사용합니다. 예를 들어 `scheduler,executor,dagrun`입니다.

- **`metrics.metrics_block_list`**— 목록의 요소로 시작하는 지표를 필터링하기 위한, 쉼표로 구분된 접두사 목록입니다. 예를 들어 `scheduler,executor,dagrun`입니다.

`metrics.metrics_allow_list` 및 `metrics.metrics_block_list`를 모두 구성하면 Apache Airflow는 `metrics.metrics_block_list`을 무시합니다. `metrics.metrics_block_list`만 구성하고 `metrics.metrics_allow_list`는 구성하지 않은 경우, Apache Airflow는 사용자가 `metrics.metrics_block_list`에서 지정한 요소를 필터링합니다.

> **Note**
>
> `metrics.metrics_allow_list`및 `metrics.metrics_block_list` 구성 옵션은 Apache Airflow v2.6.3 이상에만 적용됩니다. 이전 버전의 Apache Airflow의 경우 를 대신 사용하십시오. `metrics.statsd_allow_list` `metrics.statsd_block_list`

다음 단계

- 에서 환경 상태 지표를 게시하는 데 사용되는 Amazon MWAA API 작업을 살펴보십시오. PublishMetrics

Amazon MWAA의 컨테이너, 대기열 및 데이터베이스 지표

Apache Airflow 지표 외에도 원시 데이터를 수집하고 데이터를 읽기 가능한 거의 실시간 지표로 처리하는 를 사용하여 CloudWatch Amazon Managed Workflow for Apache Airflow 환경의 기본 구성 요소를 모니터링할 수 있습니다. 이러한 환경 지표를 사용하면 주요 성과 지표에 대한 가시성이 좋아져 환경의 크기를 적절히 조정하고 워크플로와 관련된 문제를 디버깅하는 데 도움이 됩니다. 이들 지표는 Amazon MWAA에서 지원되는 모든 Apache Airflow 버전에 적용됩니다.

Amazon MWAA는 각 Amazon Elastic Container Service(Amazon ECS) 및 Amazon Aurora PostgreSQL 인스턴스에 대한 CPU 및 메모리 사용률을 제공하고, 메시지 수와 가장 오래된 메시지의 보관기간에 대한 Amazon Simple Queue Service(Amazon SQS) 지표를 제공하며, 데이터베이스 연결, 디스크 대기열 길이, 쓰기 동작, 지연 시간, 처리량에 관한 Amazon Relational Database Service(Amazon RDS) 지표, 그리고 Amazon RDS 프록시 지표를 제공합니다. 이러한 지표에는 기본 작업자, 추가 작업자, 스케줄러 및 웹 서버의 수도 포함됩니다.

이러한 통계는 15개월간 보관되므로 기록 정보를 보고 일정에 오류가 발생하는 이유를 더 잘 파악하고 근본적인 문제를 해결할 수 있습니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

주제

용어

네임스페이스

네임스페이스는 서비스의 CloudWatch 메트릭을 담는 컨테이너입니다 AWS . Amazon MWAA의 네임스페이스는 `AWS/MWAA`입니다.

CloudWatch 메트릭

CloudWatch 지표는 특정한 시간에 따라 정렬된 데이터 요소 집합을 CloudWatch 나타냅니다.

측정기준

차원은 지표의 보안 인증에 속하는 명칭/값 쌍입니다.

단위

통계에는 측정 단위가 포함되어 있습니다. Amazon MWAA의 경우 단위에 개수가 포함됩니다.

차원

이 섹션에서는 Amazon MWAA 지표의 CloudWatch 차원 그룹화에 대해 설명합니다. CloudWatch

측정기준	설명
클러스터	Amazon MWAA 환경에서 Apache Airflow 구성 요소를 실행하는 데 사용하는 최소 3개의 Amazon ECS 컨테이너(스케줄러, 작업자, 웹 서버)에 대한 지표입니다.

측정기준	설명
대기열	스케줄러와 작업자를 분리하는 Amazon SQS 대기열의 지표입니다. 작업자가 메시지를 읽을 때는 처리 중인 것으로 간주되어 다른 작업자가 사용할 수 없습니다. 메시지가 표시 제한 시간인 12시간 이전에 삭제되지 않으면 다른 작업자가 메시지를 읽을 수 있습니다.
데이터베이스	Amazon MWAA에서 사용하는 Aurora 클러스터를 측정합니다. 여기에는 기본 데이터베이스 인스턴스에 대한 지표와 읽기 동작을 지원하는 읽기 전용 복제본이 포함됩니다. Amazon MWAA는 READER 및 WRITER 인스턴스 모두에 대한 데이터베이스 지표를 게시합니다.

콘솔에서 지표에 액세스 CloudWatch

이 섹션에서는 Amazon MWAA 지표에 액세스하는 방법을 설명합니다. CloudWatch

차원에 대한 성능 지표를 보려면

1. 콘솔에서 지표 페이지를 엽니다. CloudWatch
2. AWS 지역 선택기를 사용하여 지역을 선택합니다.
3. AWS/MWAA 네임스페이스를 선택합니다.
4. 모든 지표 탭에서 차원을 선택합니다. 예: 클러스터
5. 측정기준에 사용할 CloudWatch 측정항목을 선택합니다. CPU 사용률을 예로 들 NumSchedulers수 있습니다. 이어서, 모든 검색 결과를 그래프로 표시를 선택합니다.
6. 그래프로 표시된 지표 탭을 선택하여 성능 지표를 확인합니다.

지표 목록

다음 표에는 Amazon MWAA의 클러스터, 대기열 및 데이터베이스 서비스 지표가 나열되어 있습니다. Amazon ECS, Amazon SQS 또는 Amazon RDS에서 직접 내보낸 지표에 대한 설명을 보려면 해당 문서 링크를 선택합니다.

주제

클러스터 지표

다음 지표는 각 스케줄러, 기본 작업자, 추가 작업자 및 웹 서버에 적용됩니다. 각 클러스터 지표에 대한 자세한 내용과 설명은 Amazon ECS 개발자 가이드의 사용 가능한 지표 및 차원을 참조하십시오.

네임스페이스	지표	Unit
AWS/MWAA	CPUUtilization	%
AWS/MWAA	MemoryUtilization	%

추가 작업자 및 웹 서버 컨테이너의 수 평가

다음 절차에 설명된 대로 클러스터 차원에 제공된 구성 요소 지표를 사용하여 특정 시점에 환경에서 사용하고 있는 추가 작업자 또는 웹 서버 수를 평가할 수 있습니다. CPU 사용률 또는 MemoryUtilization지표를 그래프로 표시하고 통계 유형을 샘플 수로 설정하면 이 작업을 수행할 수 있습니다. 결과 값은 `AdditionalWorker` 구성 요소의 총 `RUNNING` 작업 수입니다. 환경에서 사용하는 추가 작업자 인스턴스의 수를 이해하면 환경이 확장되는 방식을 파악하고 추가 작업자 수를 최적화하는 데 도움이 될 수 있습니다.

Workers

를 사용하는 추가 작업자 수를 평가하려면 AWS Management Console

1. AWS/MWAA 네임스페이스를 선택합니다.
2. 모든 지표 탭에서 클러스터 차원을 선택합니다.
3. 클러스터 차원에서 에 AdditionalWorker대해 CPU 사용률 또는 지표를 선택합니다. MemoryUtilization
4. 그래프로 표시된 지표 탭에서 기간을 1분으로 T설정하고, 통계를 샘플 수로 설정합니다.

Web servers

를 사용하여 추가 웹 서버 수를 평가하려면 AWS Management Console

1. AWS/MWAA 네임스페이스를 선택합니다.
2. 모든 지표 탭에서 클러스터 차원을 선택합니다.
3. 클러스터 차원에서 에 AdditionalWebservers대해 CPU 사용률 또는 지표를 선택합니다. MemoryUtilization
4. 그래프로 표시된 지표 탭에서 기간을 1분으로 T설정하고, 통계를 샘플 수로 설정합니다.

자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 서비스 RUNNING 작업 수를 참조하십시오.

데이터베이스 지표

다음 지표는 Amazon MWAA 환경과 연결된 각 데이터베이스 인스턴스에 적용됩니다.

네임스페이스	지표	Unit
AWS/MWAA	CPUUtilization	%
AWS/MWAA	DatabaseConnections	개수
AWS/MWAA	DiskQueueDepth	개수
AWS/MWAA	FreeableMemory	바이트
AWS/MWAA	VolumeWriteIOPS	5분당 개수
AWS/MWAA	WriteIOPS	초당 개수
AWS/MWAA	WriteLatency	초
AWS/MWAA	WriteThroughput	초당 바이트

대기열 지표

다음 대기열 지표의 단위 및 설명에 대한 자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS에서 사용 가능한 CloudWatch 지표를 참조하십시오.

네임스페이스	지표	Unit
AWS/MWAA	ApproximateAgeOfOldestTask	초
AWS/MWAA	RunningTasks	개수
AWS/MWAA	QueuedTasks	개수

Application Load Balancer 지표

Application Load Balancer 지표는 사용자 환경에서 실행되는 웹 서버에 적용됩니다. Amazon MWAA는 이러한 지표를 사용하여 트래픽량에 따라 웹 서버를 확장합니다. 다음 로드 밸런서 지표의 단위 및 설명에 대한 자세한 내용은 애플리케이션 로드 밸런서 사용 설명서에서 Application Load Balancer의 CloudWatch 지표를 참조하십시오.

네임스페이스	지표	Unit
AWS/MWAA	ActiveConnectionCount	개수

Amazon Managed Workflows for Apache Airflow의 보안

클라우드 보안이 최우선 AWS 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 귀하 (고객) AWS 와 귀하 (고객) 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. Amazon MWAA에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오.
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon Managed Workflows for Apache Airflow를 사용할 때 공동 책임 모델의 적용 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 Amazon MWAA를 구성하는 방법을 보여줍니다. 또한 Amazon MWAA 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

이 섹션:

- Amazon Managed Workflows for Apache Airflow의 데이터 보호
- AWS Identity and Access Management
- Amazon Managed Workflows for Apache Airflow의 규정 준수 확인
- Amazon Managed Workflows for Apache Airflow의 복원성
- Amazon MWAA의 인프라 보안
- Amazon MWAA의 구성 및 취약성 분석
- Amazon MWAA의 보안 모범 사례

Amazon Managed Workflows for Apache Airflow의 데이터 보호

AWS 공동 책임 모델 공동 책임 모델 이 모델에 설명된 대로 은 (AWS 는) 모든 모델을 실행하는 글로벌 인프라를 보호하는 역할을 합니다. AWS 클라우드사용자는 인프라에서 호스팅되는 콘텐츠를 관리

해야 합니다. 이 콘텐츠에는 사용하는 AWS 서비스 서비스의 보안 구성과 관리 작업이 포함되어 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2 이상을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- AWS 서비스 내의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔 AWS CLI, API 또는 SDK를 사용하여 Amazon MWAA 또는 기타 AWS 서비스를 사용하는 경우가 포함됩니다. AWS 이름에 사용되는 태그 또는 자유 형식 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시켜서는 안됩니다.

Amazon MWAA에서 암호화

다음 단원에서는 Amazon MWAA에서 저장 데이터와 전송 중인 데이터를 보호하는 방법을 설명합니다. 이 정보를 사용하여 Amazon MWAA와 통합하여 저장된 데이터를 AWS KMS 암호화하는 방법과 전송 시 전송 계층 보안 (TLS) 프로토콜을 사용하여 데이터를 암호화하는 방법을 알아보십시오.

주제

- 저장된 데이터 암호화
- 전송 중 암호화

저장된 데이터 암호화

Amazon MWAA에서 저장 데이터는 서비스가 영구 매체에 저장하는 데이터입니다.

저장 데이터 암호화를 위해 AWS 소유 키를 사용하거나, 선택적으로 환경을 만들 때 추가 암호화를 위해 고객 관리형 키를 제공할 수도 있습니다. 고객 관리형 KMS 키를 사용하려면 해당 환경에서 사용 중인 다른 AWS 리소스 및 서비스와 동일한 계정에 있어야 합니다.

고객 관리형 KMS 키를 사용하려면 키 정책에 CloudWatch 액세스하는 데 필요한 정책 설명을 첨부해야 합니다. 사용자 환경에 고객 관리 KMS 키를 사용하는 경우, Amazon MWAA는 사용자를 대신하여 4가지 권한을 부여합니다. Amazon MWAA에서 고객 관리형 KMS 키에 연결하는 권한에 대한 자세한 내용은 데이터 암호화를 위한 고객 관리형 키를 참조하십시오.

고객 관리형 KMS 키를 지정하지 않는 경우 기본적으로 Amazon MWAA는 AWS 소유한 KMS 키를 사용하여 데이터를 암호화하고 해독합니다. Amazon MWAA에서 데이터 암호화를 관리하려면 AWS 소유한 KMS 키를 사용하는 것이 좋습니다.

> **Note**
>
> Amazon MWAA에서 AWS 소유하거나 고객이 관리하는 KMS 키의 저장 및 사용에 대한 비용을 지불합니다. 자세한 내용은 AWS KMS 요금을 참조하십시오.

암호화 아티팩트

Amazon MWAA 환경을 생성할 때는 AWS 소유 키 또는 고객 관리형 키를 지정하여 저장 중 암호화에 사용되는 암호화 아티팩트를 지정합니다. Amazon MWAA는 사용자가 지정한 키에 필요한 권한 부여를 추가합니다.

Amazon S3 - Amazon S3 데이터는 서버 측 암호화(SSE)를 이용하여 객체 수준에서 암호화됩니다. Amazon S3 암호화 및 복호화는 DAG 코드 및 지원 파일이 저장되는 Amazon S3 버킷에서 수행됩니다. 객체는 Amazon S3에 업로드될 때 암호화되고 Amazon MWAA 환경에 다운로드될 때 복호화됩니다. 고객 관리 KMS 키를 사용하는 경우, Amazon MWAA는 기본 설정으로 이 키를 사용하여 Amazon S3 버킷의 데이터를 읽고 해독합니다.

CloudWatch 로그 — AWS 소유한 KMS 키를 사용하는 경우 로그로 전송되는 Apache Airflow 로그는 Logs 소유의 KMS 키와 함께 서버 측 암호화 (SSE) 를 사용하여 암호화됩니다. CloudWatch CloudWatch AWS 고객 관리형 KMS 키를 사용하는 경우 KMS 키에 키 정책을 추가하여 로그에서 키를 사용하도록 허용해야 합니다. CloudWatch

Amazon SQS - Amazon MWAA는 사용자 환경을 위해 하나의 Amazon SQS 대기열을 생성합니다. Amazon MWAA는 AWS 소유한 KMS 키 또는 고객이 지정한 고객 관리형 KMS 키를 사용한 서버 측 암호화 (SSE) 를 사용하여 대기열에서 주고 받는 데이터를 암호화합니다. AWS 소유한 KMS 키를 사용하든 고객 관리형 KMS 키를 사용하든 관계없이 실행 역할에 Amazon SQS 권한을 추가해야 합니다.

Aurora PostgreSQL - Amazon MWAA는 사용자 환경에 맞는 PostgreSQL 클러스터 하나를 생성합니다. Aurora PostgreSQL은 서버 측 암호화 (SSE) 를 사용하여 AWS 소유하거나 고객이 관리하는 KMS 키로 콘텐츠를 암호화합니다. 고객 관리 KMS 키를 사용하는 경우, Amazon RDS는 이 키에 최소 두 가지 권한(클러스터용과 데이터베이스 인스턴스용)을 추가합니다. 고객 관리 KMS 키를 여러 환경에서 사용하기로 선택한 경우, Amazon RDS에서 추가 권한을 부여할 수 있습니다. 자세한 내용은 Amazon RDS의 데이터 보호를 참조하십시오.

전송 중 암호화

전송 중 데이터는 네트워크를 통해 이동할 때 도청당할 수 있는 데이터로 칭하기도 합니다.

전송 계층 보안 (TLS) 은 사용자 환경의 Apache Airflow 구성 요소와 Amazon MWAA와 통합되는 다른 서비스 (예: Amazon S3) 간에 AWS 전송되는 Amazon MWAA 객체를 암호화합니다. Amazon S3 암호화에 대한 자세한 내용은 암호화를 사용한 데이터 보호를 참조하십시오.

암호화를 위한 고객 관리형 키 사용

사용자 환경의 데이터 암호화를 위해 고객 관리형 키를 옵션으로 제공할 수 있습니다. 고객 관리형 KMS 키는 Amazon MWAA 환경 인스턴스 및 워크플로용 리소스를 저장하는 Amazon S3 버킷과 동일한 리전에 생성해야 합니다. 지정하는 고객 관리형 키가 환경을 구성하는 데 사용하는 계정이 아닌 다른 계정에 있는 경우, 크로스 계정 액세스를 위해 ARN을 이용하여 해당 키를 지정해 주어야 합니다. 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하십시오.

지원되는 항목

AWS KMS 기능:	지원
AWS KMS 키 ID 또는 ARN	예
AWS KMS 키 별칭	아니요
AWS KMS 다중 리전 키	아니요

암호화를 위한 권한 사용

이 단원에서는 데이터를 암호화하고 복호화하기 위해 Amazon MWAA에서 고객 관리형 KMS 키에 연결하는 권한을 설명합니다.

작동 방식

고객 관리형 KMS AWS KMS 키에 대해 지원되는 리소스 기반 액세스 제어 메커니즘에는 키 정책과 부여라는 두 가지가 있습니다.

키 정책은 권한이 대부분 정적이고 동기 서비스 모드에 이용되는 경우에 사용합니다. 권한 부여는 서비스에서 자체 또는 다른 계정에 대해 서로 다른 액세스 권한을 정의해야 하는 경우와 같이 보다 동적이고 세분화된 권한이 필요한 경우에 사용됩니다.

Amazon MWAA는 네 가지 권한 부여 정책을 사용하여 고객 관리형 KMS 키에 연결합니다. 이는 CloudWatch 로그, Amazon SQS 대기열, Aurora PostgreSQL 데이터베이스 데이터베이스, Secrets Manager 시크릿, Amazon S3 버킷 및 DynamoDB 테이블에 저장된 데이터를 암호화하는 환경에 필요한 세분화된 권한 때문입니다.

Amazon MWAA 환경을 생성하고 고객 관리형 KMS 키를 지정하면, Amazon MWAA는 사용자의 고객 관리형 KMS 키에 권한 부여 정책을 연결합니다. 이러한 정책을 통해 `airflow.region}.amazonaws.com`의 Amazon MWAA는 고객 관리형 KMS 키를 사용하여 Amazon MWAA 소유 리소스를 사용자를 대신하여 암호화할 수 있습니다.

Amazon MWAA는 사용자를 대신해서 추가 권한을 생성하여 지정된 KMS 키에 연결합니다. 여기에는 환경을 삭제할 경우 권한 부여를 폐기하고, 고객 관리형 KMS 키를 CSE (클라이언트 측 암호화) 에 사용하고, Secrets Manager의 고객 관리 키로 보호되는 비밀에 액세스하는 데 필요한 AWS Fargate 실행 역할에 사용하는 정책이 포함됩니다.

권한 부여 정책

Amazon MWAA는 사용자를 대신하여 고객 관리형 KMS 키에 다음과 같은 리소스 기반 정책 권한을 추가합니다. 이러한 정책을 통해 권한 피부여자와 주체(Amazon MWAA)가 정책에 정의된 작업을 수행할 수 있습니다.

권한 1: 데이터 플레인 리소스를 생성하는 데 사용

```
{
            "Name": "mwaa-grant-for-env-mgmt-role-environment name",
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:CreateGrant",
```

```
            "kms:DescribeKey",
            "kms:RetireGrant"
          ]
        }
```

권한 2: **ControllerLambdaExecutionRole** 액세스에 사용

```
{
          "Name": "mwaa-grant-for-lambda-exec-environment name",
          "GranteePrincipal": "airflow.region.amazonaws.com",
          "RetiringPrincipal": "airflow.region.amazonaws.com",
          "Operations": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey",
            "kms:RetireGrant"
          ]
        }
```

권한 3: **CfnManagementLambdaExecutionRole** 액세스에 사용

```
{
            "Name": " mwaa-grant-for-cfn-mgmt-environment name",
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:DescribeKey"
            ]
          }
```

권한 4: 백엔드 암호에 액세스하기 위한 Fargate 실행 역할에 사용

```
{
```

```
            "Name": "mwaa-fargate-access-for-environment name",
            "GranteePrincipal": "airflow.region.amazonaws.com",
            "RetiringPrincipal": "airflow.region.amazonaws.com",
            "Operations": [
              "kms:Encrypt",
              "kms:Decrypt",
              "kms:ReEncrypt*",
              "kms:GenerateDataKey*",
              "kms:DescribeKey",
              "kms:RetireGrant"
            ]
          }
```

고객 관리형 키에 대한 키 정책 연결

Amazon MWAA에서 자체 고객 관리형 KMS 키를 사용하기로 선택한 경우, Amazon MWAA에서 데이터를 암호화하는 데 사용할 수 있도록 다음 정책을 그 키에 연결해 주어야 합니다.

Amazon MWAA 환경에 사용한 고객 관리형 KMS 키가 아직 작동하도록 구성되지 않은 경우 암호화된 로그를 허용하도록 키 정책을 업데이트해야 합니다. CloudWatch CloudWatch 자세한 내용은 서비스 사용 중 로그 데이터 암호화를 참조하십시오. CloudWatch AWS Key Management Service

다음 예는 CloudWatch 로그의 키 정책을 나타냅니다. 리전에 대해 제공된 샘플 값을 대체합니다.

```
{
          "Effect": "Allow",
          "Principal": {
          "Service": "logs.us-west-2.amazonaws.com"
        },
        "Action": [
          "kms:Encrypt*",
          "kms:Decrypt*",
          "kms:ReEncrypt*",
          "kms:GenerateDataKey*",
          "kms:Describe*"
        ],
        "Resource": "*",
        "Condition": {
          "ArnLike": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-west-2:*:*"
            }
          }
```

```
        }
```

AWS Identity and Access Management

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. AWS IAM 관리자는 Amazon Managed Workflows for Apache Airflow를 사용할 수 있는 대상자의 인증(로그인) 및 승인(권한) 업무를 관리합니다. IAM은 추가 AWS 비용 없이 사용할 수 있는 서비스입니다.

이 주제에서는 Amazon MWAA가 IAM AWS Identity and Access Management (IAM) 을 사용하는 방식에 대한 기본적인 개요를 제공합니다. Amazon MWAA에 대한 액세스 관리에 관한 자세한 내용은 Amazon MWAA 환경에 대한 액세스 관리 단원을 참조하십시오.

내용

고객

Amazon MWAA에서 수행하는 작업에 따라 사용 방법 AWS Identity and Access Management (IAM) 이 다릅니다.

서비스 사용자 – Amazon MWAAS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한은 관리자가 제공합니다. 작업 수행을 위해 더 많은 Amazon MWAA 기능을 이용한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon MWAA의 기능에 액세스할 수 없다면 Amazon Managed Workflows for Apache Airflow의 자격 증명 및 액세스 문제 해결 단원을 참조하십시오.

서비스 관리자 – 사용자가 회사에서 Amazon MWAAS 리소스를 관리하고 있다면 Amazon MWAA에 대한 완전한 액세스 권한을 갖게 됩니다. 서비스 관리자의 직무는 서비스 사용자가 액세스해야 하는 Amazon MWAA 기능과 리소스를 결정하는 것입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다.

회사가 Amazon MWAA에서 IAM을 사용하는 방법에 관한 자세한 내용은 Amazon MWAA에서 IAM을 사용하는 방법 단원을 참조하십시오.

IAM 관리자 - IAM 관리자라면 Amazon MWAA에 대한 액세스 관리 정책 작성 방법을 자세히 알아두는 것이 좋습니다. IAM에서 사용할 수 있는 Amazon MWAA 자격 증명 기반 정책의 예를 보려면 Amazon MWAA 자격 증명 기반 정책 예제 단원을 참조하십시오.

자격 증명을 통한 인증

인증은 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연동 자격 증명으로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 가진 사용자 내의 자격 증명입니다. AWS 계정 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입

니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용하여 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자

또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

자격 증명 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 유형

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

사용자가 자체 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 AWS CLI 권한이 포함됩니다. 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": "*"
        }
    ]
}
```

Amazon Managed Workflows for Apache Airflow의 자격 증명 및 액세스 문제 해결

다음 정보를 이용하여 Amazon MWAA 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

본인은 Amazon SNS에서 작업을 수행할 권한이 없습니다.

작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 도움을 요청해야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon MWAA에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 AWS 서비스 수 있는 기능도 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음에 예로 보인 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amazon MWAA에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사용자가 내 Amazon MWAA 리소스에 액세스할 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- Amazon MWAA에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon MWAA에서 IAM을 사용하는 방법 단원을 참조하십시오.
- 소유하고 AWS 계정 있는 모든 리소스에 대한 액세스 권한을 AWS 계정 부여하는 방법을 알아보려면 IAM 사용 설명서의 다른 IAM 사용자에게 액세스 권한 제공을 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

Amazon MWAA에서 IAM을 사용하는 방법

Amazon MWAA는 IAM 자격 증명 기반 정책을 사용하여 Amazon MWAA 작업 및 리소스에 권한을 부여합니다. Amazon MWAA 리소스에 대한 액세스를 제어하는 데 사용할 수 있는 사용자 지정 IAM 정책에 대해 권장하는 예제는 the section called “Amazon MWAA 환경 액세스” 단원을 참조하십시오.

Amazon MWAA 및 기타 AWS 서비스가 IAM과 어떻게 연동되는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 연동되는AWS 서비스를 참조하십시오.

Amazon MWAA 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스는 물론, 작업이 허용되거나 거부되는 조건도 지정할 수 있습니다. Amazon MWAA는 특정 작업, 리소스 및 조건 키를 지원합니다.

다음 단계는 IAM 콘솔을 사용하여 새 JSON 정책을 생성하는 방법을 보여줍니다. 이 정책은 Amazon MWAA 리소스에 대한 읽기 전용 액세스를 허용합니다.

JSON 정책 편집기를 사용하여 정책을 생성하려면

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.

   정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작하기를 선택합니다.
3. 페이지 상단에서 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.

5. 다음 JSON 정책 문서를 입력합니다.

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

6. 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션 간에 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 정책 재구성을 참조하십시오.

7. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
8. 정책 생성을 선택하고 새로운 정책을 저장합니다.

JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는

권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

정책 설명에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. `Action` 요소는 정책에서 허용되는 작업을 나열합니다. `NotAction` 요소는 허용되지 않는 작업을 나열합니다.

Amazon MWAA에 정의된 작업은 Amazon MWAA를 사용하여 수행할 수 있는 작업을 반영합니다. Detective의 정책 작업에는 다음과 같은 접두사가 붙습니다: `airflow:`.

와일드카드(*)를 사용하여 여러 작업을 지정할 수도 있습니다. 이러한 작업을 별도로 나열하는 대신, 해당 단어로 끝나는 모든 작업에 대한 액세스 권한을 부여할 수 있습니다(예: `environment`).

Amazon MWAA 작업의 목록을 보려면 IAM 사용 설명서의 Amazon Managed Workflows for Apache Airflow에서 정의한 작업을 참조하십시오.

Amazon MWAA 자격 증명 기반 정책 예제

Amazon MWAA 정책을 보려면 Amazon MWAA 환경에 대한 액세스 관리 단원을 참조하십시오.

기본 설정으로 IAM 사용자 및 역할은 Amazon MWAA 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console AWS CLI, 또는 AWS API를 사용하여 작업을 수행할 수 없습니다.

IAM 관리자는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음, 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결합니다.

> **Important**
>
> Amazon MWAA 리소스에 대한 액세스를 제공하려면 IAM 역할 및 임시 자격 증명을 사용하는 것이 좋습니다. IAM 사용자에게 권한 정책을 직접 연결하지 않도록 하십시오.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하십시오.

주제

- 정책 모범 사례
- Amazon MWAA 콘솔 사용
- 사용자가 자체 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon MWAA 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- AWS 관리형 정책으로 시작하여 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하십시오.

Amazon MWAA 콘솔 사용

Amazon MWAA 콘솔을 사용하려면 사용자 또는 역할이 API의 해당 작업과 일치하는 관련 작업에 액세스할 수 있어야 합니다.

Amazon MWAA 정책을 보려면 Amazon MWAA 환경에 대한 액세스 관리 단원을 참조하십시오.

사용자가 자체 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다. AWS CLI 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": "*"
        }
    ]
```

```
}
```

Amazon Managed Workflows for Apache Airflow의 규정 준수 확인

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의AWS 서비스 범위별, 규정 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

  > **Note**
  >
  > 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.

- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

Amazon Managed Workflows for Apache Airflow의 복원성

AWS 글로벌 인프라는 지역 및 가용 영역을 중심으로 AWS 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

Amazon MWAA의 인프라 보안

Apache Airflow용 Amazon 관리형 워크플로는 관리형 서비스로서 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon MWAA에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

Amazon MWAA의 구성 및 취약성 분석

구성 및 IT 제어는 귀하와 당사 고객 간의 AWS 공동 책임입니다.

Amazon Managed Workflows for Apache Airflow는 사용자 환경에 Apache Airflow를 정기적으로 패치하고 업그레이드합니다. 사용자의 VPC에 적절한 액세스 정책이 사용되는지 확인해야 합니다.

자세한 내용은 다음 리소스를 참조하십시오.

- Amazon Managed Workflows for Apache Airflow의 규정 준수 확인
- 공동 책임 모델
- Web Services: 보안 프로세스의 개요
- Amazon MWAA의 인프라 보안
- Amazon MWAA의 보안 모범 사례

Amazon MWAA의 보안 모범 사례

Amazon MWAA는 사용자가 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 사용자의 환경에 적절하지 않거나 충분하지 않을 수 있으므로 규정이 아닌 참고용으로만 사용하십시오.

- 최소 허용 권한 정책을 사용합니다. 사용자가 작업을 수행하는 데 필요한 리소스 또는 작업에만 권한을 부여합니다.
- 계정의 사용자 활동을 모니터링하는 AWS CloudTrail 데 사용합니다.
- Amazon S3 버킷 정책 및 객체 ACL에서 관련 Amazon MWAA 환경의 사용자에게 객체를 버킷에 넣을 수 있는 권한을 부여하도록 해야 합니다. 이렇게 하면 버킷에 워크플로를 추가할 권한이 있는 사용자가 Airflow에서 워크플로를 실행할 권한도 갖게 됩니다.
- Amazon MWAA 환경과 연관된 Amazon S3 버킷을 사용합니다. Amazon S3 버킷은 어떤 이름이든 사용할 수 있습니다. 버킷에 다른 객체를 저장하거나 다른 서비스와 함께 버킷을 사용하지 마십시오.

Apache Airflow의 보안 모범 사례

Apache Airflow는 멀티테넌트가 아닙니다. Amazon MWAA에서 구현하고 있는 것처럼 일부 기능을 특정 사용자로 제한하는 액세스 제어 조치가 있지만, DAG 생성자는 Apache Airflow 사용자의 권한을 변경하고 기본 메타데이터베이스와 상호 작용할 수 있는 DAG를 작성할 수 능력을 갖고 있습니다.

Amazon MWAA에서 Apache Airflow를 사용할 때는 환경의 메타데이터베이스와 DAG의 안전을 확보하기 위해 다음 단계를 수행하는 것이 좋습니다.

- DAG 쓰기 권한이 있거나 Amazon S3 `/dags` 폴더에 파일을 추가할 수 있는 기능이 있는 별도의 팀에는 각각 별도의 환경을 사용합니다. 단, Amazon MWAA 실행 역할 또는 Apache Airflow 연결을 통해 액세스할 수 있는 모든 항목에는 해당 환경에 쓸 수 있는 사용자도 액세스할 수 있음을 예상해야 합니다.
- Amazon S3 DAG 폴더에 직접 액세스할 수 있는 권한을 제공하지 마십시오. 대신, Amazon S3에 DAG를 작성할 때는 CI/CD 도구를 사용하되, DAG 코드가 팀의 보안 지침을 충족하는지 확인하는 검증 단계와 함께 사용합니다.
- 환경의 Amazon S3 버킷에 대한 사용자 액세스를 차단합니다. 대신, DAG를 저장하는 Amazon MWAA Amazon S3 버킷과는 별도의 위치에 저장된 YAML, JSON 또는 기타 정의 파일을 기반으로 DAG를 생성하는 DAG 팩토리를 사용합니다.
- Secrets Manager에 암호를 저장합니다. 이렇게 해도 DAG를 작성할 수 있는 사용자가 암호를 읽는 것을 막을 수는 없지만, 사용자 환경에서 사용하는 암호를 수정하는 것은 막을 수 있습니다.

Apache Airflow 사용자 권한 변경 감지

CloudWatch 로그 인사이트를 사용하여 Apache Airflow 사용자 권한을 변경하는 DAG의 발생을 감지할 수 있습니다. 이를 위해 EventBridge 스케줄링된 규칙, Lambda 함수 CloudWatch 및 Logs Insights를 사용하여 DAG 중 하나가 Apache Airflow 사용자 권한을 변경할 때마다 CloudWatch 메트릭에 알림을 전송할 수 있습니다.

필수 조건

아래 단계를 완료하려면 다음 사항이 필요합니다.

- 모든 Apache Airflow 로그 유형이 `INFO` 로그 수준에서 활성화된 Amazon MWAA 환경 자세한 내용은 the section called “Airflow 로그 확인” 단원을 참조하십시오.

Apache Airflow 사용자 권한 변경에 대한 알림을 구성하려면

1. 5개의 Amazon MWAA 환경 로그 그룹 `DAGProcessing` (`Scheduler`,,,,`Task`) 에 대해 CloudWatch 다음 로그 인사이트 쿼리 문자열을 실행하는 Lambda 함수를 생성합니다. `WebServer` `Worker`

```
fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
 by @log
```

2. 이전 단계에서 생성한 Lambda 함수를 EventBridge 규칙의 대상으로 사용하여 일정에 따라 실행되는 규칙을 생성합니다. Cron 또는 Rate 표현식을 이용해서 일정을 구성하여 정기적으로 실행합니다.

Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전

이 페이지에서는 Amazon Managed Workflows for Apache Airflow가 지원하는 Apache Airflow 버전과 최신 버전으로 업그레이드하기 위해 권장하는 전략을 설명합니다.

주제

Amazon MWAA 버전 정보

Amazon MWAA는 Apache Airflow 릴리스를 다른 일반 바이너리 및 Python 라이브러리와 함께 번들로 제공하는 컨테이너 이미지를 구축합니다. 이미지는 지정한 버전에 대한 Apache Airflow 기본 설치를 사용합니다. 환경을 생성할 때 사용할 이미지 버전을 지정합니다. 환경이 생성되면 이후 버전으로 업그레이드할 때까지 지정된 이미지 버전을 계속 사용합니다.

최신 버전

Amazon MWAA는 둘 이상의 Apache Airflow 버전을 지원합니다. 환경을 생성할 때 이미지 버전을 지정하지 않는 경우 Amazon MWAA는 지원되는 최신 버전의 Apache Airflow를 사용하여 환경을 생성합니다.

Apache Airflow 버전

다음 Apache Airflow 버전은 Amazon Managed Workflows for Apache Airflow에서 지원됩니다.

Note

- Apache Airflow v2.2.2부터 Amazon MWAA는 Python 요구 사항, 공급자 패키지 및 사용자 지정 플러그인을 Apache Airflow 웹 서버에 직접 설치할 수 있도록 지원합니다.
- Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.

  요구 사항 파일에서 제약 조건을 설정하는 방법에 대한 자세한 내용은 Python 종속성 설치를 참조하십시오.

Apache Airflow 버전	Apache Airflow 가이드	Apache Airflow 제약 조건	Python 버전
v2.8.1	아파치 에어플로우 v2.8.1 참조 가이드	아파치 에어플로우 v2.8.1 제약 파일	Python 3.11
v2.7.2	Apache Airflow v2.7.2 참조 가이드	Apache Airflow v2.7.2 제약 조건 파일	Python 3.11
v2.6.3	Apache Airflow v2.6.3 참조 가이드	Apache Airflow v2.6.3 제약 조건 파일	Python 3.10
v2.5.1	Apache Airflow v2.5.1 참조 가이드	Apache Airflow v2.5.1 제약 조건 파일	Python 3.10
v2.4.3	Apache Airflow v2.4.3 참조 가이드	Apache Airflow v2.4.3 제약 조건 파일	Python 3.10
v2.2.2	Apache Airflow v2.2.2 참조 가이드	Apache Airflow v2.2.2 제약 조건 파일	Python 3.7
v2.0.2	Apache Airflow v2.0.2 참조 가이드	Apache Airflow v2.0.2 제약 조건 파일	Python 3.7

메타데이터 데이터베이스 백업 지침을 포함하여 자체 관리형 Apache Airflow 배포 마이그레이션 또는 기존 Amazon MWAA 환경 마이그레이션에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드를 참조하십시오.

Apache Airflow 구성 요소

이 섹션에서는 Amazon MWAA의 각 Apache Airflow 버전에 사용할 수 있는 Apache Airflow 스케줄러 및 작업자 수를 설명하고, 각 기능을 지원하는 버전을 나타내는 주요 Apache Airflow 기능 목록을 제공합니다.

스케줄러

Apache Airflow 버전	스케줄러(기본값)	스케줄러(최소)	스케줄러(최대)	
Apache Airflow v2 이상	2	2	5	

작업자

Airflow 버전	작업자(최소)	작업자(최대)	작업자(기본값)	
Apache Airflow v2	1	25	10	

Apache Airflow 버전 업그레이드

Amazon MWAA는 마이너 버전 업그레이드를 지원합니다. 즉, 환경을 x.**1**.z 버전에서 x.**2**.z 버전으로 업그레이드할 수 있지만 새 메이저 버전으로 업그레이드할 수는 없습니다(예: **1**.y.z에서 **2**.y.z로).

> **Note**
>
> 사용자 환경에 맞게 Apache Airflow 버전을 다운그레이드할 수 없습니다.

워크플로우 리소스 업데이트 및 환경을 새 버전으로 업그레이드하는 방법에 대한 자세한 내용 및 자세한 지침은 the section called “버전 업그레이드” 단원을 참조하십시오.

Apache Airflow 지원 중단 버전

다음 표에는 Amazon MWAA에서 지원되지 않는 Apache Airflow 및 각 버전의 초기 릴리스 및 지원 종료 날짜가 나와 있습니다. 새 버전으로 마이그레이션하는 방법에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드를 참조하십시오.

Apache Airflow 버전	Apache Airflow 릴리스 날짜	Amazon MWAA 출시 날짜	Amazon MWAA 제한된 지원 날짜	Amazon MWAA 지원 종료 날짜
v1.10.12	2020년 8월 25일	2020년 11월 24일	2023년 8월 21일	2024년 2월 21일
v2.0.2	2021년 4월 19일	2021년 5월 25일	2023년 11월 23일	2024년 4월 29일
v2.2.2	2021년 11월 15일	2022년 1월 27일	2024년 1월 25일	2024년 6월 27일

Apache Airflow 버전 지원 및 FAQ

Apache Airflow 커뮤니티 릴리스 프로세스 및 버전 정책에 따라 Amazon MWAA는 언제든지 최소 3개 이상의 Apache Airflow 마이너 버전을 지원하기 위해 최선을 다하고 있습니다. 지정된 Apache Airway 마이너 버전의 지원 종료일은 지원 종료일 최소 90일 전에 공지합니다.

자주 묻는 질문(FAQ)

Q: Amazon MWAA는 Apache Airflow 버전을 얼마 동안 지원합니까?

A: Amazon MWAA는 처음 출시된 후 최소 12개월 동안 Apache Airflow 마이너 버전을 지원합니다.

Q: Amazon MWAA의 Apache Airflow 버전에 대한 지원이 종료되면 알림을 받을 수 있나요?

A: 예. 계정 내 Amazon MWAA 환경에서 지원 종료일이 임박한 버전을 실행하는 경우 Amazon MWAA는 지원 종료일과 AWS Health Dashboard 함께 알림을 보냅니다.

Q: 제한된 지원 날짜에는 어떻게 됩니까?

A: 제한된 지원 날짜에는 더 이상 관련 버전으로 새로운 Amazon MWAA 환경을 생성할 수 없습니다. 기존 환경은 지원 종료 날짜까지 계속 사용할 수 있습니다.

Q: 지원 종료 날짜에는 어떻게 됩니까?

A: 지원 종료일이 되어도 더 이상 사용되지 않는 Apache Airflow 버전을 실행하는 기존 Amazon MWAA 환경에 계속 액세스할 수 있게 되는 데 따른 위험은 사용자가 감수해야 합니다. Amazon MWAA에서 최신 버전의 Apache Airflow로 업그레이드하는 방법에 대한 지침은 Amazon MWAA 마이그레이션 가이드를 참조하십시오.

Important

Amazon MWAA 버전을 최신 상태로 유지할 책임은 귀하에게 있습니다. AWS 최신 보안, 개인 정보 보호 및 가용성 보호 조치의 혜택을 누리려면 모든 고객에게 Amazon MWAA 환경을 최신 버전으로 업그레이드할 것을 촉구합니다. 지원 중단일이 지난 소프트웨어 (레거시 버전이라고 함) 가 지원되지 않는 버전이나 소프트웨어로 환경을 운영하는 경우 다운타임 이벤트를 비롯한 보안, 개인 정보 보호 및 운영 위험에 직면할 가능성이 커집니다. Amazon MWAA 환경을 레거시 버전에서 운영함으로써 귀하는 이러한 위험을 이해하고 인지하고 있음을 확인하고 가능한 한 빨리 최신 버전으로 업그레이드를 완료하는 데 동의하는 것으로 간주됩니다. 기존 버전에서 환경을 계속 운영하려면 서비스 사용에 적용되는 계약이 적용됩니다. AWS
레거시 버전은 일반적으로 사용할 수 있는 것으로 간주되지 AWS 않으며 더 이상 레거시 버전을 지원하지 않습니다. 따라서 레거시 버전이 서비스 AWS, AWS 계열사 또는 기타 제3자에 대한 보안 또는 책임 위험 또는 위해를 초래할 위험이 있다고 AWS 판단되는 경우 언제든지 레거시 버전에 대한 액세스 또는 사용을 제한할 수 있습니다. 기존 버전에서 워크로드를 계속 실행하기로 결정하면 콘텐츠가 사용 불가능하거나 손상되거나 복구가 불가능해질 수 있습니다. 레거시 버전에서 실행되는 환경에는 서비스 수준 계약 (SLA) 예외가 적용됩니다.
레거시 버전에서 실행되는 환경 및 관련 소프트웨어에는 버그, 오류, 결함 및 유해한 구성 요소가 포함될 수 있습니다. 따라서 계약 또는 서비스 약관에 상반되는 내용이 있더라도 레거시 버전은 있는 그대로 AWS 제공됩니다.
AWS의 공동 책임 모델에 대한 자세한 내용은 AWS Well-Architected 프레임워크의 공동 책임을 참조하십시오.

Amazon Managed Workflow for Apache Airflow 서비스 엔드포인트 및 할당량

Amazon Managed Workflow for Apache Airflow에는 다음과 같은 서비스 할당량 및 엔드포인트가 있습니다. 서비스 할당량 (한도라고도 함) 은 계정에 AWS 대한 서비스 리소스 또는 운영의 최대 수입니다.

목차

Service 엔드포인트

Amazon MWAA의 엔드포인트 목록을 보려면 Amazon Managed Workflows for Apache Airflow 엔드포인트 및 할당량을 참조하십시오.

Service quotas

할당량 이름	설명	기본 할당량	조정 가능
환경	리전별로 계정당 Amazon MWAA 환경의 최대 수	10	예
환경별 작업자	Amazon MWAA 환경당 최대 작업자 수입니다.	25	예
환경별 웹 서버	Amazon MWAA 환경당 최대 웹 서버 수입니다.	5	예

할당량 증가

할당량 증가 요청을 제출하여 조정 가능한 할당량 증가를 요청할 수 있습니다.

Amazon MWAA 자주 묻는 질문

이 페이지에서는 Amazon Managed Workflow for Airflow를 사용할 때 발생할 수 있는 일반적인 질문에 대해 설명합니다.

목차

- Amazon MWAA가 새 DAG 파일을 인식하는 데 시간이 얼마나 걸립니까?
- Apache Airflow에서 내 DAG 파일을 선택하지 않는 이유는 무엇입니까?
- 환경에서 plugins.zip 또는 requirements.txt를 삭제할 수 있습니까?
- Apache Airflow v2.0.2 관리자 플러그인 메뉴에 내 플러그인이 보이지 않는 이유는 무엇입니까?
- DMS (AWS 데이터베이스 마이그레이션 서비스) 운영자를 사용할 수 있습니까?

지원되는 버전

Amazon MWAA는 Apache Airflow v2에 대해 무엇을 지원합니까?

Amazon MWAA가 무엇을 지원하는지 알아보려면 Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전을 참조하십시오.

이전 버전의 Apache Airflow가 지원되지 않는 이유는 무엇입니까?

이전 버전에 대한 보안 문제로 인해 최신(출시 기준) Apache Airflow 버전 Apache Airflow v1.10.12만 지원하고 있습니다.

어떤 Python 버전을 사용해야 합니까?

다음 Apache Airflow 버전은 Amazon Managed Workflows for Apache Airflow에서 지원됩니다.

> **Note**
>
> - Apache Airflow v2.2.2부터 Amazon MWAA는 Python 요구 사항, 공급자 패키지 및 사용자 지정 플러그인을 Apache Airflow 웹 서버에 직접 설치할 수 있도록 지원합니다.
> - Apache Airflow v2.7.2부터 요구 사항 파일에 `--constraint` 문이 포함되어야 합니다. 제약 조건을 제공하지 않으면 Amazon MWAA에서 요구 사항에 나열된 패키지가 사용 중인 Apache Airway 버전과 호환되도록 제약 조건을 지정합니다.
>
>   요구 사항 파일에서 제약 조건을 설정하는 방법에 대한 자세한 내용은 Python 종속성 설치를 참조하십시오.

Apache Airflow 버전	Apache Airflow 가이드	Apache Airflow 제약 조건	Python 버전
v2.8.1	아파치 에어플로우 v2.8.1 레퍼런스 가이드	아파치 에어플로우 v2.8.1 제약 파일	Python 3.11
v2.7.2	Apache Airflow v2.7.2 참조 가이드	Apache Airflow v2.7.2 제약 조건 파일	Python 3.11
v2.6.3	Apache Airflow v2.6.3 참조 가이드	Apache Airflow v2.6.3 제약 조건 파일	Python 3.10
v2.5.1	Apache Airflow v2.5.1 참조 가이드	Apache Airflow v2.5.1 제약 조건 파일	Python 3.10
v2.4.3	Apache Airflow v2.4.3 참조 가이드	Apache Airflow v2.4.3 제약 조건 파일	Python 3.10
v2.2.2	Apache Airflow v2.2.2 참조 가이드	Apache Airflow v2.2.2 제약 조건 파일	Python 3.7
v2.0.2	Apache Airflow v2.0.2 참조 가이드	Apache Airflow v2.0.2 제약 조건 파일	Python 3.7

메타데이터 데이터베이스 백업 지침을 포함하여 자체 관리형 Apache Airflow 배포 마이그레이션 또는 기존 Amazon MWAA 환경 마이그레이션에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드를 참조하십시오.

Amazon MWAA에서는 어떤 버전의 `pip`을 사용합니까?

Apache Airflow v1.10.12를 실행하는 환경의 경우 Amazon MWAA는 `pip` 버전 21.1.2를 설치합니다.

> **Note**
>
> Amazon MWAA는 Apache Airflow v1.10.12 환경용으로 `pip`을 업그레이드하지 않습니다.

Apache Airflow v2 이상을 실행하는 환경의 경우 Amazon MWAA는 `pip` 버전 21.3.1을 설치합니다.

사용 사례

vs는 언제 사용해야 하나요? AWS Step Functions 아마존 MWAA?

1. Step Functions이 단일 주문 또는 백만 건의 주문에 대한 수요를 충족하도록 확장할 수 있으므로 Step Functions를 사용하여 개별 고객 주문을 처리할 수 있습니다.
2. 전날 주문을 처리하는 야간 워크플로우를 실행하는 경우 Step Functions나 Amazon MWAA를 사용할 수 있습니다. Amazon MWAA를 사용하면 사용 중인 AWS 리소스에서 워크플로를 추상화하는 오픈 소스 옵션을 사용할 수 있습니다.

환경 사양

각 환경에 대해 사용할 수 있는 작업 스토리지는 얼마나 됩니까?

작업 스토리지는 10GB로 제한되며, Amazon ECS Fargate 1.3에 의해 지정됩니다. RAM 용량은 사용자가 지정한 환경 클래스에 따라 결정됩니다. 환경 클래스에 대한 자세한 내용은 Amazon MWAA 환경 클래스 구성 섹션을 참조하십시오.

Amazon MWAA 환경에 사용되는 기본 운영 체제는 무엇입니까?

Amazon MWAA 환경은 Amazon Linux AMI를 실행하는 인스턴스에서 생성됩니다.

Amazon MWAA 환경에 사용자 지정 이미지를 사용할 수 있습니까?

사용자 지정 이미지는 지원되지 않습니다. Amazon MWAA는 Amazon 리눅스 AMI를 기반으로 구축된 이미지를 사용합니다. Amazon MWAA는 환경용 Amazon S3 버킷에 추가한 requirements.txt 파일에 지정된 요구 사항에 `pip3 -r install`을 실행하여 추가 요구 사항을 설치합니다.

Amazon MWAA HIPAA를 준수합니까?

Amazon MWAA는 미국 건강 보험 양도 및 책임에 관한 법(HIPAA)을 충족합니다. HIPAA 비즈니스 제휴 부록 (BAA) 이 있는 AWS 경우, 2022년 11월 14일 또는 그 이후에 생성된 환경에서 보호 건강 정보 (PHI) 를 처리하는 워크플로에 Amazon MWAA를 사용할 수 있습니다.

Amazon MWAA가 스팟 인스턴스를 지원합니까?

Amazon MWAA는 현재 Apache Airflow에 대한 온디맨드 Amazon EC2 스팟 인스턴스 유형을 지원하지 않습니다. 그러나, Amazon MWAA 환경은 Amazon EMR 및 Amazon EC2 등에 대한 스팟 인스턴스를 트리거할 수 있습니다.

Amazon MWAA가 사용자 지정 도메인을 지원합니까?

Amazon MWAA 호스트 이름에 사용자 지정 도메인을 사용할 수 있으려면 다음 중 하나를 수행합니다.

- 퍼블릭 웹 서버 액세스가 가능한 Amazon MWAA 배포의 경우, Amazon과 CloudFront Lambda @Edge 를 사용하여 트래픽을 사용자 환경으로 보내고 사용자 지정 도메인 이름을 매핑할 수 있습니다. CloudFront 공개 환경을 위한 사용자 지정 도메인을 설정하는 예와 자세한 내용은 Amazon MWAA 예제 리포지토리의 퍼블릭 웹 서버용 Amazon MWAA 사용자 지정 도메인 샘플을 참조하십시오. GitHub
- 프라이빗 웹 서버에 액세스를 사용한 Amazon MWAA 배포의 경우, Application Load Balancer(ALB)를 사용하여 트래픽을 Amazon MWAA로 보내고 사용자 지정 도메인 이름을 ALB에 매핑할 수 있습니다. 자세한 설명은 the section called “로드 밸런서 사용(고급)” 섹션을 참조하세요.

SSH를 내 환경에 연결할 수 있습니까?

Amazon MWAA 환경에서는 SSH가 지원되지 않지만 `BashOperator`을 사용한 bash 명령을 실행하는 데 DAG를 사용할 수 있습니다. 예:

```
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'] }}"
      )
```

Apache Airflow UI에서 DAG를 트리거하려면 다음을 사용합니다.

```
{ "command" : "your bash command"}
```

VPC 보안 그룹에 자기 참조 규칙이 필요한 이유는 무엇입니까?

자기 참조 규칙을 생성하여 VPC에서 동일한 보안 그룹으로 소스를 제한하고 모든 네트워크로 공개되지 않도록 합니다. 자세한 내용은 the section called “VPC 보안” 단원을 참조하십시오.

IAM에서 여러 그룹의 환경을 숨길 수 있습니까?

에서 환경 이름을 지정하여 액세스를 제한할 수 있지만 AWS 콘솔에서는 AWS Identity and Access Management 가시성 필터링을 사용할 수 없습니다. 즉, 사용자가 하나의 환경을 볼 수 있으면 모든 환경을 볼 수 있기 때문입니다.

Apache Airflow 작업자에 임시 데이터를 저장할 수 있습니까?

Apache Airflow 운영자는 작업자에 임시 데이터를 저장할 수 있습니다. Apache Airflow 작업자는 사용자 환경의 Fargate 컨테이너의 `/tmp`에 있는 임시 파일에 액세스할 수 있습니다.

> **Note**
>
> Amazon ECS Fargate 1.3에 따르면 총 작업 스토리지는 10GB로 제한됩니다. 후속 작업이 다른 `/tmp` 폴더를 사용할 수 있는 동일한 Fargate 컨테이너 인스턴스에서 실행된다는 보장은 없습니다.

25명 이상의 Apache Airflow 작업자를 지정할 수 있습니까?

예. Amazon MWAA 콘솔에서는 Apache Airflow 작업자를 25명까지 지정할 수 있지만 할당량 증가를 요청하여 한 환경에 50명까지 구성할 수 있습니다. 자세한 내용은 할당량 증가 요청을 참조하십시오.

Amazon MWAA가 공유 Amazon VPC 또는 공유 서브넷을 지원합니까?

Amazon MWAA는 공유 Amazon VPC 또는 공유 서브넷을 지원하지 않습니다. 환경을 생성할 때 선택한 Amazon VPC는 환경을 생성하려는 계정이 소유해야 합니다. 하지만, Amazon MWAA 계정의 Amazon VPC에서 들어오는 트래픽을 공유 VPC로 라우팅할 수 있습니다. 자세한 내용과 공유 Amazon VPC로 트래픽을 라우팅하는 예제를 보려면 Amazon VPC 전송 게이트웨이 가이드의 인터넷으로 중앙 집중식 아웃바운드 라우팅을 참조하십시오.

지표

작업자 규모 조정 여부를 결정하는 데 어떤 지표가 사용됩니까?

Amazon MWAA는 And In을 QueuedTasks CloudWatch 모니터링하여 RunningTasks사용자 환경에서 Apache Airflow Worker를 확장할지 여부를 결정합니다. 자세한 내용은 모니터링 및 지표을 참조하세요.

에서 사용자 지정 지표를 생성할 수 있습니까? CloudWatch

CloudWatch 콘솔에서는 사용할 수 없습니다. 하지만 사용자 지정 지표를 기록하는 DAG를 만들 수 있습니다. CloudWatch 자세한 설명은 the section called “DAG를 사용하여 사용자 지정 지표 작성” 섹션을 참조하세요.

DAG, 운영자, 연결 및 기타 질문

`PythonVirtualenvOperator`를 사용할 수 있습니까?

`PythonVirtualenvOperator`는 Amazon MWAA에서 명시적으로 지원되지 않지만 `PythonVirtualenvOperator`를 사용하는 사용자 지정 플러그인을 생성할 수 있습니다. 샘플 코드에 대한 내용은 the section called “PythonVirtualenvOperator를 패치하는 사용자 지정 플러그인 ” 단원을 참조하십시오.

Amazon MWAA가 새 DAG 파일을 인식하는 데 시간이 얼마나 걸립니까?

DAG는 Amazon S3 버킷에서 사용자 환경으로 주기적으로 동기화됩니다. 새 DAG 파일을 추가하는 경우 Amazon MWAA가 새 파일을 사용하기 시작하는 데 약 300초가 걸립니다. 기존 DAG를 업데이트하는 경우 Amazon MWAA가 업데이트를 인식하는 데 약 30초가 걸립니다.

이러한 값(새 DAG의 경우 300초, 기존 DAG의 업데이트의 경우 30초)은 Apache Airflow 구성 옵션 `dag_dir_list_interval` 및 `min_file_process_interval`에 각각 해당합니다.

Apache Airflow에서 내 DAG 파일을 선택하지 않는 이유는 무엇입니까?

이 문제에 대한 가능한 해결 방법은 다음과 같습니다.

1. 실행 역할에 Amazon S3 버킷에 대한 충분한 권한이 있는지 확인합니다. 자세한 내용은 Amazon MWAA 실행 역할 단원을 참조하십시오.

2. Amazon S3 버킷에 퍼블릭 액세스 차단이 구성되어 있고 버전 관리가 활성화되어 있는지 확인합니다. 자세한 내용은 Amazon MWAA용 Amazon S3 버킷 생성 단원을 참조하십시오.
3. DAG 파일 자체를 확인합니다. 예를 들어, 각 DAG에 고유한 DAG ID가 있어야 합니다.

환경에서 `plugins.zip` 또는 `requirements.txt`를 삭제할 수 있습니까?

현재는 plugins.zip 또는 requirements.txt를 추가한 후 해당 환경에서 삭제할 수 있는 방법은 없지만, 현재 해결 중입니다. 그 동안 해결 방법은 빈 텍스트 또는 zip 파일을 각각 가리키는 것입니다. 자세한 내용은 Amazon S3에서 파일 삭제 단원을 참조하십시오.

Apache Airflow v2.0.2 관리자 플러그인 메뉴에 내 플러그인이 보이지 않는 이유는 무엇입니까?

보안상의 이유로 Amazon MWAA의 Apache Airflow 웹 서버는 네트워크 송신이 제한되어 있으며 버전 2.0.2 환경용 Apache Airflow 웹 서버에 직접 플러그인이나 Python 종속성을 설치하지 않습니다. 표시된 플러그인을 사용하면 Amazon MWAA가 (IAM) 에서 아파치 에어플로우 사용자를 인증할 수 있습니다. AWS Identity and Access Management

플러그인과 Python 종속성을 웹 서버에 직접 설치하려면 Apache Airflow v2.2 이상을 사용하여 새 환경을 생성하는 것이 좋습니다. Amazon MWAA는 Apache Airflow v2.2 이상의 웹 서버에 직접 Python 종속성 및 사용자 지정 플러그인을 설치합니다.

DMS (AWS 데이터베이스 마이그레이션 서비스) 운영자를 사용할 수 있습니까?

Amazon MWAA는 DMS 운영자를 지원합니다. 하지만 이 운영자는 Amazon MWAA 환경과 연결된 Amazon Aurora PostgreSQL 메타데이터 데이터베이스에서 작업을 수행할하는 데 사용할 수 없습니다.

Amazon Managed Workflows for Apache Airflow 문제 해결

이 주제는 Amazon Managed Workflows for Apache Airflow에서 Apache Airflow를 사용할 때 발생할 수 있는 일반적인 문제 및 오류와 이러한 오류를 해결하기 위한 권장 단계를 설명합니다.

목차

문제 해결: Apache Airflow v2의 DAG, 연산자, 연결 및 기타 문제

이 페이지의 항목에서는 Amazon Managed Workflows for Apache Airflow 환경에서 발생할 수 있는 Apache Airflow v2 Python 종속성, 사용자 지정 플러그인, DAG, 연산자, 연결, 작업 및 웹 서버 문제에 대한 해결 방법을 설명합니다.

목차

연결

다음 항목에서는 Apache Airflow 연결을 사용하거나 다른 AWS 데이터베이스를 사용할 때 발생할 수 있는 오류에 대해 설명합니다.

Secrets Manager에 연결할 수 없음

다음 단계를 수행하는 것이 좋습니다.

1. the section called “Secrets Manager 구성”에서 Apache Airflow 연결 및 변수를 위한 암호 키를 만드는 방법을 알아봅니다.
2. Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용에서 Apache Airflow 변수(`test-variable`)에 대해 암호 키를 사용하는 방법을 알아봅니다.
3. Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용에서 Apache Airflow 연결(`myconn`)에 대해 암호 키를 사용하는 방법을 알아봅니다.

실행 역할 정책에서 **secretsmanager:ResourceTag/<tag-key>** 암호 관리자 조건 또는 리소스 제한을 구성하려면 어떻게 해야 합니까?

> **Note**
>
> Apache Airflow 버전 2.0 이하에 적용됩니다.

Apache Airflow의 알려진 문제로 인해 현재는 환경 실행 역할의 조건 키 또는 기타 리소스 제한을 사용하여 Secrets Manager 암호에 대한 액세스를 제한할 수 없습니다.

Snowflake에 연결할 수 없음

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. 사용자 환경의 requirements.txt 파일에 다음 항목을 추가합니다.

```
apache-airflow-providers-snowflake==1.3.0
```

3. DAG에 다음 가져오기를 추가합니다.

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

Apache Airflow 연결 객체에 다음 키-값 쌍이 포함되어 있는지 확인합니다.

1. Conn Id: snowflake_conn
2. Conn Type: Snowflake
3. Host: <my account>.<my region if not us-west-2>.snowflakecomputing.com
4. Schema: <my schema>
5. Login: <my user name>
6. Password: ********
7. Port: <port, if any>
8. 추가:

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

예:

```
>>> import json
>>> from airflow.models.connection import Connection
>>> myconn = Connection(
...    conn_id='snowflake_conn',
...    conn_type='Snowflake',
...    host='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...    schema='YOUR_SCHEMA'
...    login='YOUR_USERNAME',
...    password='YOUR_PASSWORD',
...    port='YOUR_PORT'
...    extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Airflow UI에서 내 연결이 보이지 않음

Apache Airflow는 Apache Airflow UI에 연결 템플릿을 제공합니다. 연결 유형에 관계없이 이를 사용하여 연결 URI 문자열을 생성합니다. Apache Airflow UI에서 연결 템플릿을 사용할 수 없는 경우 HTTP 연결 템플릿 사용과 같은 대체 연결 템플릿을 사용하여 연결 URI 문자열을 생성할 수 있습니다.

다음 단계를 수행하는 것이 좋습니다.

1. Amazon MWAA 환경에 설치된 Apache Airflow 공급자 패키지의 Apache Airflow UI에서 Amazon MWAA가 제공하는 연결 유형을 확인합니다.
2. Apache Airflow CLI 명령 참조의 CLI에서 Apache Airflow 연결을 생성하는 명령을 확인합니다.
3. 연결 유형 개요의 Amazon MWAA의 Apache Airflow UI에서 사용할 수 없는 연결 유형에 대해 Apache Airflow UI의 연결 템플릿을 서로 바꿔서 사용하는 방법을 알아봅니다.

웹 서버

다음 항목에서는 Amazon MWAA의 Apache Airflow 웹 서버에서 발생할 수 있는 오류에 대해 설명합니다.

웹 서버에 액세스하는 중에 5xx 오류가 표시됨

다음 단계를 수행하는 것이 좋습니다.

1. Apache Airflow 구성 옵션을 확인합니다. Apache Airflow 구성 옵션으로 지정한 키-값 쌍(예: AWS Secrets Manager)이 올바르게 구성되었는지 확인합니다. 자세한 내용은 the section called "Secrets Manager에 연결할 수 없음" 단원을 참조하십시오.
2. `requirements.txt`를 확인합니다. Airflow "extras" 패키지 및 `requirements.txt`에 나열된 기타 라이브러리가 Apache Airflow 버전과 호환되는지 확인합니다.
3. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

'스케줄러가 실행되지 않는 것 같습니다' 오류가 표시됨

스케줄러가 실행되지 않는 것으로 보이거나 마지막 "heart beat"가 몇 시간 전에 수신된 경우 DAG가 Apache Airflow에 표시되지 않을 수 있으며 새 작업이 예약되지 않습니다.

다음 단계를 수행하는 것이 좋습니다.

1. VPC 보안 그룹이 포트 5432에 대한 인바운드 액세스를 허용하는지 확인합니다. 이 포트는 사용자 환경의 Amazon Aurora PostgreSQL 메타데이터 데이터베이스에 연결하는 데 필요합니다. 이 규칙을 추가한 후 Amazon MWAA에 몇 분 정도 시간을 주면 오류가 사라집니다. 자세한 내용은 the section called "VPC 보안" 단원을 참조하십시오.

   > **Note**
   >
   > - Aurora PostgreSQL 메타데이터베이스는 Amazon MWAA 서비스 아키텍처의 일부이며 AWS 계정에서는 볼 수 없습니다.
   > - 데이터베이스 관련 오류는 일반적으로 스케줄러 오류의 증상이지 근본 원인은 아닙니다.

2. 스케줄러가 실행되지 않는 경우 종속성 설치 실패 또는 스케줄러 과부하와 같은 여러 요인 때문일 수 있습니다. CloudWatch Logs의 해당 로그 그룹을 확인하여 DAG, 플러그인 및 요구 사항이 제대로 작동하는지 확인합니다. 자세한 내용은 모니터링 및 지표 단원을 참조하십시오.

작업

다음 항목에서는 환경에서 Apache Airflow 작업을 수행할 때 발생할 수 있는 오류에 대해 설명합니다.

작업이 중단되거나 완료되지 않은 것으로 보임

Apache Airflow 작업이 "멈춘" 상태이거나 완료되지 않는 경우 다음 단계를 따르는 것이 좋습니다.

1. 정의된 DAG 수가 많을 수 있습니다. DAG 수를 줄이고 환경 업데이트(예: 로그 수준 변경)를 수행하여 강제로 재설정합니다.

    a. Airflow는 DAG의 활성화 여부를 구문 분석합니다. 환경 용량의 50% 이상을 사용하는 경우 Apache Airflow 스케줄러가 너무 많이 사용되기 시작할 수 있습니다. 이로 인해 CloudWatch 지표의 총 구문 분석 시간이 길어지거나 CloudWatch Logs의 DAG 처리 시간이 길어집니다. Apache Airflow 구성을 최적화하는 다른 방법도 있으나 이 설명서에서는 다루지 않습니다.

    b. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called "Apache Airflow 성능 조정" 단원을 참조하십시오.

2. 대기열에 작업 수가 많을 수 있습니다. 이는 보통 "None" 상태의 작업 수가 많이 증가하는 것으로 나타나거나 CloudWatch의 대기 중인 작업 및/또는 보류 중인 작업 수가 많을 때 나타납니다. 이는 다음과 같은 이유로 발생할 수 있습니다.

    a. 환경에 실행할 수 있는 용량보다 실행할 작업이 많거나 자동 크기 조정 전에 대기열에 대기중인 작업이 많을 경우 작업을 감지하고 추가 작업자를 배포할 시간이 있습니다.

    b. 환경에서 실행할 수 있는 용량보다 실행할 작업이 많으면 DAG가 동시에 실행하는 작업 수를 줄이거나 최소 Apache Airflow 작업자를 늘리는 것이 좋습니다.

    c. 자동 크기 조정이 추가 작업자를 감지하고 배포할 시간을 갖기 전에 대기열에 대기 중인 작업이 많은 경우에는 작업 배포에 시차를 두거나 최소 Apache Airflow 작업자를 늘리는 것이 좋습니다.

    d. AWS Command Line Interface(AWS CLI)의 update-environment 명령을 사용하여 사용자 환경에서 실행되는 최소 또는 최대 작업자 수를 변경할 수 있습니다.

    ```
    aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-workers 10
    ```

e. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called “Apache Airflow 성능 조정” 단원을 참조하십시오.

3. 실행 도중에 삭제되는 작업은 Apache Airflow에 더 이상 표시되지 않고 중지되는 작업 로그로 표시될 수 있습니다. 이는 다음과 같은 이유로 발생할 수 있습니다.

   a. 1) 현재 작업이 현재 환경 용량을 초과하고, 2) 몇 분 동안 아무 작업도 실행되지 않거나 대기 중인 상태가 지속되면 3) 새 작업이 대기열에 추가되는 경우 이러한 상태가 발생합니다.

   b. Amazon MWAA 자동 크기 조정은 작업자를 추가하여 첫 번째 시나리오에 대응합니다. 두 번째 시나리오에서는 추가 작업자를 제거합니다. 대기열에 있는 일부 작업의 경우 작업자가 제거되는 과정으로 이어질 수 있으며 컨테이너가 삭제되면 종료됩니다.

   c. 해당 환경의 최소 작업자 수를 늘리는 것이 좋습니다. 또 다른 옵션은 DAG 및 작업의 타이밍을 조정하여 이러한 시나리오가 발생하지 않도록 하는 것입니다.

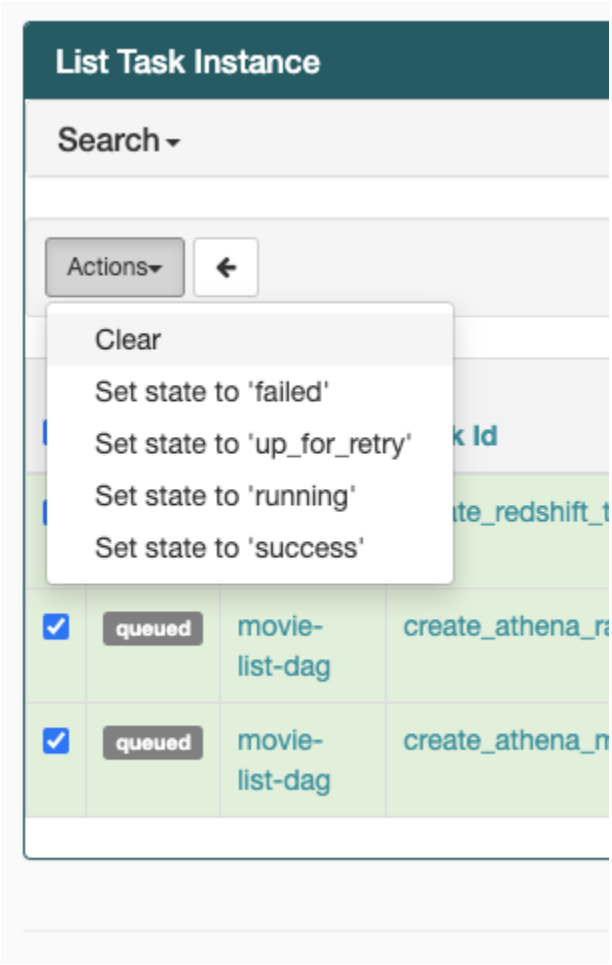
   d. 또한 최소 작업자를 사용자 환경의 최대 작업자와 동일하게 설정하여 자동 크기 조정을 효과적으로 비활성화할 수 있습니다. AWS Command Line Interface(AWS CLI)의 update-environment 명령을 사용하여 최소 및 최대 작업자 수를 동일하게 설정해 자동 크기 조정을 비활성화할 수 있습니다.

   ```
   aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-workers 5
   ```

   e. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called “Apache Airflow 성능 조정” 단원을 참조하십시오.

4. 작업이 “실행 중” 상태에서 멈춘 경우 작업을 지우거나 성공 또는 실패로 표시할 수도 있습니다. 이렇게 하면 사용자 환경의 자동 확장 구성 요소를 통해 환경에서 실행되는 작업자 수를 줄일 수 있습니다. 다음 이미지는 중단된 작업의 예를 보여줍니다.

   - 문제 있는 작업의 원을 선택한 다음 지우기(그림 참조)를 선택합니다. 이렇게 하면 Amazon MWAA에서 작업자의 규모를 축소할 수 있습니다. 그렇지 않으면 Amazon MWAA에서 어떤 DAG가 활성화되었는지 또는 비활성화되었는지 확인할 수 없으며 대기 중인 작업이 있는 경우 규모를 축소할 수 없습니다.



5. Apache Airflow 참조 가이드의 개념에서 Apache Airflow 작업 수명 주기에 대해 자세히 알아봅니다.

CLI

다음 항목에서는 AWS Command Line Interface에서 Airflow CLI 명령을 실행할 때 발생할 수 있는 오류에 대해 설명합니다.

CLI에서 DAG를 트리거할 때 '503' 오류가 표시됨.

Airflow CLI는 동시성이 제한된 Apache Airflow 웹 서버에서 실행됩니다. 일반적으로 최대 4개의 CLI 명령을 동시에 실행할 수 있습니다.

`dags backfill` Apache Airflow CLI 명령이 실패하는 이유는 무엇입니까? 해결 방법이 있나요?

> **Note**
>
> 다음은 Apache Airflow v2.0.2 환경에만 적용됩니다.

다른 Apache Airflow CLI 명령과 마찬가지로 `backfill` 명령은 CLI 작업이 적용되는 DAG에 관계없이 DAG가 처리되기 전에 모든 DAG를 로컬에서 구문 분석합니다. Apache Airflow v2.0.2를 사용하는 Amazon MWAA 환경에서는 CLI 명령이 실행될 때까지 플러그인과 요구 사항이 아직 웹 서버에 설치되지 않았으므로 구문 분석 작업이 실패하고 `backfill` 작업이 호출되지 않습니다. 환경에 요구 사항이나 플러그인이 없으면 `backfill` 작업은 성공할 수 있습니다.

`backfill` CLI 명령을 실행하려면 bash 연산자에서 CLI 명령을 호출하는 것이 좋습니다. bash 연산자에서는 작업자로부터 `backfill`이 시작되므로, 필요한 모든 요구 사항과 플러그인을 사용하여 설치할 수 있으므로 DAG를 성공적으로 구문 분석할 수 있습니다. 다음 예제에서는 `backfill`을 실행하기 위해 `BashOperator`를 사용하여 DAG를 생성하는 방법을 보여줍니다.

```
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"
    )
```

연산자

다음 항목에서는 연산자를 사용할 때 발생할 수 있는 오류에 대해 설명합니다.

S3Transform 연산자를 사용할 때 `PermissionError: [Errno 13] Permission denied` 오류가 발생함

S3Transform 연산자를 사용하여 쉘 스크립트를 실행하려고 하는데 `PermissionError: [Errno 13] Permission denied` 오류가 발생하는 경우 다음 단계를 수행하는 것이 좋습니다. 다음 단계에

서는 기존 plugins.zip 파일이 있다고 가정합니다. 새 plugins.zip 파일을 만들려면 사용자 지정 플러그인 설치 단원을 참조하십시오.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. "transform" 스크립트를 생성합니다.

```
#!/bin/bash
cp $1 $2
```

3. (선택 사항) macOS 및 Linux 사용자는 스크립트가 실행 가능한지 확인하기 위해 다음 명령을 실행해야 할 수도 있습니다.

```
chmod 777 transform_test.sh
```

4. plugins.zip 파일에 스크립트를 추가합니다.

```
zip plugins.zip transform_test.sh
```

5. Amazon S3에 plugins.zip 업로드의 단계를 따릅니다.
6. Amazon MWAA 콘솔에서 plugins.zip 버전 지정의 단계를 따릅니다.
7. 다음 DAG 파일을 생성합니다.

```
from airflow import DAG
from airflow.providers.amazon.aws.operators.s3_file_transform import
 S3FileTransformOperator
from airflow.utils.dates import days_ago
import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    file_transform = S3FileTransformOperator(
        task_id='file_transform',
        transform_script='/usr/local/airflow/plugins/transform_test.sh',
        source_s3_key='s3://YOUR_S3_BUCKET/files/input.txt',
        dest_s3_key='s3://YOUR_S3_BUCKET/files/output.txt'
    )
```

8. Amazon S3에 DAG 코드 업로드의 단계를 따릅니다.

문제 해결: Apache Airflow v1의 DAG, 연산자, 연결 및 기타 문제

이 페이지의 항목에는 Amazon Managed Workflows for Apache Airflow 환경에서 발생할 수 있는 Apache Airflow v1.10.12 Python 종속성, 사용자 지정 플러그인, DAG, 연산자, 연결, 작업 및 웹 서버 문제에 대한 해결 방법이 포함되어 있습니다.

목차

requirements.txt 업데이트

다음 항목에서는 `requirements.txt` 업데이트 시 발생할 수 있는 오류에 대해 설명합니다.

`apache-airflow-providers-amazon` 추가 시 환경이 실패함

`apache-airflow-providers-xyz`은 Apache Airflow v2와만 호환됩니다. `apache-airflow-backport-providers-xyz`은 Apache Airflow 1.10.12와 호환됩니다.

Broken DAG

다음 항목에서는 DAG를 실행할 때 발생할 수 있는 오류에 대해 설명합니다.

Amazon DynamoDB 연산자를 사용할 때 'Broken DAG' 오류가 발생했습니다.

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. `requirements.txt`에 다음 패키지를 추가합니다.

```
boto
```

3. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

'Broken DAG: psycopg2라는 이름의 모듈이 없음' 오류가 발생함

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. Apache Airflow 버전을 사용하여 `requirements.txt`에 다음을 추가합니다. 예:

```
apache-airflow[postgres]==1.10.12
```

3. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

Slack 연산자를 사용할 때 'Broken DAG' 오류가 발생함

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. `requirements.txt`에 다음 패키지를 추가하고 Apache Airflow 버전을 지정합니다. 예:

```
apache-airflow[slack]==1.10.12
```

3. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

Google/GCP/BigQuery를 설치하는 동안 여러 가지 오류가 발생함

Amazon MWAA는 특정 버전의 Cython과 암호화 라이브러리가 필요한 Amazon Linux를 사용합니다. 다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. `requirements.txt`에 다음 패키지를 추가합니다.

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow-backport-providers-amazon[google]
```

3. 백포트 공급자를 사용하지 않는 경우 다음을 사용할 수 있습니다.

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow[gcp]==1.10.12
```

4. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

'Broken DAG: Cython이라는 이름의 모듈이 없음' 오류가 발생함

Amazon MWAA는 특정 버전의 Cython이 필요한 Amazon Linux를 사용합니다. 다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. `requirements.txt`에 다음 패키지를 추가합니다.

```
cython==0.29.21
```

3. Cython 라이브러리에는 다양한 필수 pip 종속성 버전이 있습니다. 예를 들어 `awswrangler==2.4.0`을 사용하려면 `pyarrow<3.1.0,>=2.0.0`이 필요하므로 pip3는 `pyarrow==3.0.0`을 설치하려고 시도하며 이로 인해 Broken DAG 오류가 발생합니다. 허용되는 가장 오래된 버전을 명시적으로 지정하는 것이 좋습니다. 예를 들어, `awswrangler==2.4.0` 앞에 최소값 `pyarrow==2.0.0`을 지정하면 오류가 사라지고 `requirements.txt`가 올바르게 설치됩니다. 최종 요구 사항은 다음과 같아야 합니다.

```
cython==0.29.21
pyarrow==2.0.0
awswrangler==2.4.0
```

4. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

연산자

다음 항목에서는 연산자를 사용할 때 발생할 수 있는 오류에 대해 설명합니다.

BigQuery 연산자를 사용할 때 오류가 발생함.

Amazon MWAA는 UI 확장이 있는 연산자를 지원하지 않습니다. 다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. 해결 방법은 문제 연산자를 가져온 후 설정할 DAG에 `<operator name>.operator_extra_links = None`을 설정하는 줄을 추가하여 확장을 재정의하는 것입니다. 예:

```
from airflow.contrib.operators.bigquery_operator import BigQueryOperator
BigQueryOperator.operator_extra_links = None
```

3. 위의 내용을 플러그인에 추가하면 모든 DAG에 이 접근 방식을 사용할 수 있습니다. 예시는 the section called “PythonVirtualenvOperator를 패치하는 사용자 지정 플러그인 ” 단원을 참조하십시오.

연결

다음 항목에서는 Apache Airflow 연결을 사용하거나 다른 AWS 데이터베이스를 사용할 때 발생할 수 있는 오류에 대해 설명합니다.

Snowflake에 연결할 수 없음

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. 사용자 환경의 requirements.txt 파일에 다음 항목을 추가합니다.

```
asn1crypto == 0.24.0
snowflake-connector-python == 1.7.2
```

3. DAG에 다음 가져오기를 추가합니다.

```
from airflow.contrib.hooks.snowflake_hook import SnowflakeHook
from airflow.contrib.operators.snowflake_operator import SnowflakeOperator
```

Apache Airflow 연결 객체에 다음 키-값 쌍이 포함되어 있는지 확인합니다.

1. Conn Id: snowflake_conn
2. Conn Type: Snowflake
3. Host: <my account>.<my region if not us-west-2>.snowflakecomputing.com
4. Schema: <my schema>
5. Login: <my user name>
6. Password: ********

7. Port: <port, if any>
8. 추가:

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

예:

```
>>> import json
>>> from airflow.models.connection import Connection
>>> myconn = Connection(
...    conn_id='snowflake_conn',
...    conn_type='Snowflake',
...    host='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...    schema='YOUR_SCHEMA'
...    login='YOUR_USERNAME',
...    password='YOUR_PASSWORD',
...    port='YOUR_PORT'
...    extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Secrets Manager에 연결할 수 없음

다음 단계를 수행하는 것이 좋습니다.

1. the section called “Secrets Manager 구성”에서 Apache Airflow 연결 및 변수를 위한 암호 키를 만드는 방법을 알아봅니다.
2. Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용에서 Apache Airflow 변수(`test-variable`)에 대해 암호 키를 사용하는 방법을 알아봅니다.
3. Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용에서 Apache Airflow 연결(`myconn`)에 대해 암호 키를 사용하는 방법을 알아봅니다.

'<DB-identifier-name>.cluster-id.<region>.rds.amazonaws.com'에서 내 MySQL 서버에 연결할 수 없음

Amazon MWAA의 보안 그룹과 RDS 보안 그룹에는 서로 트래픽을 허용하기 위한 수신 규칙이 필요합니다. 다음 단계를 수행하는 것이 좋습니다.

1. Amazon MWAA의 VPC 보안 그룹에서의 모든 트래픽을 허용할 수 있도록 RDS 보안 그룹을 수정합니다.
2. RDS 보안 그룹에서의 모든 트래픽을 허용할 수 있도록 Amazon MWAA의 VPC 보안 그룹을 수정합니다.
3. 작업을 다시 실행하고 CloudWatch Logs에서 Apache Airflow 로그를 확인하여 SQL 쿼리가 성공했는지 확인합니다.

웹 서버

다음 항목에서는 Amazon MWAA의 Apache Airflow 웹 서버에서 발생할 수 있는 오류에 대해 설명합니다.

BigQuery Operator를 사용하고 있는데 이로 인해 웹 서버가 다운됨

다음 단계를 수행하는 것이 좋습니다.

1. `operator_extra_links`가 포함된 `BigQueryOperator` 및 `QuboleOperator`와 같은 Apache Airflow 연산자는 Apache Airflow 웹 서버의 작동을 중단시킬 수 있습니다. 이러한 연산자는 보안상의 이유로 허용되지 않는 웹 서버에 코드를 로드하려고 시도합니다. import 문 뒤에 다음 코드를 추가하여 DAG의 연산자를 패치하는 것이 좋습니다.

   ```
   BigQueryOperator.operator_extra_links = None
   ```

2. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.

웹 서버에 액세스하는 중에 5xx 오류가 표시됨

다음 단계를 수행하는 것이 좋습니다.

1. Apache Airflow 구성 옵션을 확인합니다. Apache Airflow 구성 옵션으로 지정한 키-값 쌍(예: AWS Secrets Manager)이 올바르게 구성되었는지 확인합니다. 자세한 내용은 the section called "Secrets Manager에 연결할 수 없음" 단원을 참조하십시오.
2. `requirements.txt`를 확인합니다. Airflow "extras" 패키지 및 `requirements.txt`에 나열된 기타 라이브러리가 Apache Airflow 버전과 호환되는지 확인합니다.
3. `requirements.txt` 파일에서 Python 종속성을 지정하는 방법을 살펴보려면 requirements.txt에서의 Python 종속성 관리 단원을 참조하십시오.

'스케줄러가 실행되지 않는 것 같습니다' 오류가 표시됨

스케줄러가 실행되지 않는 것으로 보이거나 마지막 "heart beat"가 몇 시간 전에 수신된 경우 DAG가 Apache Airflow에 표시되지 않을 수 있으며 새 작업이 예약되지 않습니다.

다음 단계를 수행하는 것이 좋습니다.

1. VPC 보안 그룹이 포트 5432에 대한 인바운드 액세스를 허용하는지 확인합니다. 이 포트는 사용자 환경의 Amazon Aurora PostgreSQL 메타데이터 데이터베이스에 연결하는 데 필요합니다. 이 규칙을 추가한 후 Amazon MWAA에 몇 분 정도 시간을 주면 오류가 사라집니다. 자세한 내용은 the section called "VPC 보안" 단원을 참조하십시오.

   > **Note**
   >
   > - Aurora PostgreSQL 메타데이터베이스는 Amazon MWAA 서비스 아키텍처의 일부이며 AWS 계정에서는 볼 수 없습니다.
   > - 데이터베이스 관련 오류는 일반적으로 스케줄러 오류의 증상이지 근본 원인은 아닙니다.

2. 스케줄러가 실행되지 않는 경우 종속성 설치 실패 또는 스케줄러 과부하와 같은 여러 요인 때문일 수 있습니다. CloudWatch Logs의 해당 로그 그룹을 확인하여 DAG, 플러그인 및 요구 사항이 제대로 작동하는지 확인합니다. 자세한 내용은 모니터링 및 지표 단원을 참조하십시오.

작업

다음 항목에서는 환경에서 Apache Airflow 작업을 수행할 때 발생할 수 있는 오류에 대해 설명합니다.

작업이 중단되거나 완료되지 않은 것으로 보임

Apache Airflow 작업이 “멈춘” 상태이거나 완료되지 않는 경우 다음 단계를 따르는 것이 좋습니다.

1. 정의된 DAG 수가 많을 수 있습니다. DAG 수를 줄이고 환경 업데이트(예: 로그 수준 변경)를 수행하여 강제로 재설정합니다.

   a. Airflow는 DAG의 활성화 여부를 구문 분석합니다. 환경 용량의 50% 이상을 사용하는 경우 Apache Airflow 스케줄러가 너무 많이 사용되기 시작할 수 있습니다. 이로 인해 CloudWatch 지표의 총 구문 분석 시간이 길어지거나 CloudWatch Logs의 DAG 처리 시간이 길어집니다. Apache Airflow 구성을 최적화하는 다른 방법도 있으나 이 설명서에서는 다루지 않습니다.

   b. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called “Apache Airflow 성능 조정” 단원을 참조하십시오.

2. 대기열에 작업 수가 많을 수 있습니다. 이는 보통 "None" 상태의 작업 수가 많이 증가하는 것으로 나타나거나 CloudWatch의 대기 중인 작업 및/또는 보류 중인 작업 수가 많을 때 나타납니다. 이는 다음과 같은 이유로 발생할 수 있습니다.

   a. 환경에 실행할 수 있는 용량보다 실행할 작업이 많거나 자동 크기 조정 전에 대기열에 대기중인 작업이 많을 경우 작업을 감지하고 추가 작업자를 배포할 시간이 있습니다.

   b. 환경에서 실행할 수 있는 용량보다 실행할 작업이 많으면 DAG가 동시에 실행하는 작업 수를 줄이거나 최소 Apache Airflow 작업자를 늘리는 것이 좋습니다.

   c. 자동 크기 조정이 추가 작업자를 감지하고 배포할 시간을 갖기 전에 대기열에 대기 중인 작업이 많은 경우에는 작업 배포에 시차를 두거나 최소 Apache Airflow 작업자를 늘리는 것이 좋습니다.

   d. AWS Command Line Interface(AWS CLI)의 update-environment 명령을 사용하여 사용자 환경에서 실행되는 최소 또는 최대 작업자 수를 변경할 수 있습니다.

   ```
   aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-
   workers 10
   ```

   e. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called “Apache Airflow 성능 조정” 단원을 참조하십시오.

3. 실행 도중에 삭제되는 작업은 Apache Airflow에 더 이상 표시되지 않고 중지되는 작업 로그로 표시될 수 있습니다. 이는 다음과 같은 이유로 발생할 수 있습니다.

   a. 1) 현재 작업이 현재 환경 용량을 초과하고, 2) 몇 분 동안 아무 작업도 실행되지 않거나 대기 중인 상태가 지속되면 3) 새 작업이 대기열에 추가되는 경우 이러한 상태가 발생합니다.

b. Amazon MWAA 자동 크기 조정은 작업자를 추가하여 첫 번째 시나리오에 대응합니다. 두 번째 시나리오에서는 추가 작업자를 제거합니다. 대기열에 있는 일부 작업의 경우 작업자가 제거되는 과정으로 이어질 수 있으며 컨테이너가 삭제되면 종료됩니다.

c. 해당 환경의 최소 작업자 수를 늘리는 것이 좋습니다. 또 다른 옵션은 DAG 및 작업의 타이밍을 조정하여 이러한 시나리오가 발생하지 않도록 하는 것입니다.

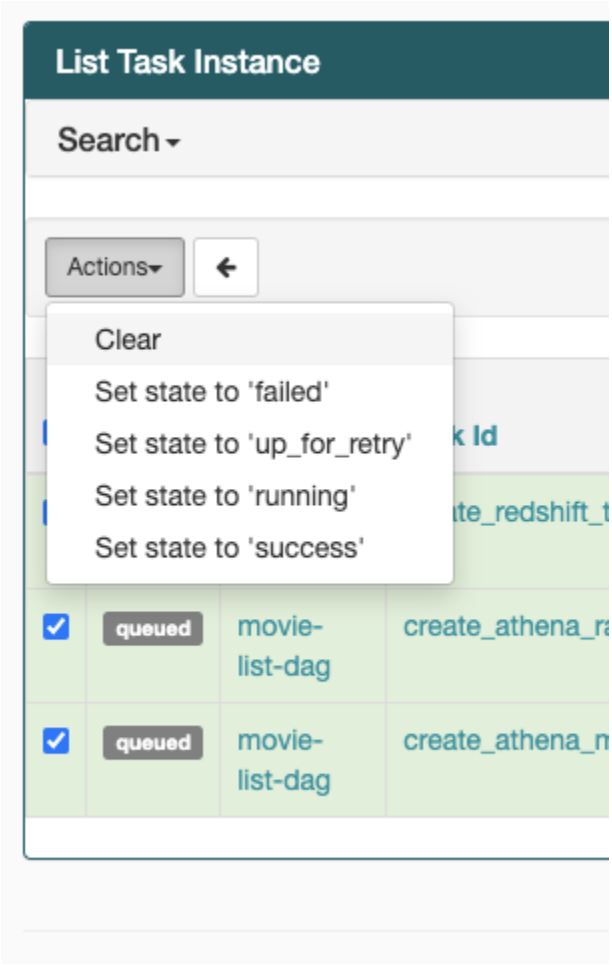
d. 또한 최소 작업자를 사용자 환경의 최대 작업자와 동일하게 설정하여 자동 크기 조정을 효과적으로 비활성화할 수 있습니다. AWS Command Line Interface(AWS CLI)의 update-environment 명령을 사용하여 최소 및 최대 작업자 수를 동일하게 설정해 자동 크기 조정을 비활성화할 수 있습니다.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-
workers 5
```

e. 환경 성능 조정을 권장하는 모범 사례에 대해 자세히 알아보려면 the section called “Apache Airflow 성능 조정” 단원을 참조하십시오.

4. 작업이 “실행 중” 상태에서 멈춘 경우 작업을 지우거나 성공 또는 실패로 표시할 수도 있습니다. 이렇게 하면 사용자 환경의 자동 확장 구성 요소를 통해 환경에서 실행되는 작업자 수를 줄일 수 있습니다. 다음 이미지는 중단된 작업의 예를 보여줍니다.

- 문제 있는 작업의 원을 선택한 다음 지우기(그림 참조)를 선택합니다. 이렇게 하면 Amazon MWAA에서 작업자의 규모를 축소할 수 있습니다. 그렇지 않으면 Amazon MWAA에서 어떤 DAG가 활성화되었는지 또는 비활성화되었는지 확인할 수 없으며 대기 중인 작업이 있는 경우 규모를 축소할 수 없습니다.



5. Apache Airflow 참조 가이드의 개념에서 Apache Airflow 작업 수명 주기에 대해 자세히 알아봅니다.

CLI

다음 항목에서는 AWS Command Line Interface에서 Airflow CLI 명령을 실행할 때 발생할 수 있는 오류에 대해 설명합니다.

CLI에서 DAG를 트리거할 때 '503' 오류가 표시됨.

Airflow CLI는 동시성이 제한된 Apache Airflow 웹 서버에서 실행됩니다. 일반적으로 최대 4개의 CLI 명령을 동시에 실행할 수 있습니다.

문제 해결: Amazon MWAA 환경 생성 및 업데이트

이 페이지의 항목에는 Amazon Managed Workflows for Apache Airflow 환경을 생성 및 업데이트할 때 발생할 수 있는 오류와 이러한 오류를 해결하는 방법이 포함되어 있습니다.

목차

`requirements.txt` 업데이트

다음 항목에서는 `requirements.txt` 업데이트 시 발생할 수 있는 오류에 대해 설명합니다.

`requirements.txt`의 새 버전을 지정했는데 환경 업데이트에 20분 이상 소요됨

사용자 환경에서 새 버전의 `requirements.txt` 파일을 설치하는 데 20분 이상 걸리는 경우, 환경 업데이트는 실패하고 Amazon MWAA는 컨테이너 이미지의 마지막 안정 버전으로 롤백합니다.

1. 패키지 버전을 확인합니다. `requirements.txt`의 Python 종속성에 대해 항상 특정 버전(==) 또는 최대 버전(>=)을 지정하는 것이 좋습니다.
2. Apache Airflow 로그를 확인합니다. Apache Airflow 로그를 활성화한 경우, CloudWatch 콘솔의 로그 그룹 페이지에서 로그 그룹이 성공적으로 생성되었는지 확인합니다. 빈 로그가 표시되는 경우 가장 일반적인 이유는 로그가 기록되는 CloudWatch 또는 Amazon S3의 실행 역할에 권한이 없기 때문입니다. 자세한 내용은 실행 역할 단원을 참조하십시오.
3. Apache Airflow 구성 옵션을 확인합니다. Secrets Manager를 사용하는 경우 Apache Airflow 구성 옵션으로 지정한 키-값 쌍이 올바르게 구성되었는지 확인합니다. 자세한 내용은 the section called “Secrets Manager 구성” 단원을 참조하십시오.
4. VPC 네트워크 구성을 확인합니다. 자세한 내용은 the section called “환경 중단” 단원을 참조하십시오.
5. 실행 역할 권한을 확인합니다. 실행 역할은 Amazon MWAA에 사용자를 대신하여 다른 AWS 서비스(예: Amazon S3, CloudWatch, Amazon SQS, Amazon ECR)의 리소스를 호출할 수 있는 권한을 부여하는 권한 정책이 있는 AWS Identity and Access Management (IAM) 역할입니다. 고객 관리형 키 또는 AWS 소유 키에도 액세스를 허용해야 합니다. 자세한 내용은 실행 역할 단원을 참조하십시오.
6. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.

플러그인

다음 항목에서는 Apache Airflow 플러그인을 구성하거나 업데이트할 때 발생할 수 있는 문제를 설명합니다.

Amazon MWAA는 사용자 지정 UI 구현을 지원합니까?

Apache Airflow v2.2.2부터 Amazon MWAA는 Apache Airflow 웹 서버에 플러그인을 설치하고 사용자 지정 UI를 구현하는 것을 지원합니다. Amazon MWAA 환경에서 Apache Airflow v2.0.2 또는 이전 버전을 실행하는 경우 사용자 지정 UI를 구현할 수 없습니다.

버전 관리 및 기존 환경 업그레이드에 대한 자세한 내용은 버전 단원을 참조하십시오.

플러그인을 통해 Amazon MWAA 로컬 러너에서 사용자 지정 UI 변경을 구현할 수 있지만 Amazon MWAA에서 동일한 작업을 시도해도 변경 내용이나 오류가 표시되지 않습니다. 왜 이런 일이 발생합니까?

Amazon MWAA 로컬 러너에는 모든 Apache Airflow 구성 요소가 하나의 이미지로 번들로 제공되므로 사용자 지정 UI 플러그인 변경을 적용할 수 있습니다.

버킷 생성

다음 항목에서는 Amazon S3 버킷을 생성할 때 발생할 수 있는 오류에 대해 설명합니다.

S3 퍼블릭 액세스 차단 설정 옵션을 선택할 수 없음

Amazon MWAA 환경의 실행 역할에는 버킷이 퍼블릭 액세스를 차단했는지 확인하기 위해 Amazon S3 버킷에서의 `GetBucketPublicAccessBlock` 작업에 대한 권한이 필요합니다. 다음 단계를 수행하는 것이 좋습니다.

1. 단계에 따라 실행 역할에 JSON 정책을 연결합니다.
2. 다음 JSON 정책을 첨부합니다.

```
{
   "Effect":"Allow",
   "Action":[
      "s3:GetObject*",
      "s3:GetBucket*",
      "s3:List*"
   ],
   "Resource":[
      "arn:aws:s3:::YOUR_S3_BUCKET_NAME",
      "arn:aws:s3:::YOUR_S3_BUCKET_NAME/*"
   ]
}
```

   *YOUR_S3_BUCKET_NAME*의 샘플 자리 표시자를 *my-mwaa-unique-s3-bucket-name*과 같은 Amazon S3 버킷 이름으로 대체합니다.

3. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.

환경 생성

다음 항목에서는 환경을 생성할 때 발생할 수 있는 오류에 대해 설명합니다.

환경을 생성하려고 했지만 "생성 중" 상태에서 멈춤

다음 단계를 수행하는 것이 좋습니다.

1. 퍼블릭 라우팅으로 VPC 네트워크를 확인합니다. 인터넷에 액세스할 수 있는 Amazon VPC를 사용하는 경우 다음을 확인합니다.

    - the section called “네트워킹에 대해”에 정의된 대로 Amazon VPC가 Amazon MWAA 환경에서 사용되는 다양한 AWS 리소스 간의 네트워크 트래픽을 허용하도록 구성되어 있어야 합니다. 예를 들어 VPC 보안 그룹은 자체 참조 규칙에서 모든 트래픽을 허용하거나 HTTPS 포트 범위 443 및 TCP 포트 범위 5432의 포트 범위를 선택적으로 지정해야 합니다.
2. 프라이빗 라우팅으로 VPC 네트워크를 확인합니다. 인터넷 액세스 없이 Amazon VPC를 사용하는 경우 다음을 확인합니다.

    - the section called “네트워킹에 대해”에 정의된 대로 Amazon VPC가 Amazon MWAA 환경의 서로 다른 AWS 리소스 간 네트워크 트래픽을 허용하도록 구성되어 있어야 합니다. 예를 들어, 두 개의 프라이빗 서브넷에는 NAT 게이트웨이(또는 NAT 인스턴스)에 대한 라우팅 테이블이나 인터넷 게이트웨이가 없어야 합니다.
3. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.

환경을 생성하려고 했지만 상태가 “생성 실패”로 표시됨

다음 단계를 수행하는 것이 좋습니다.

1. VPC 네트워크 구성을 확인합니다. 자세한 내용은 the section called “환경 중단” 단원을 참조하십시오.
2. 사용자 권한을 확인합니다. Amazon MWAA는 환경을 생성하기 전에 사용자 보안 인증을 대상으로 테스트 런을 수행합니다. AWS 계정에 AWS Identity and Access Management (IAM)에서 환경에 대한 일부 리소스를 생성할 수 있는 권한이 없을 수 있습니다. 예를 들어 프라이빗 네트워크 Apache Airflow 액세스 모드를 선택한 경우 관리자는 사용자 환경에 대한 AmazonMWAAFullConsoleAccess 액세스 제어 정책에 대한 액세스 권한을 계정에 부여해야 합니다. 이를 통해 AWS 계정은 VPC 엔드포인트를 생성할 수 있습니다.

3. 실행 역할 권한을 확인합니다. 실행 역할은 Amazon MWAA에 사용자를 대신하여 다른 AWS 서비스(예: Amazon S3, CloudWatch, Amazon SQS, Amazon ECR)의 리소스를 호출할 수 있는 권한을 부여하는 권한 정책이 있는 AWS Identity and Access Management (IAM) 역할입니다. 고객 관리형 키 또는 AWS 소유 키에도 액세스를 허용해야 합니다. 자세한 내용은 실행 역할 단원을 참조하십시오.
4. Apache Airflow 로그를 확인합니다. Apache Airflow 로그를 활성화한 경우, CloudWatch 콘솔의 로그 그룹 페이지에서 로그 그룹이 성공적으로 생성되었는지 확인합니다. 빈 로그가 표시되는 경우 가장 일반적인 이유는 로그가 기록되는 CloudWatch 또는 Amazon S3의 실행 역할에 권한이 없기 때문입니다. 자세한 내용은 실행 역할 단원을 참조하십시오.
5. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.
6. 인터넷 액세스 없이 Amazon VPC를 사용하는 경우, Amazon S3 게이트웨이 엔드포인트를 생성하고 Amazon S3에 액세스하는 데 필요한 최소 권한을 Amazon ECR에 부여했는지 확인합니다. Amazon S3 게이트웨이 엔드포인트 생성에 대한 자세한 내용은 다음을 참조하십시오.
   - 인터넷 액세스 없이 Amazon VPC 네트워크 생성
   - Amazon Elastic Container Registry 사용 설명서의 Amazon S3 게이트웨이 엔드포인트 생성

VPC를 선택하려고 했는데 “네트워크 장애” 오류가 발생함

다음 단계를 수행하는 것이 좋습니다.

- 환경을 생성하는 경우 Amazon VPC를 선택하려고 할 때 “네트워크 장애” 오류가 표시되면 실행 중인 브라우저 내 프록시를 모두 끄고 다시 시도하십시오.

환경을 생성하려고 시도했지만 서비스, 파티션 또는 리소스가 "전달되어야 함" 오류가 발생함

다음 단계를 수행하는 것이 좋습니다.

- Amazon S3 버킷에 지정한 URI의 URI 끝에 '/'가 포함되어 있기 때문에 이 오류가 표시될 수 있습니다. 경로에서 '/'를 제거하는 것이 좋습니다. 값은 다음 형식이어야 합니다.

```
s3://your-bucket-name
```

환경을 생성하려고 시도했는데 상태가 “사용 가능”으로 표시되지만 Airflow UI에 액세스하려고 하면 “서버에서 빈 응답” 또는 “502 Bad Gateway” 오류가 표시됨

다음 단계를 수행하는 것이 좋습니다.

1. VPC 보안 그룹 구성을 확인합니다. 자세한 내용은 the section called “환경 중단” 단원을 참조하십시오.
2. `requirements.txt`에 나열된 모든 Apache Airflow 패키지가 Amazon MWAA에서 실행 중인 Apache Airflow 버전에 해당하는지 확인합니다. 자세한 내용은 Python 종속성 설치 단원을 참조하십시오.
3. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.

환경을 생성하려고 했는데 사용자 이름이 여러 개의 무작위 문자 이름임

- Apache Airflow의 사용자 이름은 최대 64자입니다. AWS Identity and Access Management (IAM) 역할이 이 길이를 초과하는 경우 해시 알고리즘을 사용하여 고유성을 유지하면서 이를 줄입니다.

환경 업데이트

다음 항목에서는 환경을 업데이트할 때 발생할 수 있는 오류에 대해 설명합니다.

환경 클래스를 변경하려고 시도했지만 업데이트에 실패함

환경을 다른 환경 클래스로 업데이트(예: `mw1.medium`에서 `mw1.small`로 변경)했는데 환경 업데이트 요청이 실패하면 환경 상태가 상태가 `UPDATE_FAILED` 상태로 전환되고 환경이 이전의 안정 버전 환경으로 롤백되며 이에 따라 요금이 청구됩니다.

다음 단계를 수행하는 것이 좋습니다.

1. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.
2. Amazon MWAA 환경에 대한 Amazon VPC 네트워크 설정 및 구성을 확인하는 문제 해결 스크립트를 실행하려면 GitHub의 AWS 지원 도구에서 환경 확인 스크립트를 참조하십시오.

액세스 환경

다음 항목에서는 환경에 액세스할 때 발생할 수 있는 오류에 대해 설명합니다.

Apache Airflow UI에 액세스할 수 없음

다음 단계를 수행하는 것이 좋습니다.

1. 사용자 권한을 확인합니다. Apache Airflow UI를 볼 수 있는 권한 정책에 대한 액세스 권한이 부여되지 않았을 수 있습니다. 자세한 내용은 the section called “Amazon MWAA 환경 액세스” 단원을 참조하십시오.
2. 네트워크 액세스를 확인합니다. 프라이빗 네트워크 액세스 모드를 선택했기 때문일 수 있습니다. Apache Airflow UI의 URL이 다음 형식의 `387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-west-2.airflow.amazonaws.com`이면 Apache Airflow 웹 서버에 프라이빗 라우팅을 사용하고 있음을 의미합니다. Apache Airflow 액세스 모드를 퍼블릭 네트워크 액세스 모드로 업데이트하거나 Apache Airflow 웹 서버의 VPC 엔드포인트에 액세스하는 메커니즘을 생성할 수 있습니다. 자세한 내용은 the section called “VPC 엔드포인트에 대한 액세스 관리” 단원을 참조하십시오.

문제 해결: CloudWatch Log 및 CloudTrail 오류

이 페이지의 항목에는 Amazon Managed Workflows for Apache Airflow 환경에서 발생할 수 있는 Amazon CloudWatch Logs 및 AWS CloudTrail 오류에 대한 해결 방법이 포함되어 있습니다.

목차

- 작업 로그에 'log_group에서 원격 로그를 읽을 수 없음: airway-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*{*n.log.' 내 작업 로그에

로그

다음 항목에서는 Apache Airflow 로그를 볼 때 발생할 수 있는 오류에 대해 설명합니다.

내 작업 로그를 볼 수 없거나 'Cloudwatch log_group에서 원격 로그를 읽는 중' 오류가 발생함

Amazon MWAA는 Amazon CloudWatch Logs에서 직접 로그를 읽고 쓰도록 Apache Airflow를 구성했습니다. 작업자가 작업을 시작하지 못하거나 로그를 작성하지 못하면 다음과 같은 오류가 표시됩니다.

```
*** 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.
```

- 다음 단계를 수행하는 것이 좋습니다.

  a. 사용자 환경에 대해 INFO 수준에서 작업 로그를 활성화했는지 확인합니다. 자세한 내용은 Amazon에서 에어플로우 로그 보기 CloudWatch 단원을 참조하십시오.

  b. 환경 실행 역할에 올바른 권한 정책이 있는지 확인합니다.

  c. 연산자 또는 작업이 올바르게 작동하고 있는지, DAG를 구문 분석할 수 있는 충분한 리소스가 있는지, 로드할 적절한 Python 라이브러리가 있는지 확인합니다. 종속성이 올바른지 확인하려면 문제를 일으키는 항목을 찾을 때까지 가져오기를 제거해 보십시오. Amazon MWAA 로컬 러너 도구를 사용하여 Python 종속성을 테스트하는 것이 좋습니다.

로그 없이 작업이 실패함

워크플로우에서 작업이 실패하고 실패한 작업에 대한 로그를 찾을 수 없는 경우 다음과 같이 기본 인수에 queue 파라미터를 설정하고 있는지 확인합니다.

```
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'] }}"
    )
```

문제를 해결하려면 코드에서 queue를 제거하고 DAG를 다시 호출합니다.

CloudTrail에 'ResourceAlreadyExistsException' 오류가 표시됨

```
"errorCode": "ResourceAlreadyExistsException",
    "errorMessage": "The specified log stream already exists",
    "requestParameters": {
        "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
        "logStreamName": "scheduler_cross-account-eks.py.log"
    }
```

apache-airflow-backport-providers-amazon과 같은 특정 Python 요구 사항은 Amazon MWAA가 CloudWatch와 통신하는 데 사용하는 watchtower 라이브러리를 이전 버전으로 롤백합니다. 다음 단계를 수행하는 것이 좋습니다.

- 다음 라이브러리를 requirements.txt에 추가합니다.

  ```
  watchtower==1.0.6
  ```

CloudTrail에 '잘못된 요청' 오류가 표시됨

```
Invalid request provided: Provided role does not have sufficient permissions for s3
 location airflow-xxx-xxx/dags
```

동일한 AWS CloudFormation 템플릿을 사용하여 Amazon MWAA 환경과 Amazon S3 버킷을 생성하는 경우 AWS CloudFormation 템플릿 내에 DependsOn 섹션을 추가해야 합니다. 두 리소스(MWAA 환경 및 MWAA 실행 정책)는 AWS CloudFormation에 종속성을 갖습니다. 다음 단계를 수행하는 것이 좋습니다.

- AWS CloudFormation 템플릿에 다음 **DependsOn** 문을 추가합니다.

```
...
      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:
...
```

예시는 Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼 단원을 참조하십시오.

Apache Airflow 로그에 '64비트 Oracle Client 라이브러리를 찾을 수 없습니다: “libclntsh.so: 공유 객체 파일을 열 수 없음: 해당 파일 또는 디렉터리가 없습니다'라는 메시지가 표시됨

- 다음 단계를 수행하는 것이 좋습니다.
  - Apache Airflow v2를 사용하는 경우 Apache Airflow 구성 옵션으로 `core.lazy_load_plugins : False`을 추가합니다. 자세한 내용은 2에서 구성 옵션을 사용하여 플러그인 로드를 참조하십시오.

스케줄러 로그에 psycopg2 '서버가 예기치 않게 연결을 종료했습니다'라는 메시지가 표시됨

다음과 비슷한 오류가 표시되면 Apache Airflow 스케줄러에 리소스가 부족한 것일 수 있습니다.

```
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.
```

다음 단계를 수행하는 것이 좋습니다.

- 최대 5개의 스케줄러를 지정할 수 있는 Apache Airflow v2.0.2로 업그레이드하는 것을 고려해 보십시오.

DAG 처리 로그에 '작업에 %라고 표시되지만 실행 프로그램이 작업 인스턴스 % 완료(%)라고 보고합니다'라고 표시됩니다.

다음과 비슷한 오류가 표시되면 장기 실행 작업이 Amazon MWAA의 작업 시간 제한에 도달했을 수 있습니다. Amazon MWAA는 하나의 Airflow 작업을 12시간으로 제한하여 작업이 대기열에 갇히거나 자동 크기 조정과 같은 활동을 차단하는 것을 방지합니다.

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
 %s) Was the task killed externally
```

다음 단계를 수행하는 것이 좋습니다.

- 작업을 실행 시간이 짧은 여러 개의 작업으로 나누는 것을 고려해 보십시오. Airflow에는 일반적으로 작업자가 비동기적인 모델이 있습니다. 외부 시스템에서 활동을 호출하고 Apache Airflow Sensors가 폴링하여 완료 시점을 확인합니다. 센서에 장애가 발생하는 경우 작업자의 기능에 영향을 주지 않고 안전하게 재시도할 수 있습니다.

작업 로그에 'log_group에서 원격 로그를 읽을 수 없음: airway-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*{*n.log.' 내 작업 로그에

다음과 비슷한 오류가 표시되는 경우 사용자 환경의 실행 역할에 작업 로그에 대한 로그 스트림을 생성하는 권한 정책이 포함되어 있지 않을 수 있습니다.

```
Could not read remote logs from log_group: airflow-*{*environmentName}-Task
 log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

다음 단계를 수행하는 것이 좋습니다.

- the section called "실행 역할"의 샘플 정책 중 하나를 사용하여 환경의 실행 역할을 수정합니다.

Apache Airflow 버전과 호환되지 않는 `requirements.txt` 파일에 공급자 패키지를 지정했을 수도 있습니다. 예를 들어 Apache Airflow v2.0.2를 사용하는 경우 Airflow 2.1 이상과만 호환되는 apache-airflow-providers-databricks 패키지와 같은 패키지를 지정했을 수 있습니다.

다음 단계를 수행하는 것이 좋습니다.

1. Apache Airflow v2.0.2를 사용하는 경우 `requirements.txt` 파일을 수정하고 `apache-airflow[databricks]`을 추가합니다. 이렇게 하면 Apache Airflow v2.0.2와 호환되는 올바른 버전의 Databricks 패키지가 설치됩니다.
2. GitHub의 aws-mwaa-local-runner를 사용하여 DAG, 사용자 지정 플러그인, Python 종속성을 로컬에서 테스트합니다.

Amazon MWAA 문서 기록

다음 표에서는 2020년 11월 이후의 Amazon MWAA 서비스 설명서에 대한 중요 추가 사항을 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
Amazon MWAA는 웹 서버 자동 크기 조정 및 아파치 에어플로우 REST API를 지원합니다.	Amazon MWAA는 이제 웹 서버의 자동 확장뿐만 아니라 Apache Airflow REST API에 액세스하고 사용할 수 있는 기능을 지원합니다. • the section called “웹 서버 자동 스케일링 구성” • the section called “아파치 에어플로우 REST API 사용”	2024년 5월 16일
자동 스케일링 동작에 대한 설명 개선	Fargate 작업자가 규모를 축소함에 따라 작업자가 새 작업을 수주할 때의 새로운 Amazon MWAA 자동 조정 동작을 반영하도록 다음 주제를 업데이트했습니다. • the section called “작업자 자동 스케일링 구성”	2024년 5월 10일
더 큰 인스턴스 크기 지원	Amazon MWAA는 이제 더 큰 워크로드를 위한 두 가지 더 큰 인스턴스 크기 옵션을 지원합니다., `mw1.xlarge` `mw1.2xlarge` • the section called “환경 기능”	2024년 4월 16일

새 Apache Airflow 버전	Amazon MWAA는 이제 아파치 에어플로우 v2.8.1을 지원합니다. 이 업데이트에는 업데이트된 공급자 패키지에 대한 정보와 Amazon MWAA에서의 Apache Airflow v2.8.1 사용에 대한 세부 정보가 포함되어 있습니다. • 버전 • the section called “아파치 에어플로우 v2.8.1 연결용 공급자 패키지”	2024년 2월 22일
공유 Amazon VPC 지원	Amazon MWAA는 Amazon OpenSearch Service를 사용하는 조직이 소유자 계정의 중앙 공유 Amazon VPC를 사용하여 Amazon MWAA 리소스를 관리할 수 있도록 계정 간 환경 생성을 지원합니다. 이번 출시의 일부로 Amazon MWAA에서는 사용자가 Amazon VPC 엔드포인트를 직접 생성하고 관리할 수 있습니다. • the section called “자체 Amazon VPC 엔드포인트 관리”	2023년 11월 15일

새 Apache Airflow 버전	Amazon MWAA는 이제 Apache Airflow v2.7.2를 지원합니다. 이 업데이트에는 업데이트된 공급자 패키지에 대한 정보와 Amazon MWAA에서 Apache Airflow v2.7.2 사용에 대한 세부 정보가 포함되어 있습니다. • 버전 • the section called “Apache Airflow v2.7.2 연결용 공급자 패키지”	2023년 11월 6일
새 Apache Airflow 버전	Amazon MWAA는 이제 Apache Airflow v2.6.3을 지원합니다. 이 업데이트에는 업데이트된 공급자 패키지에 대한 정보와 Amazon MWAA에서 Apache Airflow v2.6.3 사용에 대한 세부 정보가 포함되어 있습니다. • 버전 • the section called “Apache Airflow v2.6.3 연결을 위한 공급자 패키지”	2023년 8월 9일
버전 사용 중단 정보	Apache Airflow v2.0.2 및 Apache Airflow v2.2.2의 사용 중단 알림 및 일정을 포함하도록 버전 사용 중단에 대한 주제 업데이트. • the section called “Apache Airflow 지원 중단 버전”	2023년 7월 31일

새 주제 및 사용 사례	Amazon MWAA는 마이너 버전 업그레이드를 지원합니다. 이 업데이트에는 환경을 업그레이드하는 방법과 워크플로우 리소스가 업그레이드 대상 Apache Airflow의 버전과 호환되는지 확인하는 방법을 설명하는 다음 새 주제가 포함되어 있습니다. • the section called “버전 업그레이드”	2023년 6월 5일
업데이트된 주제	사용자에게 Amazon MWAA에 대한 전체 콘솔 및 API 액세스 권한을 부여하는 고객 관리형 IAM 정책 업데이트. 이 업데이트에서는 사용자가 Amazon MWAA에 역할을 전달할 수 있도록 하기 위해 `iam:PassRole` 에 권한을 제공해야 하는 이유를 설명합니다. Amazon MWAA는 이러한 권한을 사용하여 사용자를 대신하여 작업을 수행합니다. • the section called “Amazon MWAA 환경 액세스”	2023년 8월 12일

새 지침	조회 패턴 사용에 대한 지침을 제공하기 위해 Amazon MWAA의 AWS Secrets Manager 백엔드로 구성하는 방법에 대한 주제를 업데이트했습니다. 조회 패턴을 사용하면 Apache Airflow가 검색하는 암호의 범위가 좁아지고 Amazon MWAA가 연결과 변수를 검색하기 위해 Secrets Manager에 보내는 API 호출 횟수가 줄어듭니다. 이를 통해 Secrets Manager를 백엔드로 사용하는 것과 관련된 비용이 줄어듭니다. • Secrets Manager 백엔드를 Apache Airflow 구성 옵션으로 생성	2023년 8월 12일
새 Apache Airflow 버전	Amazon MWAA는 이제 Apache Airflow v2.5.1을 지원합니다. 이 업데이트에는 업데이트된 공급자 패키지에 대한 정보와 Amazon MWAA에서 Apache Airflow v2.5.1 사용에 대한 세부 정보가 포함되어 있습니다. • 버전 • the section called “Apache Airflow v2.5.1 연결을 위한 공급자 패키지”	2023년 4월 11일

새 주제 및 사용 사례	Amazon MWAA 환경에서 시작 스크립트 사용에 대한 새 주제를 추가. 이 주제에서는 기존 환경에 대한 시작 스크립트 구성, 이를 사용하여 Linux 런타임을 설치하는 방법 및 환경 변수를 설정에 대해 설명합니다. • the section called “시작 스크립트 사용”	2023년 4월 3일
프라이빗 웹 서버 액세스에 대한 섹션 업데이트	프라이빗 웹 서버 액세스에 대한 다음 주제를 업데이트. 이 업데이트에서는 프라이빗 웹 서버에 액세스할 수 있는 환경에서 Python 휠 아카이브(`.whl`)를 사용하여 종속성을 패키징하고 설치해야 한다는 점을 명확히 합니다. • 프라이빗 웹 서버 액세스 모드	2023년 2월 24일

사용 중단 Apache Airflow 버전에 대한 정보 추가	Amazon MWAA가 Apache Airflow 버전 중단을 관리한 방법에 대한 새로운 정보로 버전 주제를 업데이트. Apache Airflow 최신 버전으로 업그레이드에 대한 섹션 및 Apache Airflow v1과 Apache Airflow v2 간의 변경 사항을 설명하는 섹션 삭제. 새 버전의 Apache Airflow로 마이그레이션하는 방법에 대한 자세한 내용은 Amazon MWAA 마이그레이션 가이드를 참조하십시오. • the section called “Apache Airflow 지원 중단 버전” • the section called “Apache Airflow 버전 지원 및 FAQ”	2023년 2월 17일
Amazon MWAA 컨테이너 지표의 수정 사항	컨테이너 지표 주제를 업데이트하고 `Cluster` 차원 아래에 없는 일련의 잘못된 지표 삭제. `AdditionalWorker` 구성 요소에 대한 `CPUUtilization` 또는 `MemoryUtilization` 지표를 그래프로 표시하고 통계 유형을 설정하여 해당 환경에서 특정 시간에 사용 중인 추가 작업자 수를 평가하는 방법을 설명하는 추가 섹션을 `Sample Count`에 추가. • the section called “추가 작업자 및 웹 서버 컨테이너의 수 평가”	2023년 1월 20일

새 Apache Airflow 버전	Amazon MWAA는 이제 Apache Airflow v2.4.3을 지원합니다. 이 업데이트에는 업데이트된 공급자 패키지에 대한 정보, Amazon MWAA에서 Apache Airflow v2.4.3 사용에 대한 세부 정보, Amazon MWAA에서 각 Apache Airflow 버전에 지원되는 기능에 대한 통합 정보가 포함되어 있습니다. • 버전 • the section called “Apache Airflow v2.4.3 연결을 위한 공급자 패키지”	2023년 1월 5일
서비스 연결 역할에 대한 주제 업데이트	더 이상 필요하지 않을 때 서비스 연결 역할을 삭제하는 방법에 대한 정보를 포함하여 Amazon MWAA가 사용자를 대신하여 AWS 리소스를 생성하고 관리하는 데 사용하는 서비스 연결 역할에 대한 정보가 업데이트되었습니다. 여기에는 Amazon MWAA가 네임스페이스 아래에 추가 CloudWatch 지표를 게시할 수 있도록 허용하는 업데이트된 서비스 연결 역할 권한 정책이 포함됩니다. `AWS/MWAA` • the section called “서비스 연결 역할”	2022년 11월 18일

서비스 지표에 관한 새 주제	AWS/MWAA 네임스페이스 아래에 Amazon MWAA가 내보내는 서비스 지표를 설명하는 새 주제 추가. 여기에는 Amazon ECS 클러스터 지표 스케줄러, 작업자와 웹 서버, Amazon MWAA가 스케줄러와 작업자를 분리할 수 있게 해주는 대기열에 대한 Amazon SQS 지표뿐만 아니라 메타데이터 데이터베이스에 대한 Amazon RDS 지표가 포함됩니다. • the section called “컨테이너, 큐, 데이터베이스 지표”	2022년 11월 18일
새 주제	Amazon MWAA 환경에서 사용할 새 버전의 공급자 패키지를 지정하도록 제약 조건 파일 수정에 대한 새 지침 추가. • the section called “새 제공자 패키지 지정”	2022년 11월 18일
FAQ 항목 업데이트	Amazon MWAA의 HIPAA 자격과 관련된 정보가 업데이트됨. • the section called “HIPAA 규정 준수”	2022년 11월 15일

새 주제	서비스 간 혼동된 대리자를 방지하기 위해 Amazon MWAA 실행 역할 신뢰 정책에 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키 사용에 대한 새 주제 추가. • the section called “교차 서비스 혼동된 대리인 방지”	2022년 10월 21일
새 샘플 코드	사용자 지정 OS 수준 지표를 작성하는 DAG 코드 예제와 업데이트된 지침이 추가되었습니다. CloudWatch • the section called “DAG를 사용하여 사용자 지정 지표 작성”	2022년 9월 13일
새 샘플 코드	업데이트된 지침과 Apache Airflow CLI 토큰을 검색한 다음 지정된 Amazon MWAA 환경에서 DAG를 호출하는 새로운 AWS Lambda Python 코드 예제가 추가되었습니다. • the section called “Lambda를 사용한 DAG 호출”	2022년 9월 12일
새로운 아키텍처 다이어그램	퍼블릭 및 프라이빗 웹 서버를 사용해 Amazon MWAA 환경을 보여주는 새로운 아키텍처 다이어그램 추가. • the section called “Apache Airflow 액세스 모드”	2022년 9월 12일

새 샘플 코드	Apache Airflow CLI 토큰을 검색한 다음 다른 Amazon MWAA 환경에서 또 다른 DAG를 호출하는 업데이트된 지침과 새로운 DAG 코드 예제 추가. • the section called “여러 환경에서 DAG 호출”	2022년 8월 16일
새 샘플 코드	메타데이터 정보에 대해 환경의 Aurora PostgreSQL을 쿼리하고 결과를 CSV 파일에 쓰고 Amazon S3에 파일을 저장하는 업데이트된 지침과 새로운 DAG 추가. • the section called “Amazon S3으로 환경 메타데이터 내보내기”	2022년 8월 12일
새 샘플 코드	업데이트된 지침과 런타임 시 AWS CodeArtifact 토큰을 새로 고치고 Amazon S3에 결과를 저장하는 새 DAG가 추가되었습니다. • the section called “런타임 시 AWS CodeArtifact 토큰 새로 고침”	2022년 8월 3일
새 샘플 코드	Amazon MWAA에서 `ECSOperator` 사용에 대한 업데이트된 지침 및 DAG 코드 샘플 추가. • the section called “`ECSOperator` 사용”	2022년 7월 26일

새 샘플 코드	Amazon MWAA에서 `SSHOperator` 사용에 대한 업데이트된 지침 및 DAG 코드 샘플 추가. • the section called "`SSHOperator` 사용"	2022년 7월 15일
새 샘플 코드	Amazon MWAA에서 dbt Postgres 사용에 대한 새로운 지침 및 DAG 코드 샘플 추가. • the section called "Amazon MWAA에서 dbt 사용"	2022년 6월 17일
새 주제 및 사용 사례	퍼블릭 및 프라이빗 액세스를 통해 Amazon MWAA 환경에서 Python 휠 파일을 사용하여 종속성을 설치하기 위한 새 지침 및 DAG 코드 샘플 추가. • Python 휠을 사용한 종속성 관리	2022년 5월 13일
새 주제 및 사용 사례	Amazon MWAA가 전송할 Apache Airflow 메트릭을 선택하는 방법에 대한 새 지침이 추가되었습니다. CloudWatch • 어떤 Apache Airflow 지표를 보고할지 선택	2022년 4월 19일

새 가이드	Amazon MWAA는 기존 Amazon MWAA 환경뿐만 아니라 자체 관리형 배포에서 Apache Airflow 워크플로우를 마이그레이션하기 위한 마이그레이션 가이드를 제공합니다. • Amazon MWAA 마이그레이션 가이드	2022년 3월 7일
새 주제 및 사용 사례	Apache Airflow 사용자 권한의 변경 사항을 감지하는 솔루션을 포함하여 Apache Airflow를 사용한 작업에 대한 새로운 보안 모범 사례 추가. • the section called “Apache Airflow의 보안 모범 사례”	2022년 2월 18일
새 샘플 코드	Pendulum을 사용하여 시간대 인식 DAG 생성에 대한 새 코드 샘플을 추가하고 사용자 지정 플러그인을 사용하여 Apache Airflow 로그가 생성되는 시간대를 변경하는 방법을 명확히 함. • the section called “DAG 시간대 변경”	2022년 2월 11일

Apache Airflow v2.2.2 출시	Amazon Managed Workflows for Apache Airflow가 이제 Apache Airflow v2.2.2를 지원합니다. v2.2부터 Amazon MWAA는 Apache Airflow 웹 서버에 직접 Python 패키지와 사용자 지정 플러그인을 설치하므로 환경을 보다 유연하게 관리할 수 있습니다. 자세한 내용은 다음을 참조하십시오. • Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전. • the section called “Apache Airflow v2.2.2 연결을 위한 공급자 패키지”. • Apache Airflow 설명서 웹 사이트의 Apache Airflow v2.2.2 변경 로그.	2022년 1월 27일
새로운 튜토리얼	새로운 사용자 지정 Apache Airflow 역할을 생성하고 지정된 DAG의 하위 집합에 대한 사용자 액세스를 제한하기 위해 IAM에서 매핑된 Apache Airflow 사용자에게 역할을 할당하는 방법을 보여주는 새 튜토리얼 추가. • the section called “튜토리얼: 사용자를 DAG의 하위 집합으로 제한”	2021년 12월 8일

수정 사항	CPU 사용을 최적화하기 위해 `scheduler.min_file_process_interval` 의 값 설정에 대한 모범 사례 권장 사항 수정. 실행 역할에서 Secrets Manager 리소스에 대한 액세스 권한을 부여하는 IAM 정책 예제 추가. Secrets Manager 조건 키 사용에 대한 문제 해결 주제 추가. • 스케줄러가 DAG를 구문 분석하는 방식 성능 튜닝 • Amazon MWAA에 Secrets Manager 암호 키에 액세스할 수 있는 권한 제공 • Secrets Manager에 대한 Amazon MWAA 실행 역할의 조건 키 구성	2021년 11월 22일
새 샘플 코드	사용자 지정 플러그인을 사용하여 DAG가 처리되는 시간대를 수정하기 위한 다음 새 코드 샘플 및 bash 운영자 내에서 `dags backfill` Apache Airflow CLI 명령 호출에 대한 새로운 문제 해결 주제가 추가. • the section called “DAG 시간대 변경” • bash 연산자를 사용하여 CLI 명령 다시 채우기	2021년 11월 1일

수정 사항	Amazon ECS 운영자 코드 샘플의 문제를 수정하고, 환경이 로그의 Amazon ECS 작업 로그 그룹에 액세스할 수 있도록 Amazon MWAA 실행 역할에 필요한 추가 권한을 명확히 했습니다. CloudWatch • Amazon ECS 운영자 권한.	2021년 10월 26일
새 샘플 코드	Aurora PostgreSQL 데이터베이스에 DAG 실행과 관련된 정보를 쿼리하고 Amazon S3에 저장된 CSV 파일에 결과를 쓰는 새 코드 샘플 추가. • the section called “Amazon S3으로 환경 메타데이터 내보내기”.	2021년 10월 1일
수정 사항	Amazon MWAA가 대상 Amazon S3 버킷의 새 객체 및 변경된 객체를 스케줄러 및 작업자에게 자동으로 동기화하는 방법에 대한 정보 수정. • DAG 폴더의 작동 방식.	2021년 10월 1일
이제 지원됨	Amazon MWAA는 이제 Apache Airflow 2.0+용 추가 공급자 패키지를 지원합니다. 지원되는 패키지에 대한 자세한 내용은 다음을 참조하십시오. • the section called “Apache Airflow v2.0.2 연결용 공급자 패키지”.	2021년 9월 24일

새 명령 및 절차	인터넷에 액세스하지 않고 Amazon VPC를 사용할 때 Amazon S3 게이트웨이 엔드포인트를 생성하기 위한 추가 지침 및 AWS CLI 명령 예제가 추가되었습니다. • 인터넷에 접속하지 않고 Amazon VPC 네트워크 생성.	2021년 9월 24일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon Elastic Container Service 운영자를 사용하는 새 코드 샘플을 the section called “ECSOperator 사용”에 추가. • Apache Airflow 플러그인 구성 문제에 대한 새로운 문제 해결 주제를 the section called “플러그인”에 추가.	2021년 9월 19일

지원되는 새로운 리전	이제 다음 리전에서 Amazon MWAA를 사용할 수 있습니다. • 아시아 태평양(뭄바이) - ap-south-1 • 아시아 태평양(서울) - ap-northeast-2 • 유럽(런던) - eu-west-2 • 유럽(파리) - eu-west-3 • 캐나다(중부) - ca-central-1 • 남아메리카(상파울루) - sa-east-1 리전 가용성 및 서비스 엔드포인트에 대한 자세한 내용은 다음을 참조하십시오. • AWS 일반 참조에서 Amazon MWAA 엔드포인트 및 할당량	2021년 8월 31일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon MWAA가 계정 수준의 Amazon S3 설정(`s3:GetAccountPublicAccessBlock`)을 가져올 수 있도록 샘플 정책을 Amazon MWAA 실행 역할에서 업데이트.	2021년 8월 27일

수정 사항	다음과 같은 변경 사항이 추가됨: • 의 보안 AWS CloudFormation 그룹에 자체 참조 인바운드 규칙을 사용하도록 템플릿을 수정했습니다. VPC 네트워크 생성 • 의 보안 AWS CloudFormation 그룹에 자체 참조 인바운드 규칙을 사용하도록 템플릿을 수정했습니다. Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼	2021년 8월 27일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Apache Airflow v2.0.2 Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전에서 지원되는 항목 목록에 DAG 데코레이터를 추가.	2021년 8월 20일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • `celery.sync_parallelism` 사용 사례를 Amazon MWAA의 Apache Airflow 성능 튜닝에 추가. • 할당량 페이지에 서비스 엔드포인트를 추가하고 이름을 Amazon Managed Workflow for Apache Airflow 서비스 엔드포인트 및 할당량로 변경. • 사용자 피드백을 기반으로 Amazon Managed Workflows for Apache Airflow 시작하기에서 네트워킹 필수 조건을 명확히 함. • `dags list-runs` 및 `dags next-execution` 을 Apache Airflow CLI 명령 참조의 지원되지 않는 Airflow CLI 명령으로 이동.	2021년 8월 13일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Apache Airflow v2.0.2 변수를 설정, 가져오기 또는 삭제하는 bash 예제를 Apache Airflow CLI 명령 참조에 추가. • Apache Airflow v2.0.2 종속성 및 Airflow 연결 예제를 Amazon RDS for Microsoft SQL Server와 함께 Amazon MWAA 사용에 추가.	2021년 8월 13일
수정 사항	다음과 같은 변경 사항이 추가됨: • 사용자 피드백을 기반으로 Python 코드 샘플을 SSHOperator 을(를) 사용하여 SSH 연결 생성에서 수정.	2021년 8월 13일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • `variables set`을 Apache Airflow CLI 명령 참조의 지원되는 Airflow CLI 명령으로 이동. • 사용자 피드백을 기반으로 Airflow 버전 페이지에서 v2.0.2의 변경 사항에 대한 요약을 Python 종속성 설치에 추가. • 사용자 피드백을 기반으로 Airflow 버전 페이지에서 v2.0.2의 변경 사항에 대한 요약을 Apache Airflow CLI 명령 참조에 추가. • 사용자 피드백을 기반으로 Airflow 버전 페이지에서 v2.0.2의 변경 사항에 대한 요약을 연결 유형 개요에 추가. • 사용자 피드백을 기반으로 Airflow 버전 페이지에서 v2.0.2의 변경 사항에 대한 요약을 사용자 지정 플러그인 설치에 추가. • 사용자 피드백을 기반으로 Airflow 버전 페이지에서 v2.0.2의 변경 사항에 대한 요약을 DAG 추가 또는 업데이트에 추가.	2021년 8월 6일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Apache Airflow v2.0.2 샘플 코드를 DAG를 사용하여 CLI에서 변수 가져오기에 추가. • Apache Airflow v2.0.2 샘플 코드를 Lambda 함수를 사용한 DAG 호출에 추가.	2021년 8월 6일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon Managed Workflows for Apache Airflow 문제 해결에 'Airflow UI에서 내 연결을 볼 수 없습니다'에 대한 문제 해결 주제 추가. • Amazon MWAA가 지원하는 Amazon VPC 목록을 Amazon MWAA에서의 네트워킹에 대해에 추가.	2021년 7월 29일

수정 사항	다음과 같은 변경 사항이 추가됨: • 웹 로그인 토큰을 인쇄할 수 있도록 사용자 피드백을 기반으로 Python 코드 샘플을 아파치 에어플로우 웹 서버 액세스 토큰 생성에서 수정. • 웨어하우스 파라미터에 작은따옴표를 사용하도록 사용자 피드백을 기반으로 Snowflake 연결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에서 수정.	2021년 7월 29일
주제 삭제 또는 이동	다음과 같은 변경 사항이 추가됨: • Amazon Managed Workflows for Apache Airflow에 대한 모니터링 및 지표에서 모든 모니터링 및 지표 설명서 페이지를 포함하도록 기존 페이지를 재구성. • 아파치 에어플로우 v2 환경 메트릭은 다음과 같습니다. CloudWatch 을 모니터링 및 지표 탐색 메뉴로 이동.	2021년 7월 23일

새 가이드	다음과 같은 변경 사항이 추가됨: • Amazon MWAA 환경에 설치된 Apache Airflow 공급자 패키지 생성 완료. • Amazon MWAA의 모니터링 개요 생성 완료. • 감사 로그인 보기 AWS CloudTrail 생성 완료. • Amazon에서 에어플로우 로그 보기 CloudWatch 생성 완료.	2021년 7월 23일
수정 사항	다음과 같은 변경 사항이 추가됨: • 올바른 순서로 Airflow 연결 문자열을 생성하도록 사용자 피드백을 기반으로 Python 코드 샘플을 수정하고 포트 파라미터를 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 추가. • 사용자 피드백을 기반으로 압축 해제 패키지를 로컬에 설치하는 단계를 Oracle을 사용하여 사용자 지정 플러그인 생성에 추가.	2021년 7월 23일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 에 AWS DMS 운영자에 대한 항목이 추가되었습니다. Amazon MWAA 자주 묻는 질문 • 원격 로그 오류에 대한 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • `variables set`을 Apache Airflow CLI 명령 참조의 지원되지 않는 Airflow CLI 명령으로 이동.	2021년 7월 16일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 사용자 피드백을 기반으로 requirements.txt 파일을 생성하는 순차적 단계를 Python 종속성 설치에 추가. • 사용자 피드백을 기반으로 plugins.zip 파일을 생성하는 순차적 단계를 사용자 지정 플러그인 설치에 추가. • 사용 설명서 전체에서 상호 참조 링크를 Amazon Managed Workflows for Apache Airflow API 참조 가이드의 API 참조 가이드에 추가. • Airflow 2.0 관리자 > 플러그인 메뉴에서 플러그인이 표시되지 않는 이유에 대한 주제를 Amazon MWAA 자주 묻는 질문에 추가.	2021년 7월 9일
새 가이드	다음과 같은 변경 사항이 추가됨: • Amazon S3에서 파일 삭제 생성 완료.	2021년 7월 9일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 지원되는 값 목록을 암호화를 위한 고객 관리형 키 사용에 추가. • 사용자 피드백을 기반으로 프라이빗 리포지토리 URL의 예제를 requirements.txt에서의 Python 종속성 관리에서 업데이트하고 명확히 함.	2021년 7월 2일
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • 에서 AWS Secrets Manager SSH 연결에 개인 키를 사용하기 위한 아파치 에어플로우 v1.10.12 샘플 코드를 추가했습니다. SSHOperator 을(를) 사용하여 SSH 연결 생성	2021년 7월 2일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • StartedTaskInstances FinishedTaskInstances 에 아파치 에어플로우 v2 환경 메트릭은 다음과 같습니다. CloudWatch 메트릭이 추가되었습니다.	2021년 6월 25일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Apache Airflow v2.0.2 샘플 코드를 Amazon EKS에서 Amazon MWAA 사용에 추가.	2021년 6월 25일
새 가이드	다음과 같은 변경 사항이 추가됨: • Amazon MWAA의 Apache Airflow 성능 튜닝 생성 완료.	2021년 6월 25일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Apache Airflow CLI 명령 참조에서 `connections add`및 `connections delete`를 지원되는 Apache Airflow v2.0.2 CLI 명령에 추가. • 에서 사용할 수 있는 최신 버전은 Apache Airflow v2.0.2 AWS CloudFormation at라고 추가했습니다. Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼 • Apache Airflow Workers에서 임시 데이터 저장에 대한 질문을 Amazon MWAA 자주 묻는 질문에 추가. • '실행기가 작업 인스턴스 %s이(가) 완료되었다고 보고합니다' 오류에 대한 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • '서버가 예기치 않게 연결을 종료했습니다' 로그에 대한 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • Bastion Host에 대한 SSH 터널에서 CLI 명령을 실행하는	2021년 6월 18일

예제를 Apache Airflow CLI 토큰 생성에 추가.

- 무작위로 생성된 사용자 이름에 대한 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.
- CLI에서 DAG를 실행할 때 발생하는 503 오류에 대한 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.
- 버전의 기본 설정을 재정의하기 위해 각 Airflow 프로세스를 시작할 때 플러그인을 로드하는 데 `core.lazy_load_plugins : False`의 Airflow 구성 옵션이 필요한 Apache Airflow v2.0.2의 사용자 지정 플러그인에 대한 항목을 Amazon MWAA에서 Apache Airflow 구성 옵션 사용에 추가.
- Apache Airflow v2.0.2 플러그인 샘플 코드에 대한 Airflow 구성 옵션 단계를 Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인 생성에 추가.
- Apache Airflow v2.0.2 플러그인 샘플 코드에 대한 Airflow 구성 옵션 단계를 런타임 환경 변수를 생성하는 사용자 지정 플러그인 생성에 추가.

	• Apache Airflow v2.0.2 플러그인 샘플 코드에 대한 Airflow 구성 옵션 단계를 Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성에 추가. • Apache Airflow v2.0.2 플러그인 샘플 코드에 대한 Airflow 구성 옵션 단계를 Oracle을 사용하여 사용자 지정 플러그인 생성에 추가.	
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Apache Airflow Snowflake 연결에 대한 샘플 코드를 Apache Airflow Snowflake 연결에 AWS Secrets Manager의 암호 키 사용에 추가.	2021년 6월 18일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 서버 측 암호화 지침을 Amazon MWAA용 Amazon S3 버킷 생성에 추가. • Apache Airflow v2.0.2용 암호 백엔드를 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager에 추가. • Apache Airflow Workers 할당량 증가 요청에 대한 질문을 Amazon MWAA 자주 묻는 질문에 추가. • Apache Airflow Worker를 확장할지 여부를 결정하는 데 어떤 지표를 사용하는지에 대한 질문을 Amazon MWAA 자주 묻는 질문에 추가. • 에 사용자 지정 메트릭을 생성하기 위한 질문을 추가했습니다. CloudWatch Amazon MWAA 자주 묻는 질문 • 프라이빗 라우팅을 사용하는 VPC에 대해 Amazon S3 VPC 인터페이스 엔드포인트의 프라이빗 IP 주소를 활성화하는 단계를 프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 서비스 엔드포인트 생성에 추가. • 로컬 포트 전달을 사용하여 SSH 터널을 설정하는 옵션	2021년 6월 2일

	을 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성에 추가.	
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Amazon Aurora PostgreSQL 메타데이터 데이터베이스를 쿼리하고 사용자 지정 지표를 Amazon에 게시하는 DAG용 샘플 코드가 추가되었습니다. CloudWatch CloudWatch에서 DAG를 사용하여 사용자 지정 지표 작성	2021년 6월 2일
새 가이드	다음과 같은 변경 사항이 추가됨: • 연결 유형 개요에서 Apache Airflow UI에서 연결 템플릿을 서로 바꿔서 사용하는 방법에 대한 가이드 생성.	2021년 6월 2일
수정 사항	다음과 같은 변경 사항이 추가됨: • 옵션 3: 인터넷에 액세스할 수 없는 VPC 네트워크 생성에서 AWS CloudFormation 템플릿에 Apache Airflow VPC 엔드포인트를 추가했습니다. VPC 네트워크 생성	2021년 6월 2일

Apache Airflow v2.0.2 출시	Amazon Apache Airflow v2.0.2의 정식 출시 • Amazon Managed Workflows for Apache Airflow의 Apache Airflow 버전 생성 완료. • 아파치 에어플로우 v2 환경 메트릭은 다음과 같습니다. CloudWatch 생성 완료. • Apache Airflow v2.0.2에 대한 버전별 링크를 Amazon MWAA에서 Apache Airflow 구성 옵션 사용에 추가. • Apache Airflow v2.0.2 버전별 지침을 Python 종속성 설치에 추가. • Apache Airflow v2.0.2 버전별 지침을 requirements.txt에서의 Python 종속성 관리에 추가. • Apache Airflow v2.0.2 샘플 플러그인을 사용자 지정 플러그인 설치에 추가. • Apache Airflow v2.0.2 샘플 코드를 Amazon MWAA 환경에서 Aurora PostgreSQL 데이터베이스 정리에 추가. • Apache Airflow v2.0.2 샘플 코드를 Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용에 추가. • Apache Airflow v2.0.2 샘플 코드를 Apache Airflow	2021년 5월 26일

	PythonVirtualenvOperator 용 사용자 지정 플러그인 생성에 추가. • Apache Airflow v2.0.2 명령을 Apache Airflow CLI 명령 참조에 추가. • Apache Airflow v2.0.2 스크립트를 Apache Airflow CLI 토큰 생성에 추가. • Amazon MWAA가 기본적으로 최신 Apache Airflow 버전을 사용한다는 메모를 Amazon MWAA 환경 생성에 추가.	
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 중단되거나 실행되지 않는 Airflow 작업의 문제 해결 지침을 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.	2021년 5월 14일

수정 사항	다음과 같은 변경 사항이 추가됨: • Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인 생성에서 최신 Java 버전을 사용하도록 샘플 플러그인 코드를 업데이트했습니다. 이전에는 `os.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"` 이었습니다.	2021년 5월 12일
주제 삭제 또는 이동	다음과 같은 변경 사항이 추가됨: • Amazon Managed Workflows for Apache Airflow 문제 해결의 주제를 범주별로 새 페이지로 이동.	2021년 5월 10일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon S3 버킷 개요가 Amazon MWAA에서 DAG 작업에 추가.	2021년 5월 10일

주제 삭제 또는 이동	다음과 같은 변경 사항이 추가됨: • 아파치 에어플로우 액세스을 최상위 탐색으로 이동하고, 아파치 에어플로우 웹 서버 액세스 토큰 생성, Apache Airflow CLI 토큰 생성 및 Apache Airflow CLI 명령 참조에 대한 페이지 추가.	2021년 5월 7일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 지원되는 모든 Airflow CLI 명령 및 지원되지 않는 모든 Airflow CLI 명령에 대한 Apache Airflow 참조 가이드에 대한 버전별 링크를 Apache Airflow CLI 명령 참조에 추가. • 모든 구성 옵션에 대한 Apache Airflow 참조 가이드에 대한 버전별 링크를 Amazon MWAA에서 Apache Airflow 구성 옵션 사용에 추가. • requirements.txt에서의 Python 종속성 관리에 Amazon MWAA CLI 유틸리티 추가.	2021년 5월 7일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • plugins.zip 구조를 구성하는 방법에 대한 플랫 및 중첩 예제를 사용자 지정 플러그인 설치에 추가. • DAG 추가 또는 업데이트, 사용자 지정 플러그인 설치 및 Python 종속성 설치페이지에 Amazon MWAA CLI 유틸리티 추가. • 내용을 개요로 재구성하고, Amazon S3에 업로드하고, 사용자 지정 플러그인 설치 및 Python 종속성 설치페이지의 사용자 피드백을 기반으로 Amazon MWAA 섹션에 설치. • 인터넷에 접속하지 않고도 필수 VPC 엔드포인트를 생성하여 기존 Amazon VPC에 연결하는 사용 사례 예제를 Amazon MWAA에서의 네트워킹에 대해에 추가.	2021년 4월 30일
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Secrets Manager for an Apache Airflow 변수에 암호 키를 사용하는 샘플 코드를 Apache Airflow 변수에 AWS Secrets Manager 암호 키 사용에 추가.	2021년 4월 30일

새 가이드	다음과 같은 변경 사항이 추가됨: • 프라이빗 라우팅을 사용하여 Amazon VPC에 필요한 VPC 서비스 엔드포인트 생성 생성 완료.	2021년 4월 30일
수정 사항	다음과 같은 변경 사항이 추가됨: • 이런! Amazon MWAA에서 Apache Airflow 구성 옵션 사용에서 `core.default_ui_timezone` 을 `webserver.default_ui_timezone` 로 업데이트했습니다.	2021년 4월 30일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • SSH 터널에 대한 Windows(PuTTY) 단계를 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성에 추가. • Apache Airflow 2.0과만 호환되는 `apache-airflow-providers-amazon` 에 대한 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.	2021년 4월 23일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Secrets Manager for an Apache Airflow 연결에 암호 키를 사용하는 샘플 코드를 Apache Airflow 연결을 위한 AWS Secrets Manager의 암호 키 사용에 추가.	2021년 4월 23일
새 가이드	다음과 같은 변경 사항이 추가됨: • Amazon MWAA에서의 네트워킹에 대해 생성 완료. • Amazon MWAA에서 VPC 보안 생성 완료. • 아마존 MWAA의 서비스별 아마존 VPC 엔드포인트에 대한 액세스 관리 생성 완료.	2021년 4월 23일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 인터넷에 접속하지 않고도 Amazon VPC 네트워크를 생성할 수 있는 새 AWS CloudFormation 템플릿이 추가되었습니다. VPC 네트워크 생성 • in을 생성하기 위한 새 자습서가 추가되었습니다 AWS Client VPN . 튜토리얼: AWS Client VPN을(를) 사용한 프라이빗 네트워크 액세스 구성 • Apache Airflow 액세스 모드에서 사용자 피드백을 기반으로 네트워킹 액세스 페이지의 이름을 Apache Airflow 액세스 모드로 변경하고 문서를 간소화. • VPC 네트워크 생성에서 사용자 피드백을 기반으로 Amazon VPC 시작하기 정보 및 템플릿만 포함하도록 문서를 간소화. • 에 BigQuery 운영자 해결 방법을 Amazon Managed Workflows for Apache Airflow 문제 해결 추가했습니다. • Apache Airflow v1.10.12 제약 조건 파일 모범 사례를 Python 종속성 설치에 추가.	2021년 4월 16일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • Oracle을 사용하여 사용자 지정 플러그인을 생성하는 샘플 코드를 Oracle을 사용하여 사용자 지정 플러그인 생성에 추가. • 런타임 환경 변수를 생성하는 사용자 지정 플러그인을 생성하기 위한 샘플 코드를 런타임 환경 변수를 생성하는 사용자 지정 플러그인 생성에 추가. •	2021년 4월 16일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • VPC 보안 그룹의 자기 참조 규칙 요구 사항에 대한 주제를 Amazon MWAA 자주 묻는 질문에 추가. • 사용자 지정 플러그인 디렉터리 및 크기 제한을 사용자 지정 플러그인 설치에 추가. • 요구 사항 디렉터리 및 크기 제한을 Python 종속성 설치에 추가. • requirements.txt에서의 Python 종속성 관리에서 `foo.user` 및 `foo.pass`에 대한 Apache Airflow 구성 옵션을 명확히 함. • 구성 옵션 개요를 Amazon MWAA에서 Apache Airflow 구성 옵션 사용에 추가.	2021년 4월 9일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • in을 사용하여 사용자 지정 플러그인을 만들 수 있는 샘플 코드를 추가했습니다 PythonVirtualenvOperator . Apache Airflow PythonVirtualenvOperator용 사용자 지정 플러그인 생성 • Apache Hive 및 Hadoop에서 사용자 지정 플러그인을 생성하는 샘플 코드를 Apache Hive 및 Hadoop을 사용하여 사용자 지정 플러그인 생성에 추가.	2021년 4월 9일
수정 사항	다음과 같은 변경 사항이 추가됨: • 이런! requirements.txt에 대한 형식을 업데이트하고 Python 종속성 설치에서 Apache Airflow v1.10.12와 호환되는 예제를 추가했습니다.	2021년 3월 31일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • requirements.txt 또는 plugins.zip 삭제를 위한 해결 방법을 Amazon MWAA 자주 묻는 질문에 추가. • 환경에서 SSH에 대한 bash 해결 방법을 Amazon MWAA 자주 묻는 질문에 추가. • 에 CloudTrail ResourceAlreadyExistsException 오류에 대한 주제를 추가했습니다Amazon Managed Workflows for Apache Airflow 문제 해결.	2021년 3월 26일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 에 사용된 AWS 서비스 목록을 추가했습니다Amazon MWAA 실행 역할. • 에 사용된 AWS 서비스 목록을 추가했습니다Amazon MWAA의 서비스 연결 역할. • Amazon MWAA용 Python 3.7 버전에 대한 질문을 Amazon MWAA 자주 묻는 질문에 추가. • PythonVirtualenvOperator에 대한 질문을 추가했습니다Amazon MWAA 자주 묻는 질문. • VPC 및 환경 구성과 관련된 모든 주제에 대해 다음 단계로 문제 해결 스크립트를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • Linux Bastion은 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성에서 환경과 동일한 리전에 있어야 한다고 문서를 명확히 함.	2021년 3월 19일

새 가이드	다음과 같은 변경 사항이 추가됨: • at용 AWS Secrets Manager 아파치 에어플로우 연결 안내서를 작성했습니다. 시크릿을 사용하여 Apache 에어플로우 연결 구성 AWS Secrets Manager • 에서 Amazon VPC 인프라, Amazon S3 버킷, Amazon MWAA 환경을 생성하는 AWS CloudFormation 템플릿을 사용하여 퀵 스타트 자습서를 만들었습니다. Amazon Managed Workflows for Apache Airflow용 빠른 시작 튜토리얼	2021년 3월 19일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon S3 버킷 생성 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • JSON 정책을 생성하여 연결하는 단계를 Amazon MWAA 실행 역할에 추가.	2021년 3월 12일

새 샘플 코드	다음과 같은 변경 사항이 추가됨: • DAG를 트리거할 때 구성을 추가하는 샘플 코드를 아파치 에어플로우 액세스에 추가.	2021년 3월 12일
새 가이드	다음과 같은 변경 사항이 추가됨: • requirements.txt에서의 Python 종속성 관리에서 모범 사례 가이드 생성.	2021년 3월 12일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 에 구글/GCP/ 문제 해결 주제를 BigQuery 추가했습니다. Amazon Managed Workflows for Apache Airflow 문제 해결 • Cython 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • MySQL 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • 5xx 웹 서버 오류 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.	2021년 3월 5일

이제 지원됨	다음과 같은 변경 사항이 추가됨: • `backend_kwargs` 이전에는 지원되지 않았으므로 Secrets Manager 함수 호출을 재정의하려면 해결 방법이 필요했습니다. AWS Secrets Manager 이제 `backend_kwargs` 이 지원됩니다. 의 AWS Secrets Manager 문제 해결 항목을 참조하십시오. Amazon Managed Workflows for Apache Airflow 문제 해결	2021년 3월 4일
수정 사항	다음과 같은 변경 사항이 추가됨: • 이런! Amazon MWAA 환경 클래스 구성에서 실제 GB를 반영하도록 각 환경 클래스의 크기를 업데이트했습니다.	2021년 3월 4일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • VPC 엔드포인트 정책을 사용하는 프라이빗 네트워크 액세스를 Apache Airflow 액세스 모드에 추가. • 환경 문제 해결 주제 생성에 대한 추가 확인을 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • `requirements.txt` 에 대한 로그를 볼 수 있는 단계를 Python 종속성 설치에 추가.	2021년 2월 26일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Apache Hive 사용 사례를 Python 종속성 설치에 추가. • Apache Airflow 패키지의 필수 종속성이 `requirements.txt` 의 Python 종속성 설치 파일에 포함되어야 한다고 문서를 명확히 함. • requirements.txt 업데이트 문제 해결 주제를 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가.	2021년 2월 25일

새로운 튜토리얼	다음과 같은 변경 사항이 추가됨: • 프라이빗 네트워크 튜토리얼을 튜토리얼: Linux Bastion Host를 사용한 프라이빗 네트워크 액세스 구성에 추가.	2021년 2월 22일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • 프라이빗 및 퍼블릭 네트워크 구성을 Apache Airflow 액세스 모드에 추가. • 개발 그룹 사용 사례 및 사용자 시나리오를 Amazon MWAA 실행 역할에 추가.	2021년 2월 22일
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • 웹 로그인 토큰 및 CLI 토큰에 대한 샘플 Python 스크립트를 아파치 에어플로우 액세스에 추가. • 다른 환경에서 DAG를 트리거하는 샘플 코드를 Amazon Managed Workflows for Apache Airflow용 코드 예제에 추가. • Lambda 함수를 사용하여 DAG를 트리거하는 샘플 코드를 Lambda 함수를 사용한 DAG 호출에 추가.	2021년 2월 22일

새 명령 및 절차	다음과 같은 변경 사항이 추가됨: • 아파치 에어플로우 액세스에서 모든 스크립트에 단계별 절차 추가.	2021년 2월 22일
새 샘플 코드	다음과 같은 변경 사항이 추가됨: • 아파치 에어플로우 액세스에서 웹 로그인 토큰의 curl 예제를 업데이트. • Amazon RDS Microsoft SQL Server에 연결하기 위한 샘플 코드가 Amazon RDS for Microsoft SQL Server와 함께 Amazon MWAA 사용에 추가.	2021년 2월 17일

새 명령 및 절차	다음과 같은 변경 사항이 추가됨: • Amazon MWAA에서 DAG 작업페이지에 AWS CLI 명령이 추가되었습니다. • Apache Airflow는 CLI 명령에서 직렬화된 DAG를 지원하지 않습니다. CLI는 보안상의 이유로 플러그인이나 요구 사항이 없는 웹 서버에서 실행되므로 plugins.zip 또는 requirements.txt 가 있는 MWAA 환경은 이러한 명령을 지원하지 않습니다. Apache Airflow `list_dags` 및 `backfill` 명령을 아파치 에어플로우 액세스에서 지원되지 않는 명령으로 이동.	2021년 2월 17일
GitHub 시작	사용자 가이드 문서는 이제 오픈 소스로 제공됩니다. GitHub 아무 페이지에서나 “이 페이지 편집”을 선택하십시오. GitHub	2021년 2월 17일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Step Functions 대 Amazon MWAA 사용 사례에 대한 질문을 Amazon Managed Workflows for Apache Airflow 문제 해결에 추가. • CLI 액세스 정책을 Amazon MWAA 환경 액세스에 추가. • 지원되는 모든 Apache Airflow 구성 옵션을 Amazon MWAA에서 Apache Airflow 구성 옵션 사용에서 지정할 수 있다고 문서를 명확히 함. • 한 가용 영역의 Fargate 컨테이너에 오류가 발생하면 MWAA가 VPC 네트워크 생성의 다른 가용 영역의 다른 컨테이너로 전환한다고 문서를 명확히 함.	2021년 2월 12일
새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon MWAA 환경 클래스 구성 추가.	2021년 2월 5일

주제 삭제 또는 이동	다음과 같은 변경 사항이 추가됨: • Amazon Managed Workflows for Apache Airflow 시작하기에서 `airflow-`로 시작하는 Amazon S3 버킷 이름에 대한 요구 사항 삭제. • Amazon MWAA 환경 액세스 및 Amazon MWAA 실행 역할를 Amazon MWAA 환경에 대한 액세스 관리으로 이동.	2021년 2월 4일
아마존 MWAA CloudFormation	파라미터를 업데이트하여 Amazon CloudFormation MWAA에 환경을 생성하십시오. • 제거. SubnetList • 제거 TagList. • 추가 NetworkConfiguration. • 추가 TagMap. • 환경 생성 요청 예제 추가.	2021년 2월 4일

새 주제 및 사용 사례	다음과 같은 변경 사항이 추가됨: • Amazon MWAA에서 Apache Airflow 구성 옵션 사용에 예제 이메일 구성 추가. • 에 PostgresHook 문제 해결 항목이 추가되었습니다Amazon Managed Workflows for Apache Airflow 문제 해결. • 에 AWS Secrets Manager 문제 해결 항목이 추가되었습니다Amazon Managed Workflows for Apache Airflow 문제 해결. • Amazon MWAA 작업자 자동 크기 조정 구성에 고성능 사용 사례 추가됨.	2021년 1월 29일
Amazon MWAA 출시	Amazon Managed Workflow for Apache Airflow의 정식 출시 • 사용 설명서 • AWS CloudFormation 설명서	2020년 11월 24일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.