
Amazon Managed Workflows for Apache Airflow

用户指南



Amazon Managed Workflows for Apache Airflow: 用户指南

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其它商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 Amazon MWAA ?	1
功能	1
架构	2
集成	4
区域可用性	4
支持的版本	4
接下来做什么?	4
快速启动	5
在本教程中:	5
先决条件	6
第一步: 将AWS CloudFormation模板保存到本地	6
步骤二: 使用创建堆栈AWS CLI	12
第三步: 将 DAG 上传到 Amazon S3 并在 Apache Airflow 用户界面中运行	13
第四步: 在日志中查看 CloudWatch 日志	13
接下来做什么?	14
开始使用	15
先决条件	15
关于本指南	15
开始前的准备工作	15
可用区	16
创建存储桶	16
开始前的准备工作	16
创建存储桶	17
接下来做什么?	18
创建 VPC 网络	18
先决条件	18
开始前的准备工作	18
创建亚马逊 VPC 网络的选项	18
接下来做什么?	28
创建环境	28
开始前的准备工作	28
Apache Airflow	28
创建环境	29
接下来做什么?	31
接下来做什么?	18
管理访问权限	32
访问Amazon MWAA 环境	32
工作原理	33
完整的控制台访问权限	33
完整的 API 访问权限	36
控制台的只读访问权限	37
Apache Airflow	38
Apache Airflow	38
创建 JSON 策略	38
示例使用案例	39
接下来做什么?	40
服务相关角色	40
Amazon MWAA 的服务相关角色权限	41
为 Amazon MAA 创建服务相关角色	43
为 Amazon MAA 编辑服务相关角色	43
删除适用于 Amazon MAA 的服务相关角色	43
Amazon MWAA 服务相关角色支持的区域	43
策略更新	43
执行角色	44

执行角色概述	44
创建新角色	45
查看和更新执行角色策略	46
使用账户级公有访问区块授予对 Amazon S3 存储桶的访问权限	47
使用 Apache 气流连接	47
示例策略	47
接下来做什么？	51
跨服务混淆代理问题防范	51
Apache Airflow	52
Apache Airflow	52
访问模式概述	56
私有和公共访问模式的设置	56
访问您的 Apache Airflow Web 服务器的 VPC 终端节点（私有网络访问权限）	57
访问 Apache Airflow 界面	58
先决条件	58
访问	58
AWS CLI	58
打开 Airflow	58
登录 Apache Airflow	58
Apache Airflow 网络登录令牌	59
先决条件	59
使用 AWS CLI	59
使用 bash 脚本	60
使用 POST API 请求	60
使用 Python 脚本	61
接下来做什么？	61
Apache 气流 CLI 令牌	61
先决条件	62
使用 AWS CLI	62
使用 curl 脚本	62
使用 bash 脚本	63
使用 Python 脚本	64
接下来做什么？	66
Apache Airflow Airflow	66
先决条件	67
v2 中有什么变化	67
支持的 CLI 命令	67
示例代码	70
管理连接	73
概览	73
Apache 气流套餐	73
Apache Airflow v2.5.1 连接的提供者软件包	74
Apache Airflow v2.4.3 连接的提供者软件包	74
Apache Airflow v2.2.2 连接的提供者软件包	75
Apache Airflow v2.0.2	75
指定更新的提供程序包	75
连接类型	76
连接 URI 字符串示例	76
示例连接模板	76
使用 HTTP 连接模板进行 Jdbc 连接的示例	77
配置 Secrets Manager	79
第一步：向亚马逊 MWAA 提供访问密钥 Secrets Manager 密钥的权限	79
第二步：将 Secrets Manager 后端创建为 Apache Airflow 配置选项	80
第三步：生成 Apache Airflow AWS 连接 URI 字符串	81
第四步：在 Secrets Manager 中添加变量	82
第五步：在 Secrets Manager 中添加连接	83
示例代码	84

资源	84
接下来做什么？	84
管理环境	85
配置环境类	85
环境类别	85
环境功能	86
气流调度器	87
配置自动扩展	87
工作线程计数	88
工作原理	88
使用 Amazon MWAA 控制台	89
高性能用例示例	89
对卡在运行状态的任务进行故障排除	90
接下来做什么？	90
使用配置选项	90
先决条件	91
工作原理	91
在 Apache Airflow v2 中使用配置选项加载插件	91
配置桶概述	91
配置参考	92
示例和示例代码	96
接下来做什么？	97
使用启动脚本	97
配置启动脚本	97
安装 Linux 运行时	100
设置环境变量	100
使用 DAG	103
Amazon S3 存储桶概述	103
添加或更新 DAG	103
先决条件	104
工作原理	104
v2 中有什么变化	104
使用亚马逊 MWAA CLI 实用程序测试 DAG	105
上传 DAG 代码到 Amazon S3	105
指定 DAG 文件夹的路径	106
在 Apache Airflow UI 上查看更改	106
接下来做什么？	106
安装自定义插件	106
先决条件	107
工作方式	107
v2 中有什么变化	107
自定义插件概述	108
自定义插件示例	108
创建 plugins.zip 文件	114
上传 plugins.zip 到 Amazon S3	115
在您的环境中安装自定义插件	116
plugins.zip 的使用案例示例	116
接下来做什么？	117
安装 Python 的	117
先决条件	117
工作原理	117
Python 依赖关系	118
正在创建 requirements.txt 文件	118
上传 requirements.txt 到 Amazon S3	120
在您的环境中安装 Python 依赖项	121
查看您的日志 requirements.txt	122
接下来做什么？	122

删除 Amazon S3 上的文件	122
先决条件	123
版本控制概述	123
工作方式	123
删除 Amazon S3 上的 DAG	123
正在删除“当前”plugins.zip 或 requirements.txt	124
删除“非当前”plugins.zip 或 requirements.txt	124
删除具有生命周期的文件	124
生命周期策略示例	124
接下来做什么？	125
联网	126
关于网络	126
条款	126
什么受支持	126
VPC 基础设施概述	127
亚马逊 VPC 和 Apache Airflow 访问模式的示例用例	128
您的 VPC 安全	129
条款	130
安全性概述	130
网络访问控制列表 (ACL)	130
VPC 安全组	131
VPC 终端节点策略（仅限私有路由）	132
管理对 VPC 终端节点的访问	132
定价	133
VPC 终端节点概述	133
允许使用其他AWS服务	133
查看 VPC 终端节点	134
访问 Apache Airflow Web 服务器的 VPC 终端节点（专用网络访问）	135
私有亚马逊 VPC 中的 VPC 服务终端节点	136
定价	136
专用网络和私有路由	137
（必需）VPC 终端节点	138
连接所需的 VPC 终端节点	138
（可选）为您的 Amazon S3 VPC 接口终端节点启用私有 IP 地址	141
教程	142
教程：持续交付 (CD) 管道	142
关于工作 GitHub 流程	143
先决条件	143
第一步：（可选）创建 Amazon VPC	143
第二步：构建 Docker 镜像	146
第三步：在亚马逊 ECR 中创建 Docker 镜像注册表	147
第四步：对 Docker 客户端进行身份验证	148
第五步：推送 Docker 镜像	148
第六步：创建 Amazon ECS 集群	150
第七步：将任务定义推送到 GitHub	151
第八步：将您AWS和亚马逊 ECR 信息添加到 GitHub	152
第九步：设置入 GitHub 门工作流程	152
第十步：测试工作 GitHub 流程	154
清除	154
接下来做什么？	155
教程：AWS Client VPN	155
私有网络	155
使用案例	157
开始前的准备工作	157
目标	157
（可选）第一步：确定您的 VPC、CIDR 规则和 VPC 安全性	157
步骤二：创建服务器和客户端证书	158

第三步：将AWS CloudFormation模板保存到本地	159
第四步：创建Client VPNAWS CloudFormation 堆栈	160
第五步：将子网关联到您的Client VPN	160
第六步：向您的Client VPN 添加授权入口规则	160
步骤七：下载 Client VPN 终端节点配置文件	161
第八步：Connect 到AWS Client VPN	162
接下来做什么？	162
教程：Linux 堡垒主机	162
私有网络	163
使用案例	164
开始前的准备工作	164
目标	164
步骤一：创建堡垒实例	164
步骤二：创建 ssh 隧道	165
第三步：将堡垒安全组配置为入站规则	166
第四步：复制 Apache Airflow 网址	166
第五步：配置代理设置	166
第六步：打开 Apache Airflow 用户界面	168
接下来做什么？	168
教程：限制用户访问 DAG 的子集	169
先决条件	169
第一步：使用默认 APublic pache Airflow 角色向您的 IAM 委托人提供亚马逊 MWAA Web 服务 器访问权限。	169
步骤二：创建新的 Adit Aribules	170
第三步：将您创建的角色分配给您的亚马逊 MWAA 用户	170
后续步骤	171
相关资源	171
代码示例	172
导入变量 DAG	172
版本	173
先决条件	173
权限	173
附属物	173
代码示例	173
接下来做什么？	174
使用 SSHOperator	174
版本	175
先决条件	175
权限	175
要求	175
将您的密钥复制到 Amazon S3	175
创建新的 Apache Airflow 连接	175
代码示例	176
Apache Airflow Snowflake 在	177
版本	177
先决条件	177
权限	177
要求	178
代码示例	178
接下来做什么？	178
使用 DAG 写入自定义指标	179
版本	179
先决条件	179
权限	179
附属物	179
代码示例	179
Aurora PostgreSQL 数据库	181

版本	182
先决条件	182
附属物	182
代码示例	182
将环境元数据导出到 Amazon S3	185
版本	185
先决条件	185
权限	185
要求	186
代码示例	186
在Secrets Manager 中使用 Apache Airflow 变	187
版本	188
先决条件	188
权限	188
要求	188
代码示例	188
接下来做什么？	189
在Secrets Manager 中使用 Apache Airflow 连接	189
版本	189
先决条件	189
权限	190
要求	188
代码示例	190
接下来做什么？	192
使用 Oracle 自定义插件	192
版本	192
先决条件	193
权限	193
要求	193
代码示例	193
创建自定义插件	194
气流配置选项	196
接下来做什么？	196
带有环境变量的自定义插件	196
版本	196
先决条件	197
权限	197
要求	197
自定义插件	197
Plugins.zip	197
气流配置选项	198
接下来做什么？	198
更改 DAG 的时区	198
版本	198
先决条件	198
权限	198
创建一个插件来更改 Airflow 日志中的时区	198
创建目标plugins.zip	199
代码示例	199
接下来做什么？	200
RefreshAWS CodeArtifact运行时的令牌	200
版本	201
先决条件	201
权限	201
要求	188
代码示例	202
接下来做什么？	203

使用 Apache Hive 和 Hadoop 的自定义插件	204
版本	204
先决条件	204
权限	204
要求	188
下载依赖项	205
自定义插件	205
Plugins.zip	206
代码示例	206
气流配置选项	206
接下来做什么？	206
要修补的自定义插件 PythonVirtualenvOperator	207
版本	207
先决条件	207
权限	207
要求	207
自定义插件示例代码	208
Plugins.zip	209
代码示例	209
气流配置选项	211
接下来做什么？	211
使用 Lambda 调用 DAG	211
版本	211
先决条件	211
权限	211
附属物	212
代码示例	212
在不同的环境中调用 DAG	213
版本	213
先决条件	213
权限	213
附属物	214
代码示例	214
亚马逊 RDS 服务器	215
版本	215
先决条件	215
附属物	182
Apache Airrow v2 连接	216
代码示例	216
接下来做什么？	218
Amazon EMR 集成	218
版本	218
代码示例	218
亚马逊 EKS (eksctl)	220
版本	220
先决条件	221
为 Amazon EC2 创建公钥	221
创建集群	221
创建mwaa命名空间	222
创建角色mwaa命名空间	222
创建并附加 Amazon EKS 集群的 IAM 角色	223
创建 requirements.txt 文件	225
为亚马逊 EKS 创建身份映射	225
创建 kubeconfig	225
创建 DAG	225
添加 DAG 和kube_config.yaml到Amazon S3 存储桶	227
启用并触发示例	227

使用 ECSOperator	228
版本	228
先决条件	228
权限	228
创建 Amazon ECS 集群	229
代码示例	232
在亚马逊 MWAA 上使用 dbt	234
版本	234
先决条件	234
附属物	235
将项目 dbt 项目上传到 Amazon S3	235
使用 DAG 验证 dbt 依赖项安装	236
使用 DAG 运行 dbt 项目	236
AWS 博客和教程	237
最佳实践	238
Amahe Ahe Ahe Ahe	238
添加气流配置选项	238
气流调度器	239
DAG 文件夹	241
DAG 文件	243
任务	245
延长 Amahe 气流	248
管理 Python 依赖	248
使用亚马逊 MWAA CLI 实用程序测试 DAG	248
使用 PyPi.org 需求文件格式安装 Python 依赖项	249
在亚马逊 MWAA 控制台上启用日志	253
在日志控制台上查看 CloudWatch 日志	253
在 Apache Airflow 用户界面中查看错误	254
示例 requirements.txt 场景	254
监控和指标	255
概览	255
亚马逊 CloudWatch 概述	255
AWS CloudTrail 概览	255
查看审核日志	256
在中创建跟踪 CloudTrail	256
使用事件历史记录查看 CloudTrail 事件	256
的示例路线 CreateEnvironment	256
接下来做什么？	257
查看气流日志	257
定价	258
开始前的准备工作	258
日志类型	258
启用 Apache 气流日志	258
查看 Apache 气流日志	259
调度器日志示例	259
接下来做什么？	259
监控仪表板和警报	260
指标	260
告警状态概述	260
自定义仪表板和警报示例	260
删除指标和仪表板	264
接下来做什么？	264
Apache Airflow v2 环境指标	264
术语	264
维度	265
在 CloudWatch 控制台中访问指标	265
Apache Airflow 指标可用 CloudWatch	266

选择要报告哪些指标	270
接下来做什么？	270
Apache Airflow v1 环境指标	270
术语	271
维度	271
在 CloudWatch 控制台中访问指标	271
Apache Airflow 指标可用 CloudWatch	272
接下来做什么？	274
容器、队列和数据库指标	274
术语	275
维度	275
访问 指标	275
指标列表	276
安全性	279
数据保护	279
加密	280
使用客户托管密钥	281
AWS Identity and Access Management	283
受众	284
使用身份进行身份验证	284
使用策略管理访问	285
允许用户查看他们自己的权限	287
对 Amazon Managed Workflows for Apac	287
Amazon MWAA 如何与 IAM 协同工作	288
合规性验证	291
故障恢复能力	292
基础设施安全性	292
配置和漏洞分析	292
最佳实践	292
Apache Airflow 中的安全最佳实	293
版本	294
关于亚马逊 MWAA 版本	294
最新版本	294
Apache Airfro	294
Apache 气流组件	295
计划程序	295
工作线程	295
Apache Airflow 过时	295
AAAAAAAAAAAAAAA AAA	296
常见问题	296
终端节点和配额	297
服务终端节点	297
Service Quotas	297
提升配额	297
常见问题	298
支持的版本	298
Apache 支持	298
Apache Airfro	299
Python 版本	299
pip 版本	299
使用案例	299
我应在在何时使用AWS Step Functions MWAA？	299
环境规范	300
每个环境有多少任务存储空间可用？	300
默认操作系统	300
自定义映像	300
HIPAA 合规性	300

亚马逊 MWAA 是否支持竞价型实例？	300
自定义域	300
SSH	300
自引用规则	301
自定义指标	301
存储数据	301
工作线程配额	301
共享的 Amazon VPC	301
指标	301
工作线程指标	301
自定义指标	302
DAG、操作员、连接和其他问题	302
PythonVirtualenvOperator	302
亚马逊 MWAA 识别新的 DAG 文件需要多长时间？	302
为什么 Apache Airflow 不接收我的 DAG 文件？	302
移除 plugins.zip 或 requirements.txt	302
移除 plugins.zip 或 requirements.txt	302
我可以使用的 Data Migration Service (DMS) 操作员？	303
故障排除	304
Apache Airflow v2	305
连接	306
Web 服务器	308
任务	308
CLI	310
运算符	311
Apache Airflow v1	312
正在更新 requirements.txt	313
DAG 破碎	313
运算符	314
连接	315
Web 服务器	316
任务	317
CLI	319
亚马逊 MWAA 创建/更新	320
更新 requirements.txt	320
插件	321
创建存储桶	321
创建环境	322
更新环境	323
访问环境	323
CloudWatch 日志和 CloudTrail	324
日志	324
文档历史记录	328
.....	cccliii

什么是 Amazon Managed Workflows (MWAA) ?

Amazon Managed Workflows (MWAA) 是一项[Apache Air](#)这使其更易于设置和操作 end-to-end 云端的大规模数据管道。Apache Airflow 是一种开源工具，用于以编程方式编写、安排和监控被称为“工作流程”的流程和任务序列。使用 Amazon MWAA，您可以使用 Airflow 和 Python 创建工作流程，而无需管理底层基础设施以实现可扩展性、可用性和安全性。Amazon MWAA 会自动扩展其工作流程执行容量以满足您的需求，并且与 AWS 安全服务可帮助您快速安全地访问数据。

目录

- [功能 \(p. 1\)](#)
- [架构 \(p. 2\)](#)
- [集成 \(p. 4\)](#)
- [区域可用性 \(p. 4\)](#)
- [支持的版本 \(p. 4\)](#)
- [接下来做什么？ \(p. 4\)](#)

功能

- 自动气流设置— 通过选择 Apache Airflow 来快速设置[阿帕奇气流版本 \(p. 294\)](#)当您创建 Amazon MWAA 环境时。亚马逊 MWAA 使用与您可以在互联网上下载的相同的 Apache Airflow 用户界面和开源代码为您设置 Apache Airflow。
- 自动扩展— 自动缩放 Apache 气流工作线程通过设置最小和最大数量工作线程它在你的环境中运行。Amazon MWAA 工作线程在你的环境中并使用它[自动扩展 \(p. 87\)](#)添加工作线程以满足需求，直至达到最大数量工作线程你定义了。
- 内置身份验证— 为您的 Apache Airflow 启用基于角色的身份验证和授权 Web 服务器通过定义[访问控制策略 \(p. 85\)](#)在 AWS Identity and Access Management (我是)。Apache Airflow 工作线程假设这些策略可以安全访问 AWS 服务。
- 内置安全— Apache Airflow 工作线程和 Schedule 运行[亚马逊 MWAA 的亚马逊 VPC \(p. 132\)](#)。数据还使用自动加密 AWS Key Management Service，因此您的环境在默认情况下是安全的。
- 公共或私有访问模式— 访问你的 Apache 气流 Web 服务器使用私人或公共[访问模式 \(p. 52\)](#)。这些区域有：公共网络访问模式使用 VPC 终端节点作为你的 Apache Airflow Web 服务器这是可以访问的通过互联网。这些区域有：私有网络访问模式使用 VPC 终端节点作为你的 Apache Airflow Web 服务器这是可以访问的在你的 VPC 中。在这两种情况下，Apache Airflow 用户的访问权限均由您在中定义的访问控制策略控制 AWS Identity and Access Management (IAM)，以及 AWS 所以。
- 简化的升级和补丁— 亚马逊 MWAA 定期提供新版本的 Apache Airflow。亚马逊 MWAA 团队将更新和修补这些版本的映像。
- 工作流监控— 查看 Apache 气流日志和[Apache Airflow \(p. 255\)](#)在 Amazon CloudWatch 无需额外的第三方工具即可识别 Apache Airflow 任务延迟或工作流程错误。Amazon MWAA 会自动将环境指标 (如果启用) Apache Airflow 日志发送到 CloudWatch。
- AWS 一体化— 亚马逊 MWAA 支持与 Amazon Athena 的开源集成，AWS Batch，Amazon CloudWatch，Amazon DynamoDB，AWS DataSync，Amazon EMR，AWS Fargate、Amazon Kinesis

Data Firehose、AWS Glue、AWS Lambda、Amazon Redshift、Amazon SQS、Amazon SNS、Amazon SageMaker、Amazon S3，以及数百个内置和社区创建的操作员和传感器。

- 工作线程实例集— Amazon MWAA 支持使用容器按需扩展工作人员队列并使用以下方法减少调度器中断[Amazon ECS](#)[AWS Fargate](#)。支持在 Amazon ECS 容器上调用任务的操作员，以及在 Kubernetes 集群上创建和运行容器的 Kubernetes 操作员。

架构

外箱中包含的所有组件（如下图所示）在您的账户中显示为单个 Amazon MWAA 环境。Apache Airflow 计划程序和工作线程是 AWS Fargate (Fargate) 连接到您的环境的 Amazon VPC 中的私有子网的容器。每个环境都有自己的 Apache Airflow 元数据库，由 AWS 这是可以访问的计划程序和工作线程通过私有安全的 VPC 终端节点进行 Fargate 容器。

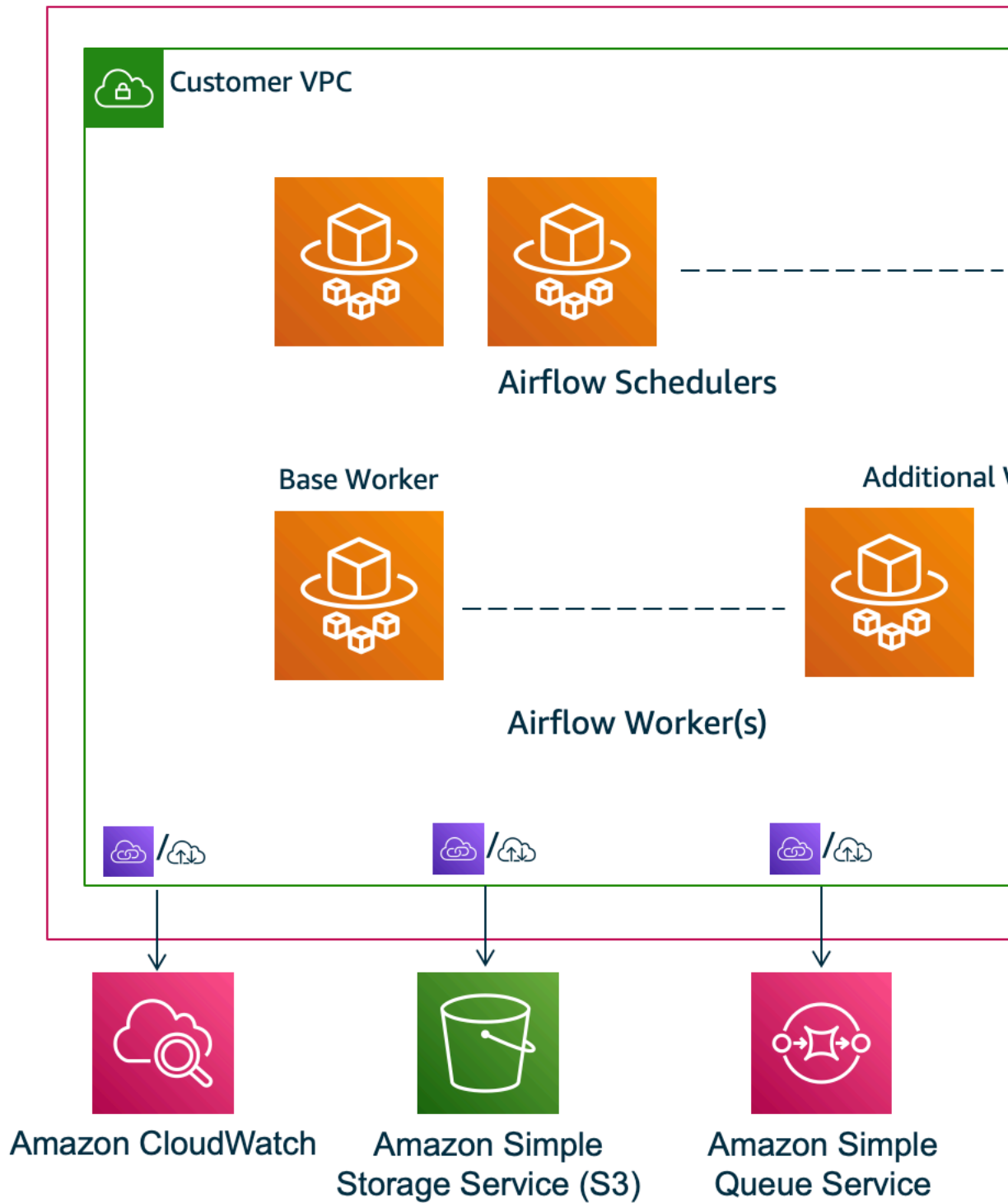
亚马逊 CloudWatch、Amazon S3、Amazon SQS、Amazon ECR 和 AWS KMS 与亚马逊 MWAA 是分开的，需要通过 Apache Airflow 进行访问 Schedule 和工作线程在 Fargate 集装箱里。

Apache Airflow Web 服务器也可以访问通过互联网通过选择公共网络 Apache Airflow 访问模式，或在您的 VPC 中通过选择私有网络 Apache 气流访问模式。在这两种情况下，Apache Airflow 用户的访问权限均由您在主中定义的访问控制策略控制 AWS Identity and Access Management (IAM)。

Note

多 Apache 气流 Schedule 仅适用于 Apache Airflow v2 及更高版本。要详细了解 Apache Airflow 任务生命周期，请访问[概念](#)在里面 Apache Airflows。

Amazon MWAA Architecture



集成

活跃且不断发展的 Apache Airflow 开源社区为运营商（简化服务连接的插件）提供了 Apache Airflow 与之集成AWS服务。这包括 Amazon S3、Amazon Redshift、Amazon EMR 和AWS Batch，和亚马逊 SageMaker，以及其他云平台上的服务。

在亚马逊 MWAA 中使用 Apache Airflow 完全支持与AWS服务和常用的第三方工具，例如 Apache Hadoop、Presto、Hive 和 Spark，用于执行数据处理任务。亚马逊 MWAA 致力于保持与亚马逊 MWAA API 的兼容性，亚马逊 MWAA 打算提供可靠的集成AWS服务并将其提供给社区，并参与社区功能开发。

有关示例代码，请参阅 [Managed Workflows for Airflow \(MWAA\) \(p. 172\)](#)。

区域可用性

Amazon MWAA 有以下可用AWS区域。

- 欧洲（斯德哥尔摩）– eu-north-1
- 欧洲（法兰克福）– eu-central-1
- 欧洲（爱尔兰）– eu-west-1
- 欧洲（伦敦）– eu-west-2
- 欧洲（巴黎）– eu-west-3
- 亚太地区（孟买）– ap-south-1
- 亚太地区（新加坡）– ap-southeast-1
- 亚太地区（悉尼）– ap-southeast-2
- 亚太地区（东京）– ap-northeast-1
- 亚太地区（首尔）– ap-northeast-2
- 美国东部（弗吉尼亚北部）– us-east-1
- 美国东部（俄亥俄）– us-east-2
- 美国西部（俄勒冈）– us-west-2
- 加拿大（中部）– ca-central-1
- 南美洲（圣保罗）– sa-east-1

支持的版本

亚马逊 MWAA 支持多个版本的 Apache Airflow。要了解有关我们支持的 Apache Airflow 版本以及每个版本附带的 Apache Airflow 组件的更多信息，请参阅[Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 294\)](#)。

接下来做什么？

- 使用单步快速入门AWS CloudFormation用于为您的 Airflow DAG 和支持文件创建 Amazon S3 存储桶的模板、带有公共路由的 Amazon VPC 以及中的 Amazon MWAA 环境[Workflows for Apache Airflow \(MWAA\) \(p. 5\)](#)。
- 为你的 Airflow DAG 和支持文件创建 Amazon S3 存储桶，从三个 Amazon VPC 联网选项中进行选择，然后在中创建 Amazon MWAA 环境，逐步开始[开始使用 Amazon Managed Workflows of Apache Airflow \(MWAA\) \(p. 15\)](#)。

Workflows for Apache Airflow (MWAA)

本快速入门教程使用的AWS CloudFormation模板可同时创建 Amazon VPC 基础架构、带dags文件夹的 Amazon S3 存储桶以及适用于 Apache Airflow (MWAA) 的亚马逊托管工作流程。

主题

- [在本教程中：\(p. 5\)](#)
- [先决条件 \(p. 6\)](#)
- [第一步：将AWS CloudFormation模板保存到本地 \(p. 6\)](#)
- [步骤二：使用创建堆栈AWS CLI \(p. 12\)](#)
- [第三步：将 DAG 上传到 Amazon S3 并在 Apache Airflow 用户界面中运行 \(p. 13\)](#)
- [第四步：在日志中查看 CloudWatch 日志 \(p. 13\)](#)
- [接下来做什么？\(p. 14\)](#)

在本教程中：

本教程将引导您完成三AWS Command Line Interface (AWS CLI) 条命令，用于将 DAG 上传到 Amazon S3、在 Apache Airflow 中运行 DAG 以及查看日志 CloudWatch。最后，它将引导您完成为 Apache Airflow 开发团队创建 IAM 策略的步骤。

Note

页面上的AWS CloudFormation模板会创建 Amazon Managed Workflows for Apache Airflow (MWAA) 环境AWS CloudFormation。可用的最新版本是 Apache Airflow v2.5.1。

此页面上的AWS CloudFormation模板创建了以下内容：

- VPC 基础架构。模板使用[通过互联网进行公共路由 \(p. 127\)](#)。它在[公网访问模式 \(p. 56\)](#) Apache Airflow Web 服务器上使用WebserverAccessMode: PUBLIC_ONLY了。
- Amazon S3 存储桶。模板会创建 Amazon S3 存储dags桶。它被配置为阻止所有公共访问，启用了存储桶版本控制，如中所定义为 [Amazon MWAA 创建 Amazon S3 存储桶 \(p. 16\)](#)。
- 亚马逊 MWAA 环境。该模板创建了一个与 Amazon S3 存储桶上的dags文件夹关联的 Amazon MWAA 环境、一个有权访问 Amazon MWAA 所用AWS服务的执行角色以及使用[AWS自有密钥](#)进行加密的默认角色，如中所定义[创建亚马逊 MWAA 环境 \(p. 28\)](#)。
- CloudWatch 日志。该模板支持 Airflow 调度器日志组、Airflow Web 服务器日志组、Airflow 工作日志组、Airflow DAG 处理日志组和 Airflow 任务日志组在“信息”及更高级别上登录 Apache Air flow 日志，如中所定义[在亚马逊上查看气流日志 CloudWatch \(p. 257\)](#)。CloudWatch

在本教程中，您将完成以下任务：

- 上传并运行 DAG。将支持亚马逊 MWAA 的最新版本 Apache Airflow 的 Apache Airflow 教程 DAG 上传到 Amazon S3，然后在 Apache Airflow 用户界面中运行，如中所定义[添加或更新 DAG \(p. 103\)](#)。
- 查看日志。在日志中查看 Airflow Web 服务器 CloudWatch 日志组，如中所定义[在亚马逊上查看气流日志 CloudWatch \(p. 257\)](#)。
- 创建访问控制策略。在 IAM 中为您的 Apache Airflow 开发团队创建访问控制策略，如中所定义[访问 Amazon MWAA 环境 \(p. 32\)](#)。

先决条件

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下条件：

- [AWS CLI— 安装版本 2。](#)
- [AWS CLI— 使用aws configure. 进行快速配置。](#)

第一步：将AWS CloudFormation模板保存到本地

- 复制以下模板的内容并在本地另存为mwaa_public_network.yaml。您也可以[下载该模板](#)。

```
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: "mwa-a-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}"
```

步骤二：使用创建堆栈AWS CLI

1. 在命令提示符下，导航到“存储mwaa_public_network.yml目录”。例如：


```
cd mwaaproject
```

2. 使用[aws cloudformation create-stack](#)命令使用创建堆栈AWS CLI。

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

Note

创建 Amazon VPC 基础设施、Amazon S3 存储桶和亚马逊 MWAA 环境需要 30 多分钟。

第三步：将 DAG 上传到 Amazon S3 并在 Apache Airflow 用户界面中运行

1. 复制[最新支持的 Apache Airflow 版本tutorial.py](#)的文件内容，并在本地另存为tutorial.py。
2. 在命令提示符下，导航到“存储tutorial.py目录”。例如：

```
cd mwaaproject
```

3. 使用以下命令列出您的所有 Amazon S3 存储桶。

```
aws s3 ls
```

4. 使用以下命令可列出环境中 Amazon S3 存储桶中的文件和文件夹。

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

5. 使用以下脚本将tutorial.py文件上传到您的文件days夹。替换 **YOUR_S3_BUCKET_NAME** 中的示例值。

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

6. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
7. 选择环境。
8. 选择打开 Airflow UI。
9. 在 Apache Airflow 用户界面上，从可用 DAG 列表中选择教程 DAG。
10. 在 DAG 详细信息页面上，选择 DAG 名称旁边的暂停/取消暂停 DAG 开关以取消暂停 DAG。
11. 选择“触发 DAG”。

第四步：在日志中查看 CloudWatch 日志

你可以在 CloudWatch 控制台中查看AWS CloudFormation堆栈启用的所有 Apache Airflow 日志的 Apache Airflow 日志。以下部分显示如何查看 Airflow Web 服务器日志组的日志。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“监控”窗格中选择 Airflow Web 服务器日志组。
4. 在webserver_console_ip日志流中选择日志。

接下来做什么？

- 详细了解如何上传 DAG、在中指定 Python 依赖关系 `requirements.txt` 以及如何在 `aplugins.zip` 中指定自定义插件 [在亚马逊上使用 DAG MWAA \(p. 103\)](#)。
- 详细了解我们推荐用于调整环境性能的最佳实践 [亚马逊 MWAA 上的 Apache Airflow 的性能调整 \(p. 238\)](#)。
- 在中为您的环境创建监控面板 [监控亚马逊 MWAA 上的控制面板和警报 \(p. 260\)](#)。
- 运行中的一些 DAG 代码示例 [Managed Workflows for Airflow \(MWAA\) \(p. 172\)](#)。

开始使用 Amazon Managed Workflows of Apache Airflow (MWAA)

Amazon Managed Workflows of Apache Airflow (MWAA) 使用 Amazon S3 存储桶中的 Amazon VPC、DAG 代码和支持文件创建环境。本指南介绍必备条件和必需AWS开始使用 Amazon MWAA 所需资源。

主题

- [先决条件 \(p. 15\)](#)
- [关于本指南 \(p. 15\)](#)
- [开始前的准备工作 \(p. 15\)](#)
- [可用区 \(p. 16\)](#)
- [为 Amazon MWAA 创建 Amazon S3 存储桶 \(p. 16\)](#)
- [创建 VPC 网络 \(p. 18\)](#)
- [创建亚马逊 MWAA 环境 \(p. 28\)](#)
- [接下来做什么？ \(p. 18\)](#)

先决条件

要创建 Amazon MWAA 环境，您可能需要采取其他步骤以确保您有权访问AWS您需要创建的资源。

- AWS帐户— 一个AWS有权使用 Amazon MWAA 和AWS您的环境使用的服务和资源。

关于本指南

本节介绍AWS您将在本指南中创建的基础架构和资源。

- Amazon VPC— 亚马逊 MWAA 环境所需的 Amazon VPC 联网组件。您可以配置满足这些要求（高级）的现有 VPC，如中所示[关于 Amazon MWAA 上的联网 \(p. 126\)](#)，或者创建 VPC 和网络组件，如中所定义[the section called “创建 VPC 网络” \(p. 18\)](#)。
- Amazon S3 存储桶— 用于存储 DAG 和相关文件的 Amazon S3 存储桶，如plugins.zip和requirements.txt. 必须将 Amazon S3 存储桶配置为阻止所有公有访问，与存储桶版本控制已启用，如中所定义[为 Amazon MWAA 创建 Amazon S3 存储桶 \(p. 16\)](#)。
- 亚马逊 MWAA 环境— 配置了 Amazon S3 存储桶位置、DAG 代码路径和任何自定义插件或 Python 依赖项以及您的 Amazon VPC 及其安全组的 Amazon MWAA 环境，如中所定义[创建亚马逊 MWAA 环境 \(p. 28\)](#)。

开始前的准备工作

要创建 Amazon MWAA 环境，您可能需要采取其他步骤来创建和配置其他环境AWS创建环境之前的资源。

要创建环境，您需要具备以下各项：

- AWS KMS密钥— 一个AWS KMS用于环境中数据加密的密钥。您可以在 Amazon MWAA 控制台上选择默认选项以创建[AWS拥有的密钥](#)当您创建环境时，或者指定现有环境[客户管理的密钥](#)具有其他权限AWS已配置环境使用的服务（高级）。要了解更多信息，请参阅[使用客户管理的密钥进行加密 \(p. 281\)](#)。
- 执行角色— 允许亚马逊 MWAA 访问的执行角色AWS环境中的资源。您可以在 Amazon MWAA 控制台上选择默认选项，以便在创建环境时创建执行角色。要了解更多信息，请参阅[Amazon MWAA 执行角色 \(p. 44\)](#)。
- VPC 安全组— 一个 VPC 安全组，允许 Amazon MWAA 访问其他AWSVPC 网络中的资源。您可以在 Amazon MWAA 控制台上选择默认选项以在创建环境时创建安全组，或者为安全组提供适当的入站和出站规则（高级）。要了解更多信息，请参阅[您在亚马逊 MWAA 上的 VPC 中的安全性 \(p. 129\)](#)。

可用区

在以下各项中提供了 Amazon MWAAAWS地区。

- 欧洲（斯德哥尔摩）– eu-north-1
- 欧洲（法兰克福）– eu-central-1
- 欧洲（爱尔兰）– eu-west-1
- 欧洲（伦敦）– eu-west-2
- 欧洲（巴黎）– eu-west-3
- 亚太地区（孟买）– ap-south-1
- 亚太地区（新加坡）– ap-southeast-1
- 亚太地区（悉尼）– ap-southeast-2
- 亚太地区（东京）– ap-northeast-1
- 亚太地区（首尔）– ap-northeast-2
- 美国东部（弗吉尼亚北部）– us-east-1
- 美国东部（俄亥俄）– us-east-2
- 美国西部（俄勒冈）– us-west-2
- 加拿大（中部）– ca-central-1
- 南美洲（圣保罗）– sa-east-1

为 Amazon MWAA 创建 Amazon S3 存储桶

本指南介绍了创建 Amazon S3 存储桶以将 Apache Airflow 定向非循环图 (DAG)、自定义插件存储在plugins.zip文件和 Python 的依赖关系requirements.txt文件。

目录

- [开始前的准备工作 \(p. 16\)](#)
- [创建存储桶 \(p. 17\)](#)
- [接下来做什么？ \(p. 18\)](#)

开始前的准备工作

- 创建存储桶后，Amazon S3 存储桶名称无法更改。要了解更多信息，请参阅[存储桶命名规则](#)中的Amazon Simple Storage Service 用户指南。
- 必须将用于 Amazon MWAA 环境的 Amazon S3 存储桶配置为阻止所有公有访问，与Bucket versioning已启用。

- 用于 Amazon MWAA 环境的 Amazon S3 存储桶必须位于同一个AWS区域作为 Amazon MWAA 环境。查看列表AWS亚马逊 MWAA 的地区，请参阅[区域可用性 \(p. 4\)](#).

创建存储桶

本节介绍了为您的环境创建 Amazon S3 存储桶的步骤。

创建存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 选择 Create bucket (创建存储桶)。
3. 在 Bucket name (存储桶名称) 中，输入符合 DNS 标准的存储桶名称。

存储桶名称必须满足以下要求：

- 在所有 Amazon S3 中是唯一的。
- 长度必须介于 3 到 63 个字符之间。
- 不包含大写字符。
- 以小写字母或数字开头。

Important

避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

4. 选择一个AWS地区区域. 这一定是相同的AWS区域作为您的 Amazon MWAA 环境。
 - 我们建议选择一个靠近您的区域可最大程度地减少延迟和成本并满足法规要求。
5. 选择 Block all public access (阻止所有公有访问)。
6. 选择启用在Bucket versioning.
7. 可选-标签. 添加键值标签对以在中识别您的 Amazon S3 存储桶标签. 例如，Bucket：Staging.
8. 可选-服务器端加密. 您可以选择性地启用Amazon S3 存储桶上的以下加密选项之一。
 - a. 选择Amazon S3 密钥 (SSE-S3)在服务器端加密为存储桶启用服务器端加密。
 - b. 选择AWS Key Management Servicekey (SSE-KMS)要使用AWS KMSAmazon S3 存储桶上的加密密钥：
 - i. AWS托管密钥 (aws/s3)-如果您选择此选项，则可以使用[AWS拥有的密钥](#)由亚马逊 MWAA 管理，或者指定[客户托管密钥](#)用于对您的 Amazon MWAA 环境进行加密。
 - ii. 从你的AWS KMS密钥要么EnterAWS KMS密钥 ARN-如果你选择指定[客户托管密钥](#)在此步骤中，您必须指定AWS KMS密钥 ID 或 ARN。[AWS KMS亚马逊 MWAA 不支持别名和多区域密钥 \(p. 281\)](#). 这些区域有：AWS KMS您指定的密钥还必须用于 Amazon MWAA 环境上的加密。
9. 可选-高级设置. 如果要启用 Amazon S3 对象锁定：
 - a. 选择高级设置、启用.

Important

启用对象锁定将永久允许锁定此存储桶中的对象。要了解更多信息，请参阅[使用 Amazon S3 对象锁定以锁定对象](#)中的Amazon Simple Storage Service 用户指南。

- b. 选择确认。
10. 选择创建存储桶。

接下来做什么？

- 了解如何在中为环境创建所需的 Amazon VPC 网络[创建 VPC 网络 \(p. 18\)](#).
- 了解如何在中管理访问权限[如何设置 ACL 存储桶权限？](#)
- 了解如何在中删除存储桶[如何删除 S3 存储桶？](#).

创建 VPC 网络

Amazon Managed Workflows for Apache Airflow (MWAA) 中创建环境。指南描述了创建 Amazon VPC ACHE Airflow (MWAA) 环境。

Note

Apache Airflow 在低延迟的网络环境中效果最佳。如果您使用现有的 Amazon VPC 将流量路由到其他区域或本地环境，我们建议为 Amazon SQS、Amazon S3 和 Amazon ECR 添加 AWS PrivateLink 终端节点。CloudWatch AWS KMS 有关配置 Amazon MWAA AWS PrivateLink 的更多信息，请参阅[创建没有互联网访问权限的 Amazon VPC 网络 \(p. 23\)](#)。

目录

- [先决条件 \(p. 18\)](#)
- [开始前的准备工作 \(p. 18\)](#)
- [创建亚马逊 VPC 网络的选项 \(p. 18\)](#)
 - [选项一：在亚马逊 MWAA 控制台上创建 VPC 网络 \(p. 18\)](#)
 - [选项二：创建具有互联网访问权限的 Amazon VPC 网络 \(p. 19\)](#)
 - [选项三：创建没有互联网访问权限的 Amazon VPC 网络 \(p. 23\)](#)
- [接下来做什么？ \(p. 28\)](#)

先决条件

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下满足以下条件：

- [AWS CLI— 安装版本 2。](#)
- [AWS CLI— 使用 aws configure. 进行快速配置。](#)

开始前的准备工作

- 环境创建后无法更改您为环境指定的 [VPC 网络 \(p. 18\)](#)。
- 您可以为亚马逊 VPC 和 Apache Airflow Web 服务器使用私有或公共路由。要查看选项列表，请参阅[the section called “亚马逊 VPC 和 Apache Airflow 访问模式的示例用例” \(p. 128\)](#)。

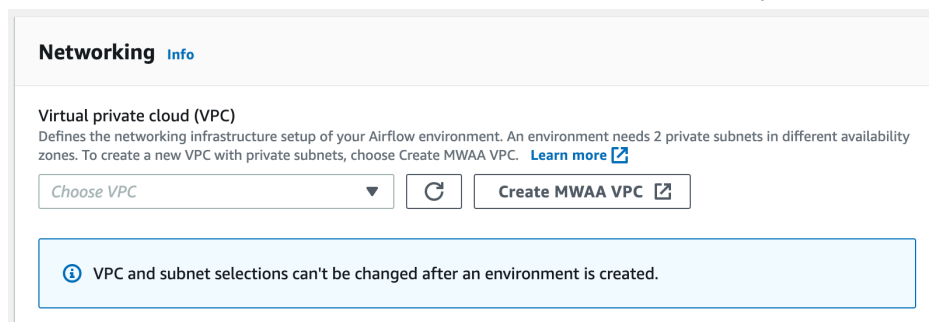
创建亚马逊 VPC 网络的选项

以下部分描述了为环境创建 Amazon VPC 网络的可用选项。

选项一：在亚马逊 MWAA 控制台上创建 VPC 网络

以下部分显示如何在亚马逊 MWAA 控制台上创建 Amazon VPC 网络。此选项使用[通过互联网进行公共路由 \(p. 127\)](#)。它可用于具有专用网络或公共网络访问模式的 Apache Airflow Web 服务器。

下图显示了您可以在亚马逊 MWAA 控制台上找到“创建 MWAA VPC”按钮的位置。



选项二：创建具有互联网访问权限的 Amazon VPC 网络

以下AWS CloudFormation模板在您的默认AWS区域创建了具有互联网访问权限的 Amazon VPC 网络。此选项使用[通过互联网进行公共路由 \(p. 127\)](#)。此模板可用于具有专用网络或公共网络访问模式的 Apache Airflow Web 服务器。

1. 复制以下模板的内容并将其作为cfn-vpc-public-private.yaml。您也可以[下载该模板](#)。

```
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: "mwa-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. 在命令提示符下，导航到存储cfn-vpc-public-private.yaml的目录。例如：

```
cd mwaaproject
```

3. 使用[aws cloudformation create-stack](#)命令使用创建堆栈AWS CLI。

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

Note

创建 Amazon VPC 基础设施需要大约 30 分钟。

选项三：创建没有互联网访问权限的 Amazon VPC 网络

以下AWS CloudFormation模板在您的默认AWS区域创建了一个无法访问互联网的 Amazon VPC 网络。

Important

使用没有互联网访问权限的 Amazon VPC 时，您必须向 Amazon ECR 授予使用网关终端节点访问 Amazon S3 的权限。您可以执行以下操作创建网关终端节点：

1. 复制以下JSON IAM 策略，并在本地将其另存为s3-gw-endpoint-policy.json。该政策授予 Amazon ECR 访问Amazon S3 资源所需的最低权限。

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

2. 使用以下AWS CLI命令创建端点。将--vpc-id和--route-table-ids的值替换为您的 Amazon VPC 的信息。根据您所在的地区替换--service-name为名称。

```
$ 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
```

有关为 Amazon ECR 创建 Amazon S3 网关终端节点的更多信息，请参阅[亚马逊弹性容器注册表用户指南中的创建 Amazon S3 网关终端节点](#)。

此选项使用[没有互联网接入的私人路 \(p. 127\)](#)。此模板只能用于专有网络访问模式的 Apache Airflow Web 服务器。它为[环境使用的AWS服务创建所需的 VPC 终端节点 \(p. 138\)](#)。

1. 复制以下模板的内容并将其作为 `cfn-vpc-private.yaml`。您也可以[下载该模板](#)。

```
AWS::TemplateFormatVersion: "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

S3VpcEndpoint:
  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}-mwa-vpc-endpoints"

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

SqsVpcEndpoint:
  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

CloudWatchLogsVpcEndpoint:
  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
```

```
CloudWatchMonitoringVpcEndpoint:
  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

KmsVpcEndpoint:
  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

EcrApiVpcEndpoint:
  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

EcrDkrVpcEndpoint:
  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

AirflowApiVpcEndpoint:
  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

AirflowEnvVpcEndpoint:
```

```
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

AirflowOpsVpcEndpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    ServiceName: !Sub "com.amazonaws.${AWS::Region}.airflow.ops"
    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. 在命令提示符下，导航到存储cfn-vpc-private.yml的目录。例如：

```
cd mwaaproject
```

3. 使用[aws cloudformation create-stack](#)命令使用创建堆栈AWS CLI。

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

Note

创建 Amazon VPC 基础设施需要大约 30 分钟。

4. 您需要创建一种机制来从您的计算机访问这些 VPC 终端节点。要了解更多信息，请参阅 [在 Amazon MWAA 上管理 VPC 终端节点的访问 \(p. 132\)](#)。

Note

您可以在 Amazon MWAA 安全组的 CIDR 中进一步限制出站访问。例如，您可以通过添加自引用的出站规则、Amazon S3 [的前缀列表](#) 和您的 Amazon VPC 的 CIDR 来对自身进行限制。

接下来做什么？

- 了解如何创建 Amazon MWAA 环境 [创建 Amazon MWAA 环境 \(p. 28\)](#)。
- 了解如何使用私有路由创建从您的计算机到 Amazon VPC 的 VPN 隧道 [教程：使用配置专用网络访问权限 AWS Client VPN \(p. 155\)](#)。

创建 Amazon MWAA 环境

Amazon Apache Airflow 托管工作流程 (MWAA) 使用与 Apache 相同的开源 Airflow 和用户界面，在您选择的版本的环境中设置 Apache Airflow。指南介绍创建 Amazon MWAA 环境的步骤。

目录

- [开始前的准备工作 \(p. 28\)](#)
- [Apache Airflow \(p. 28\)](#)
- [创建环境 \(p. 29\)](#)
 - [步骤一：指定详细信息 \(p. 29\)](#)
 - [步骤二：配置高级设置 \(p. 30\)](#)
 - [步骤三：审核和创建 \(p. 31\)](#)
- [接下来做什么？ \(p. 31\)](#)

开始前的准备工作

- 环境创建后，您无法更改您为环境指定的 [VPC 网络 \(p. 18\)](#) 的值为。
- 您需要将 Amazon S3 存储桶配置为阻止所有公共访问并启用存储桶版本控制。
- 您需要一个 AWS 拥有 [使用 Amazon MWAA 的权限 \(p. 32\)](#) 和在 AWS Identity and Access Management (IAM) 中创建 IAM 角色的权限。如果您为 Apache Airflow Web 服务器选择私有网络访问模式，从而限制 Apache Airflow 在您的亚马逊 VPC 内访问，则需要在 IAM 中获得权限才能创建 Amazon VPC 终端节点。

Apache Airflow

Amazon Managed Workflows for Apache Airflow (MWAA) 中支持以下版本。

Note

- 从 Apache Airflow v2.2.2 开始，亚马逊 MWAA 支持直接在 Apache Airflow Web 服务器上安装 Python 需求、提供程序包和自定义插件。
- 从 Apache Airflow v2.5.1 开始，如果您在使用安装依赖项时未指定该 -c 标志，Amazon MWAA 将使用 requirements.txt 该标志应用一条 --constraint 语句。亚马逊 MWAA 将应用与以下版本表中列出的 Apache Airflow 版本相对应的限制声明。

有关在需求文件 --constraint 中使用的更多信息，请参阅 [安装 Python 依赖项 \(p. 118\)](#)。

Apache Airflow	Apache Airflow	Apache 气流约束	Python 版本
v2.5.1	Apache Airflow v2.5.1 参考指南	Apache Airflow v2.5.1 限制文件	Python 3.10
v2.4.3	Apache Airflow v2.4.3 参考指南	Apache Airflow v2.4.3 限制文件	Python 3.10
v2.2.2	Apache Airflow v2.2.2 参考指南	Apache Airflow v2.2.2 限制文件	Python 3.7
v2.0.2	Apache Airflow v2.0.2 参考指南	Apache Airflow v2.0.2 限制文件	Python 3.7
v1.10.12	Apache Airflow 参考指南	Apache Airflow 约束文件	Python 3.7

有关迁移自我管理的 Apache Airflow 部署或迁移现有 Amazon MWAA 环境的更多信息，包括备份元数据数据库的说明，请参阅 [Amazon MWAA 迁移指南](#)。

创建环境

以下部分介绍创建 Amazon MWAA 环境的步骤。

步骤一：指定详细信息

指定环境的详细信息

1. 打开 [亚马逊 MWAA](#) 控制台。
2. 使用AWS地区选择器选择您的区域。
3. 选择 Create environment (创建环境) 。
4. 在“指定详细信息”页面上，在“环境详细信息”下：
 - a. 在 Name 中键入环境的唯一名称。
 - b. 在气流版本中选择 Apache Airflow 版本。

Note

如果未指定值，则默认为最新版本。可用的最新版本是 Apache Airflow v2.5.1。

5. 在 Amazon S3 的 DAG 代码下，指定以下内容：
 - a. S3 存储桶。选择浏览 S3 并选择您的 Amazon S3 存储桶，或者输入 Amazon S3 URI。
 - b. DAG 文件夹。选择 Browse S3 并选择 Amazon S3 存储桶中的dags文件夹，或输入 Amazon S3 URI。
 - c. 插件文件-可选。选择 Browse S3 并选择您的 Amazon S3 存储桶上的plugins.zip文件，或输入 Amazon S3 URI。
 - d. 需求文件-可选。选择 Browse S3 并选择您的 Amazon S3 存储桶上的requirements.txt文件，或输入 Amazon S3 URI。
 - e. 启动脚本文件-可选，选择“浏览”S3 并选择 Amazon S3 存储桶上的脚本文件，或输入 Amazon S3 URI。
6. 选择下一步。

步骤二：配置高级设置

配置高级设置

1. 在“配置高级设置”页面上，在“网络”下：
 - 选择您的[亚马逊 VPC \(p. 18\)](#)。

此步骤填充您的 Amazon VPC 中的两个私有子网。
2. 在 Web 服务器访问下，选择您首选的 [Apache Airflow 访问模式 \(p. 52\)](#)：
 - a. 私有网络。这会将 Apache Airflow UI 的访问权限限制在您的 Amazon VPC 内被授予[对您的环境的 IAM 策略的 \(p. 32\)](#)访问权限。您需要获得权限才能为此步骤创建 Amazon VPC 终端节点。

Note

如果您的 Apache Airflow 用户界面只能在公司网络内访问，并且无需访问公共存储库即可安装 Web 服务器需求，请选择“专用网络”选项。如果您选择此访问模式选项，则需要创建一种机制来访问亚马逊 VPC 中的 Apache Airflow Web 服务器。有关更多信息，请参阅[访问 Apache Airflow Web 服务器的 VPC 终端节点（专用网络访问） \(p. 135\)](#)：

- b. 公共网络。这允许被授予您环境的 IAM 策略访问权限的用户通过互联网访问 [Apache Airflow \(p. 32\)](#) 用户界面。
3. 在安全组下，选择用于保护您的 [Amazon VPC \(p. 18\)](#) 的安全组：
 - a. 默认情况下，Amazon MWAA 在您的 Amazon VPC 中创建一个安全组，其中包含创建新安全组中的特定入站和出站规则。
 - b. 可选。取消选中“创建新安全组”中的复选框可选择最多 5 个安全组。

Note

必须使用特定的入站和出站规则配置现有 Amazon VPC 安全组才能允许网络流量。要了解更多信息，请参阅[您在亚马逊 MWAA 上的 VPC 中的安全性 \(p. 129\)](#)。

4. 在“环境类”下，选择一个[环境类 \(p. 85\)](#)。
- 我们建议选择支持您的工作负载所需的最小尺寸。您可以随时更改环境类。
5. 对于最大工作器数，请指定要在环境中运行的 Apache Airflow 工作器的最大数量。
- 要了解更多信息，请参阅[高性能用例示例 \(p. 89\)](#)。
6. 在“加密”下，选择数据加密选项：

- a. 默认情况下，Amazon MWAA 使用 AWS 自有密钥来加密您的数据。
- b. 可选。选择“自定义加密设置（高级）”以选择其他 AWS KMS 密钥。如果您选择在此步骤中指定[客户托管密钥](#)，则必须指定 AWS KMS 密钥 ID 或 ARN。[AWS KMS 亚马逊 MWAA 不支持别名和多区域密钥 \(p. 281\)](#)。如果您在 Amazon S3 存储桶上指定了用于服务器端加密的 Amazon S3 密钥，则必须为 Amazon MWAA 环境指定相同的密钥。

Note

您必须拥有密钥的权限才能在 Amazon MWAA 控制台上选择该密钥。您还必须通过附加中所述的策略来授予 Amazon MWAA 使用密钥的权限[附加密钥策略 \(p. 283\)](#)。

7. 推荐。在“监控”下，为 Airflow 日志配置选择一个或多个日志类别，将 Apache Airflow 日志发送到 CloudWatch 日志：
 - a. 气流任务日志。在日志中选择要发送到 Apache Airflow CloudWatch log 任务的类型。
 - b. Airflow Web 服务器日志。选择要发送到 Apache Airflow CloudWatch log 日志的类型。
 - c. 气流调度器日志。选择要发送到 Apache Airflow CloudWatch log 日志的类型。
 - d. 气流工作人员日志。在日志中选择要发送到 Apache Airflow CloudWatch log 日志的类型。

- e. 气流 DAG 处理日志。在日志中选择要发送到 Apache Airflow CloudWatch 日志的类型。
- 8. 可选。对于 Airflow 配置选项，请选择添加自定义配置选项。

您可以从 Apache Airflow 版本的 [Apache Airflow 配置选项 \(p. 90\)](#) 的建议下拉列表中进行选择，也可以指定自定义配置选项。例如，`core.default_task_retries:3`。

- 9. 可选。在标签下，选择添加新标签以将标签关联到您的环境。例如，`Environment:Staging`。
- 10. 在“权限”下，选择一个执行角色：
 - a. 默认情况下，Amazon MWAA 在“创建新角色”中创建执行 (p. 44) 角色。您必须具有创建 IAM 角色的权限。
 - b. 可选。选择输入角色 ARN，输入现有执行角色的 Amazon 资源名称 (ARN)。
- 11. 选择下一步。

步骤三：审核和创建

查看环境摘要

- 查看环境摘要，选择创建环境。

Note

创建环境大约需要二十到三十分钟。

接下来做什么？

- 在中了解如何授予用户访问您的 Apache Airflow Web 服务器和亚马逊 MWAA 环境的权限 [管理对 Amazon MWAA 环境的访问 \(p. 32\)](#)。
- 在中了解如何访问您的 Apache Airflow Web 服务器的 VPC 终端节点（私有网络访问权限） [在 Amazon MWAA 上管理 VPC 终端节点的访问 \(p. 132\)](#)。
- 在上浏览用于创建环境的 Amazon MWAA API 操作 [CreateEnvironment](#)。

接下来做什么？

- 了解如何在创建 Amazon S3 存储桶 [为 Amazon MWAA 创建 Amazon S3 存储桶 \(p. 16\)](#)。

管理对 Amazon MWAA 环境的访问

Amazon Managed Workflows for Apache Airflow (MWAA)，才能使用环境使用的其他AWS服务和资源。您还需要获得访问亚马逊 MWAA 环境和AWS Identity and Access Management (IAM) 中的 Apache Airflow 用户界面的权限。本节介绍用于授予环境AWS资源访问权限的执行角色以及如何添加权限，以及访问 Amazon MWAA 环境和 Apache Airflow UI 所需的AWS账户权限。

主题

- [访问Amazon MWAA 环境 \(p. 32\)](#)
- [Amazon MWAA 的服务相关角色 \(p. 40\)](#)
- [Amazon MWAA 执行角色 \(p. 44\)](#)
- [跨服务混淆代理问题防范 \(p. 51\)](#)
- [Apache Airflow \(p. 52\)](#)

访问Amazon MWAA 环境

要使用 Amazon Managed Workflows for Apache Airflow (MWAA)，您必须使用账户和具有必要权限的 IAM 实体。本页介绍了您可以为Apache Airflow 开发团队和 Apache Airflow 用户附加的访问策略，适用于适用于 Apache Airflow 的亚马逊托管工作流程 (MWAA) 环境。

我们建议使用临时证书并使用群组 and 角色配置联邦身份，以访问您的 Amazon MWAA 资源。作为最佳实践，避免将策略直接附加到您的 IAM 用户，而是定义群组或角色以提供对AWS资源的临时访问权限。

IAM [角色](#)是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 AWS 身份，具有确定其在 AWS 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的[为第三方身份提供商创建角色](#)。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的[权限集](#)。

您可以在您的账户中使用 IAM 角色向另一个授予对您账户的资源的访问AWS 账户权限。有关示例，请参阅 IAM 用户指南中的教程：[跨AWS 账户使用 IAM 角色委托访问权限](#)。

小节目录

- [工作原理 \(p. 33\)](#)
- [完整的主机访问政策：AmazonmWAAPFullConsoleAccess \(p. 33\)](#)
- [完整的 API 和控制台访问政策：AmazonmWAAPFullApiAccess \(p. 36\)](#)
- [只读控制台访问策略：Amazon MWAAReadOnlyAccess \(p. 37\)](#)
- [Apache Airflow UI 访问政策：AmazonmwaWebServerAccess \(p. 38\)](#)
- [Apache Airflow CLI 政策：AmazonmAirflowCliAccess \(p. 38\)](#)
- [创建 JSON 策略 \(p. 38\)](#)

- [将策略附加到开发者群组的示例用例 \(p. 39\)](#)
- [接下来做什么？ \(p. 40\)](#)

工作原理

并非所有AWS Identity and Access Management (IAM) 实体都能访问 Amazon MWAA 环境中使用的资源和服务。您必须创建一个策略，授予 Apache Airflow 用户访问这些资源的权限。例如，您需要向您的 Apache Airflow 开发团队授予访问权限。

Amazon MWAA 使用这些策略来验证用户是否拥有在AWS控制台上或通过环境使用的 API 执行操作所需的权限。

您可以使用本主题中的 JSON 策略为 IAM 中的 Apache Airflow 用户创建策略，然后将该策略附加到 IAM 中的用户、群组或角色。

- [AmazonmWAAFullConsoleAccess \(p. 33\)](#) — 使用此策略授予在亚马逊 MWAA 控制台上配置环境的权限。
- [AmazonmWAAFullApiAccess \(p. 36\)](#) — 使用此政策授予对用于管理环境的所有亚马逊 MWAA API 的访问权限。
- [AmazonmWAAReadOnlyAccess \(p. 37\)](#) — 使用此政策授予在亚马逊 MWAA 控制台上查看环境所用资源的权限。
- [AmazonmwaaWebServerAccess \(p. 38\)](#) — 使用此政策授予对 Apache Airflow 网络服务器的访问权限。
- [AmazonmwaaAirflowCliAccess \(p. 38\)](#) — 使用此策略授予运行 Apache Airflow CLI 命令的访问权限。

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center (successor to AWS Single Sign-On) 中的用户和组：

创建权限集合。按照《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中 [创建权限集](#) 的说明进行操作。

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[为第三方身份提供商创建角色（联合身份验证）](#)的说明进行操作。

- IAM 用户：
 - 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
 - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限（控制台）](#)中的说明进行操作。

完整的主机访问政策： AmazonmWAAFullConsoleAccess

如果用户需要在 Amazon MWAA 控制台上配置环境，则可能需要访问AmazonmWAAFullConsoleAccess权限策略。

Note

您的完整控制台访问策略必须包含执行权限iam:PassRole。这允许用户将[服务相关角色和执行角色 \(p. 44\) \(p. 40\)](#)传递给 Amazon MWAA。Amazon MWAA 担任每个角色是为了代表您调

用其他AWS服务。以下示例使用iam:PassedToService条件键将 Amazon MWAA 服务委托人 (airflow.amazonaws.com) 指定为可将角色传递到的服务。
有关更多信息iam:PassRole，请参阅 IAM 用户指南中的向用户[授予将角色传递给AWS服务的权限](#)。

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

完整的 API 和控制台访问政策： AmazonMWAAPFullApiAccess

如果用户需要访问用于管理环境的所有 Amazon MWAA API，则可能需要访问 AmazonMWAAPFullApiAccess 权限策略。它不授予访问 Apache Airflow UI 的权限。

Note

完整的 API 访问策略必须包含执行权限 `iam:PassRole`。这允许用户将[服务相关角色和执行角色 \(p. 44\) \(p. 40\)](#) 传递给 Amazon MWAA。Amazon MWAA 担任每个角色是为了代表您调用其他 AWS 服务。以下示例使用 `iam:PassedToService` 条件键将 Amazon MWAA 服务委托人 (`airflow.amazonaws.com`) 指定为可将角色传递到的服务。有关更多信息 `iam:PassRole`，请参阅 IAM 用户指南中的向用户[授予将角色传递给 AWS 服务的权限](#)。

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

只读控制台访问策略：Amazon MWAAReadOnlyAccess

如果用户需要在 Amazon MWAA 控制台环境详细信息页面上查看环境使用的资源，则可能需要访问 AmazonMWAAReadOnlyAccess 权限策略。它不允许用户创建新环境、编辑现有环境或允许用户查看 Apache Airflow 用户界面。

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

```
}
```

Apache Airflow UI 访问政策 : AmazonmwaWebServerAccess

如果用户需要访问 Apache Airflow UI，则可能需要访问AmazonMWAWebServerAccess权限策略。它不允许用户在亚马逊 MWAA 控制台上查看环境或使用亚马逊 MWAA API 执行任何操作。指定AdminOp、User、Viewer或中的Public角色{airflow-role}以自定义 Web 令牌用户的访问级别。有关更多信息，请参阅 Apache Airflow 参考指南中的[默认角色](#)。

```
{
  "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 MWAA 提供 IAM 与五个[默认 Apache Airflow \(RBAC\) 角色](#)的集成。有关使用自定义 Apache Airflow 角色的更多信息，请参阅[the section called “教程：限制用户访问 DAG 的子集” \(p. 169\)](#)。

Apache Airflow CLI 政策 : AmazonmAiirflowCliAccess

如果用户需要运行 Apache Airflow CLI 命令（例如trigger_dag），则可能需要访问AmazonMWAAAirflowCliAccess权限策略。它不允许用户在亚马逊 MWAA 控制台上查看环境或使用亚马逊 MWAA API 执行任何操作。

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

创建 JSON 策略

您可以创建 JSON 策略，并将策略附加到您的 IAM 控制台上的用户、角色或组。以下步骤介绍如何在 IAM 中创建 JSon 策略。

创建 JSON 策略

1. 在 IAM 控制台上打开 Policies ([策略](#)) 页面。

2. 选择 Create policy (创建策略)。
3. 请选择 JSON 选项卡。
4. 添加您的 JSON 策略。
5. 选择 Review policy (查看策略)。
6. 在名称和描述的文本字段中输入一个值 (可选)。

例如，您可以将策略命名 AmazonMWAAReadOnlyAccess。

7. 选择 Create policy (创建策略)。

将策略附加到开发者群组的示例用例

假设你在 IAM 中使用一个名为的群组 AirflowDevelopmentGroup 向 Apache Airflow 开发团队的所有开发人员应用权限。这些用户需要访问 AmazonMWAAReadOnlyAccess、AmazonMWAAReadOnlyAccess、AmazonMWAAReadOnlyAccess、和 AmazonMWAAReadOnlyAccess 权限策略。本节介绍如何在 IAM 中创建群组、创建和附加这些策略以及如何将群组关联到 IAM 用户。这些步骤假设你使用的是 [AWS 自有密钥](#)。

创建 AmazonMWAAReadOnlyAccess 策略

1. 下载 A [amazonmwaaroadonlyaccess 访问政策](#)。
2. 在 IAM 控制台上打开 Policies ([策略](#)) 页面。
3. 选择 Create policy (创建策略)。
4. 请选择 JSON 选项卡。
5. 将 JSON 策略粘贴到 AmazonMWAARoadOnlyAccess。
6. 替换以下值：
 - a. `{your-account-id}` — 您的 AWS 账户 ID (例如 0123456789)
 - b. `{your-kms-id}` — [AWS 自有密钥](#) 的标识符
7. 选择审查政策。
8. 键 AmazonMWAARoadOnlyAccess 入名称。
9. 选择 Create policy (创建策略)。

创建 AmazonMWAARoadOnlyAccess 策略

1. 下载 A [amazonmwaaroadonlyaccess 访问政策](#)。
2. 在 IAM 控制台上打开 Policies ([策略](#)) 页面。
3. 选择 Create policy (创建策略)。
4. 请选择 JSON 选项卡。
5. 将 JSON 策略粘贴到 AmazonMWAARoadOnlyAccess。
6. 替换以下值：
 - a. `{####}` — 您的亚马逊 MWAA 环境所在的区域 (例如 us-east-1)
 - b. `{your-account-id}` — 您的 AWS 账户 ID (例如 0123456789)
 - c. `{your-environment-name}` — 您的亚马逊 MWAA 环境名称 (例如 MyAirflowEnvironment)
 - d. `{airflow-role}` — Admin Apache Airflow [默认角色](#)
7. 选择 Review policy (查看策略)。
8. 键 AmazonMWAARoadOnlyAccess 入名称。
9. 选择 Create policy (创建策略)。

创建 AmazonMWAAirflowCliAccess 策略

1. 下载 A [mazonmwaaAirflowCliAccess 访问政策](#)。
2. 在 IAM 控制台上打开 Policies ([策略](#)) 页面。
3. 选择 Create policy (创建策略) 。
4. 请选择 JSON 选项卡。
5. 将 JSON 策略粘贴到AmazonMWAAirflowCliAccess。
6. 选择审查政策。
7. 键AmazonMWAAirflowCliAccess入名称。
8. 选择 Create policy (创建策略) 。

创建群组

1. 在 IAM 控制台上打开[群组页面](#)。
2. 键入的名称AirflowDevelopmentGroup。
3. 选择 Next Step。
4. 在“筛AmazonMWAA选器”中键入筛选结果。
5. 选择您已创建的三个策略。
6. 选择 Next Step。
7. 选择 Create Group (创建组) 。

与用户关联

1. 在 IAM 控制台上打开“[用户](#)” 页面。
2. 选择用户。
3. 选择 Groups (组) 。
4. 选择“将用户添加到群组”。
5. 选择AirflowDevelopmentGroup。
6. 然后选择 Add to Groups (添加到组) 。

接下来做什么？

- 了解如何在生成令牌以访问 Apache Airflow UI[访问 Apache Airflow界面 \(p. 58\)](#)。
- 在创建 IAM 策略中了解有关[创建 IAM 策略](#)的更多信息。

Amazon MWAA 的服务相关角色

Amazon Managed Workflows for Apache AirflowAWS Identity and Access Management (MWAA) 使用[服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 Amazon MWAA 直接相关。服务相关角色是由 Amazon MWAA 预定义的，并包含服务代表您调用其他AWS服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon MWAA，因为您不必手动添加必要的权限。Amazon MWAA 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon MWAA 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon MWAA 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅[与 IAM 配合使用的 AWS 服务](#)，并查找 Service-linked roles (服务相关角色) 列中显示为 Yes (是) 的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon MWAA 的服务相关角色权限

Amazon MWAA 使用名为的服务相关角色AWSRoleForAmazonMWAA — 在您的账户中创建的服务相关角色会授予 Amazon MWAA 对以下AWS服务的访问权限：

- 亚马逊 CloudWatch 日志 (CloudWatch 日志) — 为 Apache Airflow 日志创建日志组。
- 亚马逊 CloudWatch (CloudWatch) — 向您的账户发布与您的环境及其基础组件相关的指标。
- 亚马逊Elastic Compute Cloud (Amazon EC2) — 创建以下资源：
 - 您的 VPC 中的亚马逊 VPC 终端节点，用于AWS托管的 Amazon Aurora PostgreSQL 数据库集群，供 Apache Airflow Scheduler 和 Worker
 - 如果您为 Apache Airflow Web 服务器选择[私有网络 \(p. 52\)](#)选项，则还有一个 Amazon VPC 终端节点，用于启用对 Web 服务器的网络访问。
 - 您@@ [的 Amazon VPC 中的弹性网络接口 \(ENI\)](#) 可通过网络访问您的 Amazon VPC 中托管的AWS资源。

以下信任策略允许服务委托人担任服务相关角色。亚马逊 MWAA 的服务主体airflow.amazonaws.com如该政策所示。

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

名为的角色权限策略AmazonMWAAServiceRolePolicy允许 Amazon MWAA 对指定资源完成以下操作：

```
{
  "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"
            ]
        }
    }
}

```

```
}  
  }  
}  
}
```

必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的[服务相关角色权限](#)。

为 Amazon MAA 创建服务相关角色

无需手动创建服务相关角色。当您使用 AWS Management Console、或 AWS API 创建新的 Amazon MWAA 环境时，Amazon MWAA 会为您创建服务相关角色。AWS CLI

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。当您创建另一个环境时，Amazon MWAA 会再次为您创建服务相关角色。

为 Amazon MAA 编辑服务相关角色

Amazon MWAA 不允许您编辑 AWSServiceRoleForAmazonMWAA 服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅 IAM 用户指南中的[编辑服务相关角色](#)。

删除适用于 Amazon MAA 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。

当您删除 Amazon MWAA 环境时，Amazon MWAA 会删除它在服务中使用的所有关联资源。但是，您必须等待 Amazon MWAA 完成对环境的删除后，才能尝试删除服务相关角色。如果您在 Amazon MWAA 删除环境之前删除服务相关角色，则 Amazon MWAA 可能无法删除该环境的所有关联资源。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台，即 AWS CLI 或 AWS API 来删除 AWSServiceRoleForAmazonMWAA 服务相关角色。有关更多信息，请参阅 IAM 用户指南中的[删除服务相关角色](#)。

Amazon MWAA 服务相关角色支持的区域

Amazon MWAA 支持在该服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 [Amazon Managed Workflows for Apache Airflow \(MWAA\) 端点](#)

策略更新

更改	说明	日期
Amazon MWAA 更新其服务相关角色策略	AmazonMWAAServiceRolePolicy 2022 年 11 月 18 日 Amazon MWAA 更新了其服务相关角色的权限策略，以授予 Amazon MWAA 向客户账户发布与服务基础资源相关的其他指标的权限。这些新指标发布在 AWS/MWAA	2022 年 11 月 18 日
Amazon MWAA 已开启跟踪更改	Amazon MWAA 开始跟踪其 Managed AWS service 相关角色权限策略的更改。	2022 年 11 月 18 日

Amazon MWAA 执行角色

执行角色是具有权限策略的AWS Identity and Access Management (IAM) 角色，该策略授予 Amazon Managed Workflow for Airflow (MWAA) 代表您调用其他AWS服务资源的权限。这可能包括您的 Amazon S3 存储桶、[AWS自有密钥](#)和 CloudWatch 日志等资源。Amazon MWAA 环境需要每个环境一个执行角色。本页介绍如何使用和配置环境的执行角色，以允许 Amazon MWAA 访问您的环境使用的其他AWS资源。

目录

- [执行角色概述 \(p. 44\)](#)
 - [默认情况下附加权限 \(p. 45\)](#)
 - [如何添加使用其他AWS服务的权限 \(p. 45\)](#)
 - [如何关联新的执行角色 \(p. 45\)](#)
- [创建新角色 \(p. 45\)](#)
- [查看和更新执行角色策略 \(p. 46\)](#)
 - [附加 JSON 策略以使用其他AWS服务 \(p. 46\)](#)
- [使用账户级公有访问区块授予对 Amazon S3 存储桶的访问权限 \(p. 47\)](#)
- [使用 Apache 气流连接 \(p. 47\)](#)
- [执行角色的 JSON 策略示例 \(p. 47\)](#)
 - [客户托管密钥的策略示例 \(p. 47\)](#)
 - [AWS自有密钥的政策示例 \(p. 50\)](#)
- [接下来做什么？ \(p. 51\)](#)

执行角色概述

Amazon MWAA 使用您的环境使用的其他AWS服务的权限是从执行角色获得的。Amazon MWAA 执行角色需要获得环境使用的以下AWS服务的权限：

- 亚马逊 CloudWatch (CloudWatch) — 发送 Apache Airflow 指标和日志。
- Amazon Simple Storage Service (Amazon S3) — 用于解析您环境的 DAG 代码和支持文件（例如 `arequirements.txt`）。
- Amazon Simple Queue Service (Amazon SQS) — 将您环境的 Apache Airflow (Amazon SQS)，将您环境的 Apache Air
- AWS Key Management Service(AWS KMS) — 用于环境的数据加密（使用[AWS自有密钥](#)或[客户管理的密钥](#)）。

Note

如果您选择让 Amazon MWAA 使用AWS托管 KMS 密钥加密您的数据，则必须在附加到您的 Amazon MWAA 执行角色的策略中定义权限，该策略允许通过 Amazon SQS 访问存储在您的账户外的任意 KMS 密钥。要使环境的执行角色访问任意 KMS 密钥，需要满足以下两个条件：

- 第三方账户中的 KMS 密钥需要通过其资源策略允许这种跨账户访问。
- 您的 DAG 代码需要访问以第三方账户开头的 Amazon SQS 队列，使用相同的 KMS 密钥进行加密。airflow-celery-

为了降低与跨账户访问资源相关的风险，我们建议您查看放置在 DAG 中的代码，确保您的工作流程不会访问账户之外的任意 Amazon SQS 队列。此外，您可以使用存储在您自己账户中的客户管理的 KMS 密钥来管理 Amazon MWAA 上的加密。这将您的环境的执行角色限制为仅访问账户中的 KMS 密钥。

请记住，选择加密选项后，您无法更改现有环境的选择。

执行角色还需要获得以下 IAM 操作的权限：

- `airflow:PublishMetrics`— 允许亚马逊 MWAA 监控环境的运行状况。

默认情况下附加权限

您可以使用 Amazon MWAA 控制台上的默认选项来创建执行角色和[AWS 自有密钥](#)，然后使用此页面上的步骤为您的执行角色添加权限策略。

- 当您在控制台上选择“创建新角色”选项时，Amazon MWAA 会将环境所需的最低权限附加到您的执行角色。
- 在某些情况下，亚马逊 MWAA 会附加最大权限。例如，我们建议您在创建环境时选择 Amazon MWAA 控制台上创建执行角色的选项。Amazon MWAA 使用执行角色中的正则表达式模式自动为所有 CloudWatch 日志组添加权限策略“`arn:aws:logs:your-region:your-account-id:log-group:airflow-your-environment-name-*`”。

如何添加使用其他AWS服务的权限

创建环境后，Amazon MWAA 无法向现有执行角色添加或编辑权限策略。您必须使用环境所需的其他权限策略更新您的执行角色。例如，如果您的 DAG 需要访问权限 AWS Glue，则 Amazon MWAA 无法自动检测您的环境需要这些权限，也无法向您的执行角色添加权限。

您可以通过两种方式向执行角色添加权限：

- 通过内联修改执行角色的 JSON 策略。您可以使用此页面上的示例[JSON 策略文档](#)在 IAM 控制台上添加或替换您的执行角色的 JSON 策略。
- 通过为 AWS 服务创建 JSON 策略并将其附加到您的执行角色。您可以使用此页面上的步骤将 AWS 服务的新的 JSON 策略文档关联到您在 IAM 控制台上的执行角色。

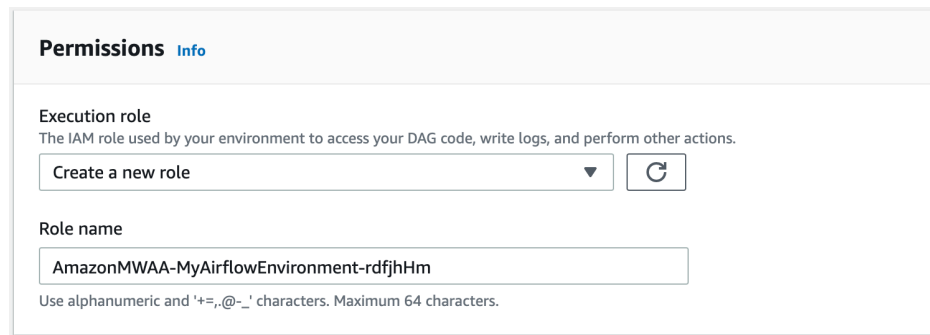
假设执行角色已经与您的环境相关联，Amazon MWAA 可以立即开始使用添加的权限策略。这也意味着，如果您从执行角色中删除任何必需的权限，则您的 DAG 可能会失败。

如何关联新的执行角色

您可以随时更改环境的执行角色。如果新的执行角色尚未与您的环境关联，请使用本页上的步骤创建新的执行角色策略，并将该角色关联到您的环境。

创建新角色

默认情况下，Amazon MWAA 会为您创建用于数据加密的[AWS 自有密钥](#)和执行角色。创建环境时，您可以在 Amazon MWAA 控制台上选择默认选项。下图显示了为环境创建执行角色的默认选项。



The screenshot shows the 'Permissions' section of the Amazon MWAA console. It includes a header 'Permissions Info', a description of the 'Execution role' as 'The IAM role used by your environment to access your DAG code, write logs, and perform other actions.', a dropdown menu set to 'Create a new role' with a refresh button, a text field for 'Role name' containing 'AmazonMWAA-MyAirflowEnvironment-rdfjhHm', and a note: 'Use alphanumeric and '+,=, @, _' characters. Maximum 64 characters.'

查看和更新执行角色策略

您可以在 Amazon MWAA 控制台上查看您的环境的执行角色，并在 IAM 控制台上更新该角色的 JSON 策略。

更新执行角色策略

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“权限”窗格中选择执行角色以在 IAM 中打开权限页面。
4. 选择执行角色名称以打开权限策略。
5. 选择编辑策略。
6. 请选择 JSON 选项卡。
7. 更新您的 JSON 策略。
8. 选择 Review policy (查看策略) 。
9. 选择保存更改。

附加 JSON 策略以使用其他AWS服务

您可以为AWS服务创建 JSON 策略并将其附加到您的执行角色。例如，您可以附加以下 JSON 策略以向中的所有资源授予只读访问权限AWS Secrets Manager。

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

向执行角色附加策略

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“权限”窗格中选择您的执行角色。
4. 选择 Attach policies (附上策略) 。
5. 选择 Create policy (创建策略) 。
6. 选择 JSON。
7. 粘贴 JSON 策略。
8. 选择“下一步：标签”，选择“下一步：查看”。
9. 输入策略的描述性名称 (例如SecretsManagerReadPolicy) 和描述。

10. 选择 Create policy (创建策略)。

使用账户级公有访问区块授予对 Amazon S3 存储桶的访问权限

您可能需要使用 A [PutPublicAccessBlock](#)amazon S3 操作来阻止对账户中所有存储段的访问。当您禁止访问账户中的所有存储段时，您的环境执行角色必须在权限策略中包含该s3:GetAccountPublicAccessBlock操作。

以下示例演示了在阻止访问您账户中的所有 Amazon S3 存储桶时必须为执行角色附加的策略。

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

有关限制访问 Amazon S3 存储桶的详细信息，请参阅 Amazon [Simple Storage Service 用户指南](#)中的 [Amazon S3 存储桶的公开访问](#)。

使用 Apache 气流连接

您还可以创建 Apache Airflow 连接，并在 Apache Airflow 连接对象中指定您的执行角色及其 ARN。要了解更多信息，请参阅 [管理与 Apache Airflow 的连接 \(p. 73\)](#)。

执行角色的 JSON 策略示例

本节中的示例权限策略显示了两个策略，您可以使用它们来替换用于现有执行角色的权限策略，或者创建新的执行角色并在您的环境中使用。这些策略包含 Apache Airflow 日志组的 [资源 ARN](#) 占位符、[Amazon S3 存储桶 \(p. 16\)](#)和[亚马逊 MWAA 环境 \(p. 28\)](#)。

我们建议复制示例策略，替换示例 ARN 或占位符，然后使用 JSON 策略创建或更新执行角色。例如，替换{your-region}为us-east-1。

客户托管密钥的策略示例

以下示例显示了可用于[客户管理密钥](#)的执行角色策略。

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

接下来，您需要允许 Amazon MWAA 担任此角色以代表您执行操作。这可以通过以下方式实现：使用 [IAM 控制台](#) 将 `airflow-env.amazonaws.com` 服务主体添加到 `airflow.amazonaws.com` 该执行角色的可信实体列表中，或者使用 `IAM create-role` 命令将这些服务主体置于该执行角色的代入角色策略文档中 AWS CLI。可以在下面找到代入角色策略文档的示例：

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

然后将以下 JSON 策略附加到您的 [客户托管密钥](#)。此策略使用 [kms:EncryptionContext](#) 条件key prefix 来允许访问日志中的 Apache Airflow CloudWatch 日志组。

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

AWS自有密钥的政策示例

以下示例显示了可用于[AWS自有密钥](#)的执行角色策略。

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

接下来做什么？

- 了解您和您的 Apache Airflow 用户访问您的环境所需的权限 [访问Amazon MWAA 环境 \(p. 32\)](#)。
- 了解 [使用客户管理的密钥进行加密 \(p. 281\)](#)。
- 浏览更多 [客户托管策略示例](#)。

跨服务混淆代理问题防范

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 AWS 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的“服务”）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为了防止这种情况，AWS 提供可帮助您保护所有服务的委托数据工具，这些服务委托人有权限访问账户中的资源。

我们建议在您的环境的执行角色中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文密钥，以限制 Amazon MWAA 授予其他服务访问资源的权限。如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:airflow:*:123456789012:environment/*`。

的值aws:SourceArn必须是您的 Amazon MWAA 环境 ARN，您正在为其创建执行角色。

以下示例演示如何在环境的执行角色信任策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文键来防范混淆代理问题。在创建新的执行角色时，可以使用以下信任策略。

```
{
  "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

适用于 Apache Airflow 的 Amazon 托管工作流程 (MWAA) 控制台包含内置选项，用于在您的环境中配置指向 Apache Airflow Web 服务器的私有或公共路由。本指南描述了 Apache Airflow 的 Amazon Managed Workflow (MWAA) 环境上可用的 Apache Airflow Web 服务器的访问模式，以及如果您选择私有网络选项，则需要在亚马逊 VPC 中配置的其他资源。

目录

- [Apache Airflow \(p. 52\)](#)
 - [公共网络 \(p. 53\)](#)
 - [私有网络 \(p. 54\)](#)
- [访问模式概述 \(p. 56\)](#)
 - [公网访问模式 \(p. 56\)](#)
 - [私有网络访问模式 \(p. 56\)](#)
- [私有和公共访问模式的设置 \(p. 56\)](#)
 - [公共网络设置 \(p. 56\)](#)
 - [私有网络设置 \(p. 57\)](#)
- [访问您的 Apache Airflow Web 服务器的 VPC 终端节点 \(私有网络访问权限\) \(p. 57\)](#)

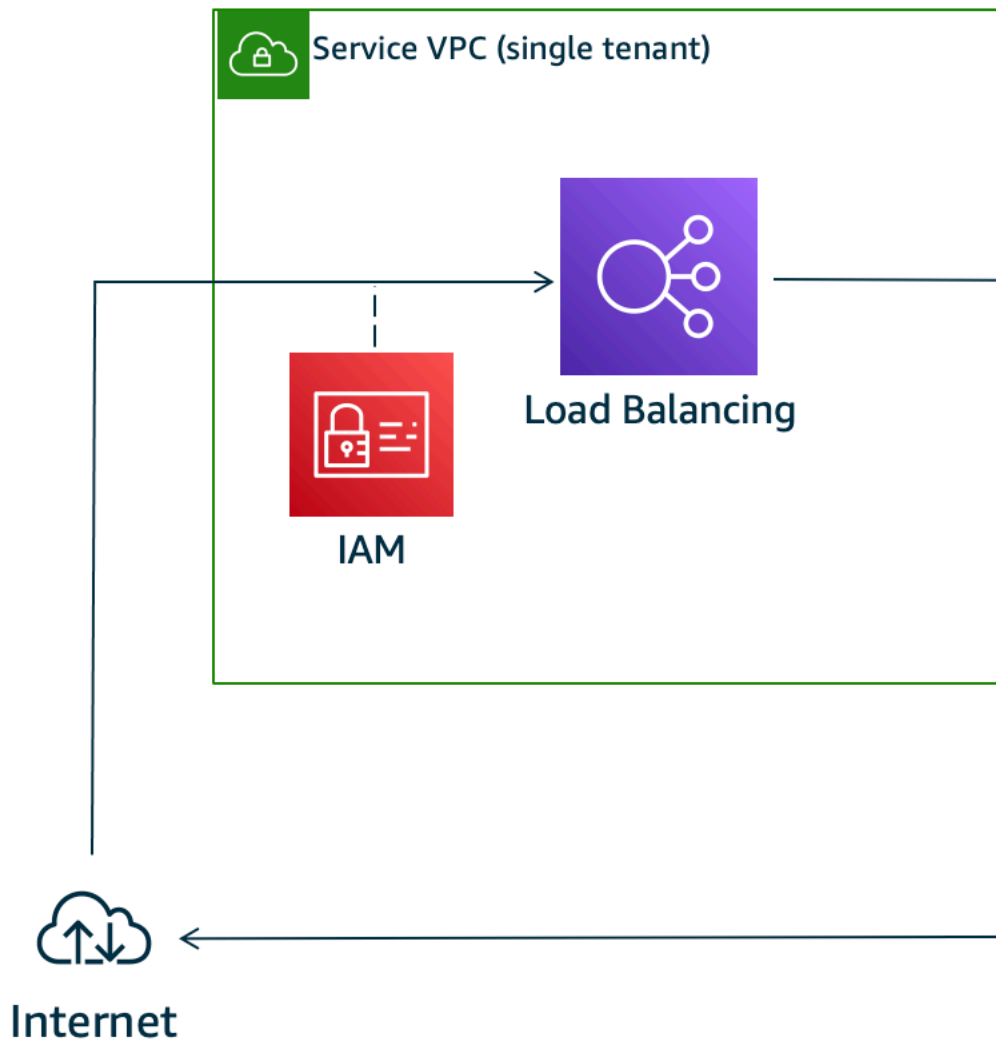
Apache Airflow

你可以为 Apache Airflow Web 服务器选择私有或公共路由。要启用私有路由，请选择专用网络。这将用户对 Apache Airflow Web 服务器的访问限制在亚马逊 VPC 内。要启用公共路由，请选择公共网络。这允许用户通过互联网访问 Apache Air flow Web 服务器。

公共网络

以下架构图显示了带有公共 Web 服务器的 Amazon MWAA 环境。

Public Web Server Opti



公共网络访问模式允许被授予[您环境的 IAM 策略 \(p. 32\)](#)访问权限的用户通过互联网访问 Apache Airflow UI。

下图显示了在 Amazon MWAA 控制台上何处可以找到公共网络选项。

Web server access

☐ **Private network (Recommended)**

Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.

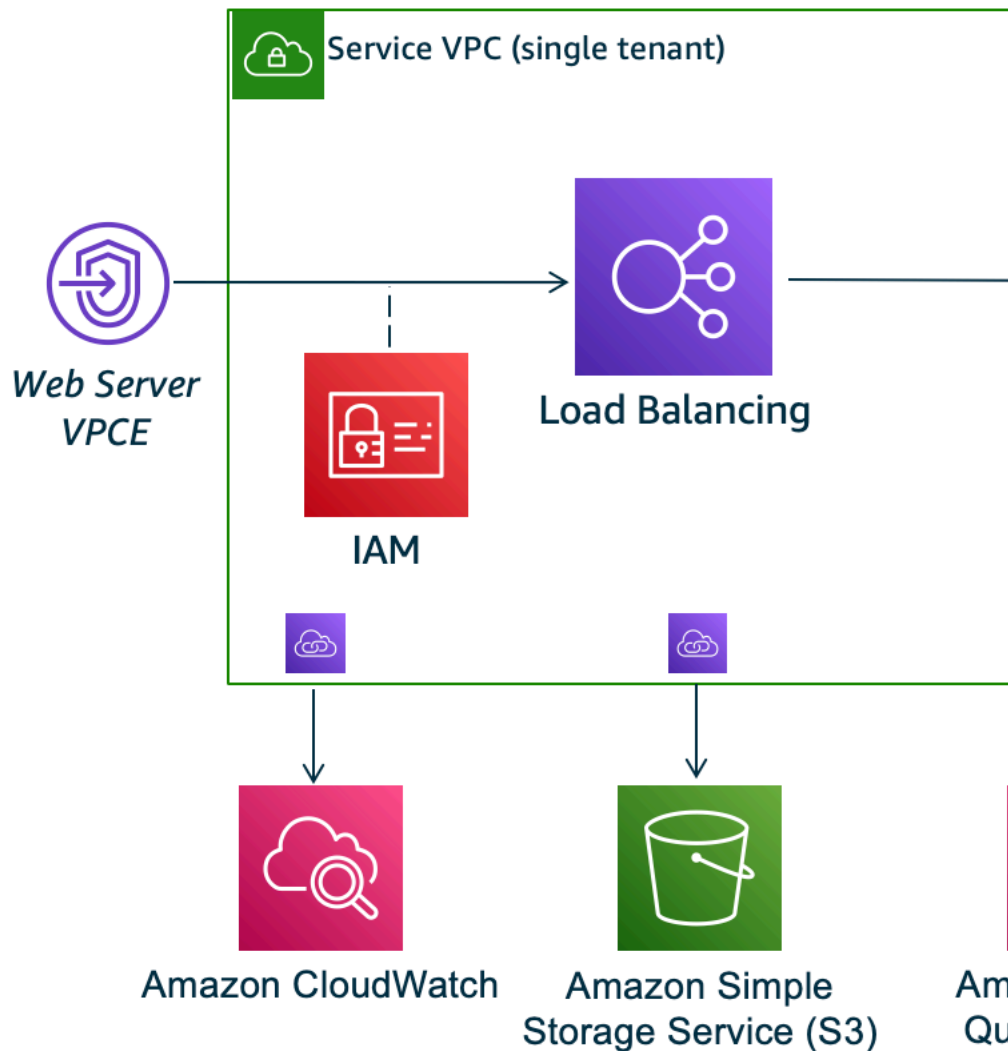
☒ **Public network (No additional setup)**

Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

私有网络

以下架构图显示了带有私有 Web 服务器的 Amazon MWAA 环境。

Private Web Server Opt



私有网络访问模式将对 Apache Airflow UI 的访问权限限制在您的 Amazon VPC 中，这些用户已被授予[对您环境的 IAM 策略的 \(p. 32\)](#)访问权限。

当您创建具有私有 Web 服务器访问权限的环境时，必须将所有依赖项打包到 Python wheel 存档中 (.whl)，然后在 .whl 中引用 requirements.txt。有关使用 wheel 打包和安装依赖项的说明，请参阅[使用 Python 轮子管理依赖关系 \(p. 250\)](#)。

下图显示了在 Amazon MWAA 控制台上可以找到私有网络选项的位置。

Web server access

☒ **Private network (Recommended)**

Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.

☐ **Public network (No additional setup)**

Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

访问模式概述

本节介绍当您选择公用网络或私有网络访问模式时在您的 Amazon VPC 中创建的 VPC 终端节点 (AWS PrivateLink)。

公网访问模式

如果您为 Apache Airflow Web 服务器选择公共网络访问模式，则网络流量将通过互联网公开路由。

- 亚马逊 MWAA 为您的 Amazon Aurora PostgreSQL 元数据数据库创建 VPC 接口终端节点。终端节点是在映射至您的私有子网的可用区中创建的，并且独立于其他 AWS 账户。
- 然后，Amazon MWAA 将您的私有子网的 IP 地址绑定到接口终端节点。这旨在支持从 Amazon VPC 的每个可用区绑定单个 IP 的最佳实践。

私有网络访问模式

如果您为 Apache Airflow Web 服务器选择私有网络访问模式，则网络流量将在您的 Amazon VPC 内以私有方式路由。

- 亚马逊 MWAA 为您的 Apache Airflow Web 服务器创建 VPC 接口终端节点，为 Amazon Aurora PostgreSQL 元数据数据库创建接口终端节点。终端节点是在映射到您的私有子网的可用区中创建的，独立于其他 AWS 账户。
- 然后，Amazon MWAA 将您的私有子网的 IP 地址绑定到接口终端节点。这旨在支持从 Amazon VPC 的每个可用区绑定单个 IP 的最佳实践。

要了解更多信息，请参阅 [the section called “亚马逊 VPC 和 Apache Airflow 访问模式的示例用例” \(p. 128\)](#)。

私有和公共访问模式的设置

下一节根据您为环境选择的 Apache Airflow 访问模式描述了您需要的其他设置和配置。

公共网络设置

如果您为 Apache Airflow Web 服务器选择公共网络选项，则可以在创建环境后开始使用 Apache Airflow 用户界面。

您需要采取以下步骤来配置用户的访问权限，并为您的环境配置使用其他 AWS 服务的权限。

1. 添加权限。亚马逊 MWAA 需要许可才能使用其他 AWS 服务。当您创建环境时，Amazon MWAA 会创建一个 [服务相关角色 \(p. 40\)](#)，允许它对亚马逊弹性容器注册表 (Amazon ECR)、CloudWatch 日志和 Amazon EC2 使用某些 IAM 操作。

您可以添加对这些服务使用其他操作的权限，也可以通过向您的执行角色添加权限来使用其他 AWS 服务。要了解更多信息，请参阅 [Amazon MWAA 执行角色 \(p. 44\)](#)。

2. 创建用户策略。您可能需要为用户创建多个 IAM 策略来配置对您的环境和 Apache Airflow UI 的访问权限。要了解更多信息，请参阅 [访问Amazon MWAA 环境 \(p. 32\)](#)。

私有网络设置

如果您为 Apache Airflow Web 服务器选择私有网络选项，则需要为用户配置访问权限，为环境配置使用其他 AWS 服务的权限，并创建从计算机访问 Amazon VPC 中的资源的机制。

1. 添加权限。亚马逊 MWAA 需要许可才能使用其他 AWS 服务。当您创建环境时，Amazon MWAA 会创建一个 [服务相关角色 \(p. 40\)](#)，允许它对亚马逊弹性容器注册表 (Amazon ECR)、CloudWatch 日志和 Amazon EC2 使用某些 IAM 操作。

您可以添加对这些服务使用其他操作的权限，也可以通过向您的执行角色添加权限来使用其他 AWS 服务。要了解更多信息，请参阅 [Amazon MWAA 执行角色 \(p. 44\)](#)。

2. 创建用户策略。您可能需要为用户创建多个 IAM 策略来配置对您的环境和 Apache Airflow UI 的访问权限。要了解更多信息，请参阅 [访问Amazon MWAA 环境 \(p. 32\)](#)。
3. 启用网络访问权限。您需要在亚马逊 VPC 中创建一个机制来连接到 Apache Airflow Web 服务器的 VPC 终端节点 (AWS PrivateLink)。例如，通过使用从您的计算机创建 VPN 隧道 AWS Client VPN。

访问您的 Apache Airflow Web 服务器的 VPC 终端节点 (私有网络访问权限)

如果您选择了私有网络选项，则需要在您的亚马逊 VPC 中创建一个机制来访问 Apache Airflow Web 服务器的 VPC 终端节点 (AWS PrivateLink)。我们建议对这些资源使用与 Amazon MWAA 环境相同的 Amazon VPC、VPC 安全组和私有子网。

要了解更多信息，请参阅 [管理 VPC 终端节点的访问权限](#)。

访问 Apache Airflow 界面

创建 Amazon Managed Workflows for Apache Airflow UI 您可以使用 Amazon MWAA 控制台在 Apache Airflow UI 中查看和调用 DAG，也可以使用 Amazon MWAA API 获取令牌并调用 DAG。本节介绍访问 Apache Airflow 用户界面所需的权限、如何生成令牌以直接在命令外壳中调用 Amazon MWAA API 以及 Apache Airflow CLI 中支持的命令。

主题

- [先决条件 \(p. 58\)](#)
- [打开 Airflow \(p. 58\)](#)
- [登录 Apache Airfl \(p. 58\)](#)
- [创建 Apache Airflow Web 登录令牌 \(p. 59\)](#)
- [创建 Apache 气流 CLI 令牌 \(p. 61\)](#)
- [Apache Airflow Air \(p. 66\)](#)

先决条件

下一节描述了使用本节中的命令和脚本所需的初步步骤。

访问

- AWS Identity and Access Management(IAM) 中的账户访问权限中的亚马逊 MWAA 权限策略 [Apache Airflow UI 访问策略 : AmazonmwaWebServerAccess \(p. 38\)](#)。
- AWS Identity and Access Management(IAM) 中对亚马逊 MWAA 权限策略的账户访问权限 [完整的 API 和控制台访问策略 : AmazonmWAAPullAccess \(p. 36\)](#)。

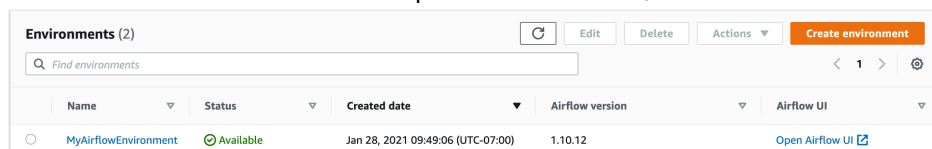
AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下条件：

- [AWS CLI— 安装版。](#)
- [AWS CLI— 使用aws configure. 进行快速配置。](#)

打开 Airflow

下图显示了亚马逊 MWAA 主机上的 Apache Airflow 用户界面的链接。



Environments (2)					
Find environments					
Name	Status	Created date	Airflow version	Airflow UI	
MyAirflowEnvironment	Available	Jan 28, 2021 09:49:06 (UTC-07:00)	1.10.12	Open Airflow UI	

登录 Apache Airfl

你需要在AWS Identity and Access Management (IAM) 中拥有AWS账户的 [Apache Airflow UI 访问策略 : AmazonmwaWebServerAccess \(p. 38\)](#) 权限才能查看你的 Apache Airflow 用户界面。

访问您的 Airflow UI 额

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择打开 Airflow UI。

创建 Apache Airflow Web 登录令牌

您可以使用此页面上的命令生成 Web 登录令牌，然后直接在命令外壳中进行 Apache Airflow (MWAA) API 的亚马逊托管工作流程调用。例如，您可以获取令牌，然后使用 Amazon MWAA API 以编程方式部署 DAG。以下部分包括使用创建 Apache Airflow Web 登录令牌的步骤AWS CLI、bash 脚本、POST API 请求或 Python 脚本。响应中返回的令牌在 60 秒内有效。

目录

- [先决条件 \(p. 59\)](#)
 - [访问 \(p. 58\)](#)
 - [AWS CLI \(p. 58\)](#)
- [使用 AWS CLI \(p. 59\)](#)
- [使用 bash 脚本 \(p. 60\)](#)
- [使用 POST API 请求 \(p. 60\)](#)
- [使用 Python 脚本 \(p. 61\)](#)
- [接下来做什么？ \(p. 61\)](#)

先决条件

以下部分介绍使用本页面上的命令和脚本所需的初步步骤。

访问

- AWS中的账户访问AWS Identity and Access Management(IAM) 到中的亚马逊 MWAA 权限策略[Apache Airflow UI 访问政策：AmazonmwaWebServerAccess \(p. 38\)](#)。
- AWS中的账户访问AWS Identity and Access Management(IAM) 到亚马逊 MWAA 权限策略[完整的 API 和控制台访问政策：AmazonmWAAPullApiAccess \(p. 36\)](#)。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页面上的步骤，您需要以下满足以下条件：

- [AWS CLI- 安装版本 2](#)。
- [AWS CLI- 快速配置，使用aws configure](#)。

使用 AWS CLI

以下示例使用[创建网络登录令牌](#)中的命令AWS CLI创建 Apache Airflow Web 登录令牌。

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

使用 bash 脚本

以下示例使用 bash 脚本来调用[创建网络登录令牌](#)中的命令AWS CLI创建 Apache Airflow Web 登录令牌。

1. 复制以下代码示例的内容并将其作为保存在本地。get-web-token.sh.

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

2. 替换中的占位符##为了YOUR_HOST_NAME和YOUR_ENVIRONMENT_NAME. 例如，公共网络的主机名可能看起来像这样（没有https://）：

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

3. （可选）macOS 和 Linux 用户可能需要运行以下命令以确保脚本可执行。

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

4. 运行以下脚本以获取 Web 登录令牌。

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

5. 命令提示符中应该会显示以下内容：

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

使用 POST API 请求

以下示例使用 POST API 请求来创建 Apache Airflow Web 登录令牌。

1. 复制以下 URL 并粘贴到 REST API 客户端的 URL 字段中。

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

2. 替换中的占位符##为了YOUR_HOST_NAME. 例如，公共网络的主机名可能看起来像这样（没有https://）：

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

3. 复制以下 JSON 并粘贴到 REST API 客户端的正文字段中。

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

4. 替换中的占位符##为了YOUR_ENVIRONMENT_NAME.
5. 在授权字段中添加键值对。例如，如果您使用的是邮差，则选择AWS签名，然后输入您的：
 - AWS_ACCESS_KEY_ID在AccessKey
 - AWS_SECRET_ACCESS_KEY在SecretKey
6. 您应看到以下响应：


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

使用 Python 脚本

以下示例使用[boto3 创建 web_login_token](#)方法在 Python 脚本中创建 Apache Airflow Web 登录令牌。您可以在亚马逊 MWAA 之外运行此脚本。您唯一需要做的是安装 boto3 库。您可能需要创建虚拟环境来安装库。假设你有[已配置AWS验证凭据](#)为您的账户提供。

1. 复制以下代码示例的内容并将其作为保存在本地。create-web-login-token.py。

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

2. 替换中的占位符##为了YOUR_ENVIRONMENT_NAME.
3. 运行以下脚本以获取 Web 登录令牌。

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

接下来做什么？

- 了解用于在以下位置创建 Web 登录令牌的亚马逊 MWAA API 操作[创建 WebloginToken 令牌](#).

创建 Apache 气流 CLI 令牌

您可以使用此页面上的命令生成 CLI 令牌，然后直接在命令行管理程序中调用适用于 Apache Airflow 的 Amazon 托管工作流 (MWAA) API。例如，您可以获取令牌，然后使用 Amazon MWAA API 以编程方式部署 DAG。以下部分包括使用创建 Apache Airflow CLI 令牌的步骤AWS CLI、卷曲脚本、Python 脚本或 bash 脚本。响应中返回的令牌在 60 秒内有效。

Note

这些区域有：AWS CLI令牌旨在替代 Shell 操作，而不是异步 API 命令。因此，可用的并发性是有限的。为确保 Web 服务器对用户保持响应，建议不要打开新的AWS CLI请求直到上一个成功完成。

目录

- [先决条件 \(p. 62\)](#)
 - [访问 \(p. 58\)](#)
 - [AWS CLI \(p. 58\)](#)

- [使用 AWS CLI \(p. 62\)](#)
- [使用 curl 脚本 \(p. 62\)](#)
- [使用 bash 脚本 \(p. 63\)](#)
- [使用 Python 脚本 \(p. 64\)](#)
- [接下来做什么？ \(p. 66\)](#)

先决条件

以下部分介绍了使用本页上的命令和脚本所需的初步步骤。

访问

- AWS账户访问权限AWS Identity and Access Management(IAM) 到中的亚马逊 MWAA 权限策略[Apache Airflow UI 访问政策：AmazonmwaWebServerAccess \(p. 38\)](#).
- AWS账户访问权限AWS Identity and Access Management(IAM) 到亚马逊 MWAA 权限策略[完整的 API 和控制台访问政策：AmazonmWAAPullApiAccess \(p. 36\)](#).

AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下满足以下条件：

- [AWS CLI- 安装版本 2](#).
- [AWS CLI- 快速配置，使用aws configure](#).

使用 AWS CLI

以下示例使用[create-cli-token](#)命令在AWS CLI以创建 Apache Airflow CLI 令牌。

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

使用 curl 脚本

以下示例使用 curl 脚本调用[create-web-login-token](#)命令在AWS CLI通过 Airflow Web 服务器上的终端节点调用 Airflow CLI。

Apache Airflow v2

1. 从文本文件中复制 curl 语句并将其粘贴到命令 shell 中。

Note

将其复制到剪贴板后，您可能需要使用编辑 > 粘贴从你的 shell 菜单中。

```
CLI_JSON=$(aws mwa --region YOUR_REGION create-cli-token --  
name YOUR_ENVIRONMENT_NAME) \  
&& CLI_TOKEN=$(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_mwa/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. 将占位符替换为YOUR_REGION用AWS您的环境所在的区域，YOUR_DAG_NAME，和YOUR_ENVIRONMENT_NAME。例如，公共网络的主机名可能如下所示（没有https://）：

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

3. 命令提示符中会显示以下内容：

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

Apache Airflow v1

1. 从文本文件中复制 cURL 语句并将其粘贴到命令 shell 中。

Note

将其复制到剪贴板后，您可能需要使用编辑 > 粘贴从你的 shell 菜单中。

```
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. 将占位符替换为YOUR_REGION用AWS您的环境所在的区域，YOUR_DAG_NAME，和YOUR_HOST_NAME。例如，公共网络的主机名可能如下所示（没有https://）：

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

3. 命令提示符中会显示以下内容：

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

4. 将占位符替换为YOUR_ENVIRONMENT_NAME和YOUR_DAG_NAME。

使用 bash 脚本

以下示例使用 bash 脚本调用[create-cli-token](#)命令在AWS CLI以创建 Apache Airflow CLI 令牌。

Apache Airflow v2

1. 复制以下代码示例的内容并将其作为get-cli-token.sh.

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

2. 替换中的占位符##为了YOUR_ENVIRONMENT_NAME、YOUR_HOST_NAME, 和YOUR_DAG_NAME. 例如, 公共网络的主机名可能如下所示 (没有https://):

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

3. (可选) macOS 和 Linux 用户可能需要运行以下命令以确保脚本可执行。

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

4. 运行以下脚本可创建 Airflow CLI 令牌。

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

Apache Airflow v1

1. 复制以下代码示例的内容并将其作为get-cli-token.sh.

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

2. 替换中的占位符##为了YOUR_ENVIRONMENT_NAME、YOUR_HOST_NAME, 和YOUR_DAG_NAME. 例如, 公共网络的主机名可能如下所示 (没有https://):

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

3. (可选) macOS 和 Linux 用户可能需要运行以下命令以确保脚本可执行。

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

4. 运行以下脚本可创建 Airflow CLI 令牌。

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

使用 Python 脚本

以下示例使用[boto3 create_cli_token](#)方法来创建 Apache Airflow CLI 令牌并触发 DAG。您可以在 Amazon MWAA 之外运行此脚本。您需要做的唯一一件事是安装 boto3 库。您可能需要创建一个虚拟环境来安装该库。它假设你有[已配置AWS身份验证凭证](#)用于您的账户。

Apache Airflow v2

1. 复制以下代码示例的内容并将其作为create-cli-token.py.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import boto3
import json
import requests
import base64

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

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

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

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

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

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

2. 将占位符替换为YOUR_ENVIRONMENT_NAME和YOUR_DAG_NAME.
3. 运行以下脚本可创建 Airflow CLI 令牌。

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

Apache Airflow v1

1. 复制以下代码示例的内容并将其作为 `create-cli-token.py`。

```
import boto3
import json
import requests
import base64

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

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

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

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

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

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

2. 将占位符替换为 `YOUR_ENVIRONMENT_NAME` 和 `YOUR_DAG_NAME`。
3. 运行以下脚本可创建 Airflow CLI 令牌。

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

接下来做什么？

- 在以下位置浏览用于创建 CLI 令牌的亚马逊 MWAA API 操作 [CreateCliToken](#)。

Apache Airflow Air

本页描述了亚马逊 Managed Workflows for Apache Airflow (MWAA) 上支持和不支持的。

目录

- [先决条件 \(p. 67\)](#)

- [访问 \(p. 58\)](#)
- [AWS CLI \(p. 58\)](#)
- [v2 中有什么变化 \(p. 67\)](#)
- [支持的 CLI 命令 \(p. 67\)](#)
 - [支持的命令 \(p. 68\)](#)
 - [使用解析 DAG 的命令 \(p. 70\)](#)
- [示例代码 \(p. 70\)](#)
 - [设置、获取或删除 Apache Airflow v2 变量 \(p. 70\)](#)
 - [触发 DAG 时添加配置 \(p. 71\)](#)
 - [在通往堡垒主机的 SSH 隧道上运行 CLI 命令 \(p. 71\)](#)
 - [中的示例 GitHub 和AWS教程 \(p. 72\)](#)

先决条件

下一节描述了使用此页上的命令和脚本所需的初步步骤。

访问

- AWS Identity and Access Management(IAM) 中的账户访问权限中的亚马逊 MWAA 权限策略 [Apache Airflow UI 访问政策 : AmazonmwaWebServerAccess \(p. 38\)](#)。
- AWS Identity and Access Management(IAM) 中对亚马逊 MWAA 权限策略的账户访问权限 [完整的 API 和控制台访问政策 : AmazonmWAAPFullApiAccess \(p. 36\)](#)。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要满足以下条件：

- [AWS CLI— 安装版本 2](#)。
- [AWS CLI— 使用aws configure. 进行快速配置](#)。

v2 中有什么变化

- 新增：Airflow CLI 命令结构。Apache Airflow v2 CLI 的组织方式是将相关命令分组为子命令，这意味着如果你想升级到 Apache Airflow v2，你需要更新 Apache Airflow v1 脚本。例如，unpause在 Apache 中，Airflow v1 现在在 dags unpause在 Apache Airflow v2 中。要了解更多信息，请参阅 Apache Airflow 参考指南中 2 中的 [Air flow CLI 更改](#)。

支持的 CLI 命令

以下部分列出了亚马逊 MWAA 上可用的 Apache Airflow CLI 命令。

支持的命令

Apache Airflow v2

次要版本	命令	
v2.0+	备忘单	
v2.0+	连接添加	
v2.0+	连接删除	
v2.2+ (注意 (p. 70))	dags 回填	
v2.0+	dags 删除	
v2.2+ (注意 (p. 70))	天数清单	
v2.0+	dags list-jobs	
v2.2+ (注意 (p. 70))	dags list-runs	
v2.2+ (注意 (p. 70))	dags 下一次处决	
v2.0+	停顿几天	
v2.0+	每日报告	
v2.4+	dags 重新序列化	
v2.0+	每日秀	
v2.0+	dags state	
v2.0+	dags 测试	
v2.0+	dags 触发器	
v2.0+	dags 取消暂停	
v2.4+	db clean	
v2.0+	提供者的行为	
v2.0+	提供商获得	
v2.0+	提供者挂钩	
v2.0+	提供者链接	
v2.0+	提供者名单	
v2.0+	提供商小部件	
v2.0+	角色创建	
v2.0+	角色列表	
v2.0+	任务已清除	
v2.0+	任务 failed-deps	

次要版本	命令	
v2.0+	任务清单	
v2.0+	任务渲染	
v2.0+	任务状态	
v2.0+	任务 states-for-dag-run	
v2.0+	任务测试	
v2.0+	变量删除	
v2.0+	变量获取	
v2.0+	变量集	
v2.0+	变量列表	
v2.0+	版本	

Apache Airflow v1

次要版本	命令	
v1.10.10.12 (注意 (p. 70))	回填	
v1.10.10.12 接受的	清除	
v1.10.10.12 (注意 (p. 70))	dag_state	
v1.10.10.12 接受的	delete_dag	
v1.10.10.12 (注意 (p. 70))	list_dag_runs	
v1.10.10.12 注意 (p. 70)	list_dags	
v1.10.10.12 (注意 (p. 70))	列出任务	
v1.10.10.12 接受的	下一步执行	
v1.10.10.12 接受的	暂停	
v1.10.10.12 接受的	池	
v1.10.10.12 接受的	给予	
v1.10.10.12 接受的	跑	
v1.10.10.12 (注意 (p. 70))	show_dag	
v1.10.10.12 接受的	task_failed_deps	
v1.10.10.12 (注意 (p. 70))	任务状态	
v1.10.10.12 (注意 (p. 70))	测试	
v1.10.10.12 接受的	trigger_dag	
v1.10.10.12 接受的	取消暂停	

次要版本	命令	
v1.10.10.12 接受的	变量	
v1.10.10.12 接受的	版本	

使用解析 DAG 的命令

如果您的环境运行的是 Apache Airflow v1.10.12 或 v2.0.2，则如果 DAG 使用依赖于通过以下方式安装的软件包的插件，则解析 DAG 的 CLI 命令将失败 requirements.txt：

Apache Airflow Ap

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

Apache Airflow Apach

- backfill
- list_dag_runs
- list_dags
- list_tasks
- show_dag
- dag_state
- task_state
- test

如果您的 DAG 不使用依赖于通过安装的软件包的插件，则可以使用这些 CLI 命令 requirements.txt。

示例代码

以下部分包含使用 Apache Airflow CLI 的不同方法的示例。

设置、获取或删除 Apache Airflow v2 变量

您可以使用以下示例代码设置、获取或删除格式为的变量<script> <mwa env name> get | set | delete <variable> <variable value> </variable> </variable>。

```
[ $# -eq 0 ] && echo "Usage: $0 MWAA environment name " && exit
if [[ $2 == "" ]]; then
    dag="variables list"

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

else
    echo "Not a valid command"
```

```

    exit 1
fi

CLI_JSON=$(aws mwaas --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/awsmwaas/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

```

触发 DAG 时添加配置

你可以在 Apache Airflow v1 和 Apache Airflow v2 中使用以下示例代码在触发 DAG 时添加配置，例如 `airflow trigger_dag 'dag_name' -conf '{"key":"value"}'`。

```

import boto3
import json
import requests
import base64

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

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/awsmwaas/'
cli = 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)

```

在通往堡垒主机的 SSH 隧道上运行 CLI 命令

以下示例显示了如何使用 SSH 隧道代理在 Linux 堡垒主机上运行 Airflow CLI 命令。

使用 curl

1.

```
ssh -D 8080 -f -C -q -N YOUR_USER@YOUR_BASTION_HOST
```
2.

```
curl -x socks5h://0:8080 --request POST https://YOUR_HOST_NAME/aws_mwaa/cli --  
header YOUR_HEADERS --data-raw YOUR_CLI_COMMAND
```

中的示例 GitHub 和AWS教程

- [在 Amazon Managed Workflows for Apache Airflow \(MWAA\) 中使用阿帕奇Airflow \(](#)
- [通过命令行与亚马逊 MWAA 上的 Apache Airflow v1.10.12 进行互动](#)
- [亚马逊 MWAA 上的 Apache Airflow v1.10.12 和 Bash Operator 开启的交互式命令](#) GitHub

管理与 Apache Airflow 的连接

本节介绍了为 Amazon Managed Workflow in Apache Airflow (MWAA) 环境配置 Apache Airflow (MWAA) 环境。

主题

- [Apache 气流变量和连接概述 \(p. 73\)](#)
- [安装在亚马逊 MWAA 环境中的 Apache Airflow 提供商软件包 \(p. 73\)](#)
- [连接类型概览 \(p. 76\)](#)
- [使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)

Apache 气流变量和连接概述

在某些情况下，您可能希望指定其他连接或变量，例如，AWS配置文件，或者要在 Apache Airflow 元数据仓库中的连接对象中添加执行角色，然后参考 DAG 中的连接。

- 自行管理的 Apache Airflow. 在自我管理的 Apache Airflow 安装中，你设置中的 [Apache 气流配置选项airflow.cfg](#).

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

- 亚马逊上的 Apache 气流 MWAA. 在亚马逊 MWAA 上，您需要将这些配置设置添加为[Apache 气流配置选项 \(p. 90\)](#)在亚马逊 MWAA 控制台上。Apache Airflow 配置选项作为环境变量写入您的环境，并覆盖相同设置的所有其他现有配置。

安装在亚马逊 MWAA 环境中的 Apache Airflow 提供商软件包

当您创建新环境时，Amazon MWAA 会为 Apache Airflow v2 及更高版本的连接类型安装[提供商附加功能](#)。安装提供程序包允许您在 Apache Airflow UI 中查看连接类型。这也意味着您无需在requirements.txt文件中将这些包指定为 Python 依赖项。本页列出了亚马逊 MWAA 为所有 Apache Airflow v2 环境安装的 Apache Airflow 提供商软件包。

Note

对于 Apache Airflow v2 及更高版本，[亚马逊 MWAA 在执行pip3 install -r requirements.txt后安装 Watchtower 版本 2.0.1](#)，以确保与 CloudWatch 日志的兼容性不会被其他 Python 库安装所覆盖。

目录

- [Apache Airflow v2.5.1 连接的提供者软件包 \(p. 74\)](#)

- [Apache Airflow v2.4.3 连接的提供者软件包 \(p. 74\)](#)
- [Apache Airflow v2.2.2 连接的提供者软件包 \(p. 75\)](#)
- [Apache Airflow v2.0.2 \(p. 75\)](#)
- [指定更新的提供程序包 \(p. 75\)](#)

Apache Airflow v2.5.1 连接的提供者软件包

当您在 Apache Airflow v2.5.1 中创建亚马逊 MWAA 环境时，亚马逊 MWAA 会安装以下用于 Apache Airflow 连接的提供程序包。

Note

您可以指定支持的最新版本 `apache-airflow-providers-amazon` 来升级此提供商。有关指定更新版本的更多信息，请参阅 [the section called “指定更新的提供程序包” \(p. 75\)](#)。

连接类型	程序包
AWS 连接	apache-airflow-providers-amazon==7.1.0
Postgres 连接	apache-airflow-providers-postgres==5.4.0
FTP 连接	apache-airflow-providers-ftp==3.3.0
芹菜连接	apache-airflow-providers-celery==3.1.0
HTTP 连接	apache-airflow-providers-http==4.1.1
IMAP 连接	apache-airflow-providers-imap==3.1.1
常见的 SQL	apache-airflow-providers-common-sql==1.3.3
SQLite 连接	apache-airflow-providers-sqlite==3.3.1

Apache Airflow v2.4.3 连接的提供者软件包

当您在 Apache Airflow v2.4.3 中创建亚马逊 MWAA 环境时，亚马逊 MWAA 会安装以下用于 Apache Airflow 连接的提供程序包。

连接类型	程序包
AWS 连接	apache-airflow-providers-amazon==6.0.0
Postgres 连接	apache-airflow-providers-postgres=5.2.2
FTP 连接	apache-airflow-providers-ftp==3.1.0
芹菜连接	apache-airflow-providers-celery==3.0.0
HTTP 连接	apache-airflow-providers-http==4.0.0
IMAP 连接	apache-airflow-providers-imap==3.0.0
常见的 SQL	apache-airflow-providers-common-sql==1.2.0
SQLite 连接	apache-airflow-providers-sqlite=3.2.1

Apache Airflow v2.2.2 连接的提供者软件包

当您在 Apache Airflow v2.2.2 中创建亚马逊 MWAA 环境时，亚马逊 MWAA 会安装以下用于 Apache Airflow 连接的提供程序包。

连接类型	程序包
AWS 连接	apache-airflow-providers-amazon==2.4.0
Postgres 连接	apache-airflow-providers-postgres==2.3.0
FTP 连接	apache-airflow-providers-ftp==2.0.1
芹菜连接	apache-airflow-providers-celery==2.1.0
HTTP 连接	apache-airflow-providers-http==2.0.1
IMAP 连接	apache-airflow-providers-imap==2.0.1
SQLite 连接	apache-airflow-providers-sqlite==2.0.1

Apache Airflow v2.0.2

当您在 Apache Airflow v2.0.2 中创建亚马逊 MWAA 环境时，亚马逊 MWAA 会安装以下用于 Apache Airflow 连接的提供程序包。

连接类型	程序包
Tableau	apache-airflow-providers-tableau==1.0.0
Databricks 连接	apache-airflow-providers-databricks==1.0.1
SSH 连接	apache-airflow-providers-ssh==1.3.0
Postgres 连接	apache-airflow-providers-postgres==1.0.2
Docker 连接	apache-airflow-providers-docker==1.2.0
Oracle 连接	apache-airflow-providers-oracle==1.1.0
Presto 连接	apache-airflow-providers-presto==1.0.2
SFTP 连接	apache-airflow-providers-sftp==1.2.0

指定更新的提供程序包

从 Apache Airflow v2.5.1 开始，您的需求文件必须包含 `--constraint` 声明。如果您未提供限制条件，Amazon MWAA 将为您指定一个限制条件，以确保您的要求中列出的软件包与您正在使用的 Apache Airflow 版本兼容。

Apache Airflow 限制文件指定了 Apache Airflow 发布时可用的提供者版本。但是，在许多情况下，较新的提供商与该版本的 Apache Airflow 兼容。由于必须使用约束，因此要指定提供程序包的更新版本，您可以修改特定提供者版本的限制文件。

例如，要将亚马逊提供商软件包版本 7.3.0 与 Apache Airflow v2.5.1 一起使用，请执行以下操作：

1. 从 <https://raw.githubusercontent.com/apache/airflow/constraints-2.5.1/constraints-3.10.txt> 下载特定版本的限制文件”
2. 将约束文件中的 apache-airflow-providers-amazon 版本修改为 7.3.0。
3. 例如，将修改后的限制文件保存到亚马逊 MWAA 环境的 Amazon S3 dags 文件夹，例如 constraints-3.10-mod.txt
4. 指定您的要求，如下所示。

```
--constraint "/usr/local/airflow/dags/constraints-3.10-mod.txt"
apache-airflow-providers-amazon==7.3.0
```

Note

如果您使用的是私有 Web 服务器，我们建议您使用 Amazon MWAA Local-runner 将所需的库打包为 WHL 文件 (p. 250)。

连接类型概览

Apache Airflow 将连接存储为连接 URI 字符串。无论连接类型如何，它都会在 Apache Airflow UI 中提供连接模板以生成连接 URI 字符串。如果 Apache Airflow UI 中没有连接模板，则可以使用备用连接模板生成此连接 URI 字符串，例如使用 HTTP 连接模板。主要区别在于 URI 前缀，例如 my-conn-type://，Apache Airflow 提供商在连接时通常会忽略该前缀。本页介绍如何将 Apache Airflow UI 中的连接模板互换用于不同的连接类型。

Warning

不要重写亚马逊 MWAA 中的 `aws_default` 连接。Amazon MWAA 使用此连接来执行各种关键任务，例如收集任务日志。重写此连接可能会导致数据丢失和环境可用性中断。

主题

- [连接 URI 字符串示例 \(p. 76\)](#)
- [示例连接模板 \(p. 76\)](#)
- [使用 HTTP 连接模板进行 Jdbc 连接的示例 \(p. 77\)](#)

连接 URI 字符串示例

以下示例显示了 MySQL 连接类型的连接 URI 字符串。

```
'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'
```

示例连接模板

以下示例显示 Apache Airflow 用户界面中的 HTTP 连接模板。

Apache Airflow v2

以下示例显示了 Apache Airflow 用户界面中 Apache Airflow v2 的 HTTP 连接模板。

Apache Airflow v1

以下示例显示了 Apache Airflow 用户界面中 Apache Airflow v1 的 HTTP 连接模板。

使用 HTTP 连接模板进行 Jdbc 连接的示例

以下示例显示如何在 Apache Airflow v2.0.2 中将 HTTP 连接模板用于 Jdbc 连接类型，以及如何在 Apache Airflow UI 中 Apache Airflow v1.10.12 的 Jdbc 连接模板中使用相同的值。

Apache Airflow v2

以下示例显示了本节示例中由 Apache Airflow 生成的连接 URI 字符串。

```
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
```

以下示例显示了如何在 Apache Airflow UI 中使用 HTTP 连接模板进行 Apache Airflow v2 的 Jdbc 连接。

Add Connection	
Conn Id *	my_jdbc_conn
Conn Type *	HTTP Conn Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Host	myconnectionurl/some/path
Schema	
Login	mylogin
Password	
Port	
Extra	<pre>{ "extra__jdbc__dry_path": "/usr/local/airflow/dags/classpath/redshift-jdbc42-2.0.0.1.jar", "extra__jdbc__dry_clsname": "redshift-jdbc42-2.0.0.1" }</pre>
Save ↩	

Apache Airflow v1

以下示例显示了本节示例中由 Apache Airflow 生成的连接 URI 字符串。

```
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
```

以下示例显示了 Apache Airflow 用户界面中 Apache Airflow v1.10.12 的 Jdbc 连接模板。

Add Connection	
Conn Id *	my_jdbc_conn
Conn Type	Jdbc Connection
Connection URL	myconnectionrurl/some/path
Login	mylogin
Password	
Driver Path	/usr/local/airflow/dags/classpath/redshift-jdbc42-2.0.0.1.jar
Driver Class	redshift-jdbc42-2.0.0.1
Save ↩	

使用AWS Secrets Manager密钥配置 Apache Airflow 连接

AWS Secrets Manager是 Amazon Managed Workflows for Apache Airflow (MWAA) 环境中支持的备用式 Apache 本指南介绍如何在 Amazon Managed Workflows for Apache Airflow (MWAA) 上使用AWS Secrets Manager Apache Airflow 连接安全地存储 Apache Airflow 连接

Note

- 您将需要为您创建的Secret付费。有关 Secrets Manager 定价的更多信息，请参阅[AWS定价](#)。

目录

- [第一步：向亚马逊 MWAA 提供访问密钥Secrets Manager 钥的权限 \(p. 79\)](#)
- [第二步：将Secrets Manager 后端创建为 Apache Airflow 配置选项 \(p. 80\)](#)
- [第三步：生成 Apache AirflowAWS 连接 URI 字符串 \(p. 81\)](#)
- [第四步：在Secrets Manager 中添加变量 \(p. 82\)](#)
- [第五步：在Secrets Manager 中添加连接 \(p. 83\)](#)
- [示例代码 \(p. 84\)](#)
- [资源 \(p. 84\)](#)
- [接下来做什么？ \(p. 84\)](#)

第一步：向亚马逊 MWAA 提供访问密钥Secrets Manager 钥的权限

您的 Amazon MWAA 环境的[执行角色 \(p. 44\)](#)需要对中的密钥具有读取权限AWS Secrets Manager。以下 IAM 策略允许使用AWS托管[SecretsManagerReadWrite](#)策略进行读写访问。

将策略附加到执行角色

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“权限”窗格中选择您的执行角色。
4. 选择 Attach policies (附上策略) 。
5. SecretsManagerReadWrite在筛选策略文本字段中键入。
6. 选择 Attach policy (附上策略) 。

如果您不想使用AWS托管权限策略，则可以直接更新环境的执行角色，以允许对您的 Secrets Manager 资源进行任何级别的访问。例如，以下策略声明授予您在 Secrets Manager 中在特定AWS区域创建的所有密钥的读取权限。

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

第二步：将Secrets Manager 后端创建为 Apache Airflow 配置选项

以下部分介绍如何在亚马逊 MWAA 控制台上为AWS Secrets Manager后端创建 Apache Airflow 配置选项。如果您在中使用同名的配置设置airflow.cfg，则您在以下步骤中创建的配置将优先于并覆盖配置设置。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 选择下一步。
5. 在 Airflow 配置选项窗格中选择添加自定义配置。添加以下键/值对：
 - a. **secrets.backend:**
airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
 - b. **secrets.backend_kwargs:** {"connections_prefix" : "airflow/connections",
"variables_prefix" : "airflow/variables"}这将 Apache Airflow 配置为在
和airflow/variables/*路径上查找连接字符串airflow/connections/*和变量。

您可以使用[查询模式](#)减少亚马逊 MWAA 代表您向 Secrets Manager 发出 API 调用的次数。如果您未指定查找模式，Apache Airflow 会在配置的后端中搜索所有连接和变量。通过指定模式，可以缩小 Apache Airflow 显示的可能路径。这将降低您在亚马逊 MWAA 上使用Secrets Manager 时的成本。

要指定查找模式，请指定connections_lookup_pattern和variables_lookup_pattern参数。这些参数接受 RegEx 字符串作为输入。例如，要查找以开头的密钥test，请输入以下内容secrets.backend_kwargs：

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

Note

要使用connections_lookup_pattern和variables_lookup_pattern，必须安装7.3.0 或更高apache-airflow-providers-amazon版本。有关将提供程序包更新到新版本的更多信息，请参阅[the section called “指定更新的提供程序包” \(p. 75\)](#)。

6. 选择保存。

第三步：生成 Apache AirflowAWS 连接 URI 字符串

要创建连接字符串，请使用键盘上的“tab”键缩进 Connection 对象中的键值对。我们还建议在 shell 会话中为该extra对象创建一个变量。以下部分将引导您完成使用 [Apache Airflow 或 Python 脚本为亚马逊 MWAA 环境生成 Apache Airflow 连接 URI](#) 字符串的步骤。

Apache Airflow CLI

以下 shell 会话使用您的本地 Airflow CLI 生成连接字符串。如果您没有安装 CLI，我们建议使用 Python 脚本。

1. 打开 Python 外壳会话：

```
python3
```

2. 输入以下命令：

```
>>> import json
```

3. 输入以下命令：

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

4. 在 shell 会话中为该extra对象创建一个变量。将 **YOUR_EXECUTION_ROLE_ARN** 中的示例值替换为执行角色 ARN 和 **YOUR_REGION** **####** (例如us-east-1)。

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

5. 创建连接对象。将中的myconn示例值替换为 Apache Airflow 连接的名称。

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

6. 使用键盘上的“tab”键在连接对象中缩进以下每个键值对。用**##**替换样本值。

- a. 指定AWS连接类型：

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

- b. 指定 Apache Airflow 数据库选项：

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

- c. 在亚马逊 MWAA 上指定 Apache Airflow 用户界面网址：

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

- d. 指定AWS访问密钥 ID (用户名) 以登录亚马逊 MWAA：

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

- e. 指定用于登录亚马逊 MWAA 的私有访问AWS密钥 (密码)：

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

- f. 指定extra shell 会话变量：

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

- g. 关闭连接对象。

```
... )
```

7. 打印连接 URI 字符串：

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

你应该在响应中看到连接 URI 字符串：

```
'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

以下 Python 脚本不需要 Apache Airflow CLI。

1. 复制以下代码示例的内容并在本地另存为mwaa_connection.py。

```
import urllib.parse

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

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

2. 用##替换占位符。
3. 运行以下脚本可生成连接字符串。

```
python3 mwaa_connection.py
```

第四步：在Secrets Manager 中添加变量

下一节介绍如何在密钥管理器中为变量创建密钥。

创建密钥

1. 打开 [AWS Secrets Manager 控制台](#)。
2. 选择 Store a new secret (存储新密钥)。
3. 选择其他类型的密钥。
4. 在“指定要存储在此密钥中的键/值对”窗格上，选择 Plaintext。

5. 将变量值添加为以下格式的纯文本。

```
"YOUR_VARIABLE_VALUE"
```

例如，要指定一个整数：

```
14
```

例如，要指定字符串：

```
"mystring"
```

6. 对于加密密钥，从下拉列表中选择一个AWS KMS密钥选项。
7. 在密钥名称的文本字段中输入以下格式的名称。

```
airflow/variables/YOUR_VARIABLE_NAME
```

例如：

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

8. 选择下一步。
9. 在配置密钥页面的密钥名称和描述窗格上，执行以下操作。
 - a. 对于密钥名称，请提供您的密钥的名称。
 - b. （可选）对Secrets（可选），对密钥的描述，对密钥的描述

选择下一步。

10. 在“配置轮换-可选”上，保留默认选项，然后选择“下一步”。
11. 对于要添加的任何其他变量，请在 Secrets Manager 中重复这些步骤。
12. 在“查看”页面上，查看您的密钥，然后选择“存储”。

第五步：在Secrets Manager 中添加连接

以下部分介绍如何在密钥管理器中为连接字符串 URI 创建密钥。

创建密钥

1. 打开 [AWS Secrets Manager 控制台](#)。
2. 选择 Store a new secret (存储新密钥)。
3. 选择其他类型的密钥。
4. 在“指定要存储在此密钥中的键/值对”窗格上，选择 Plaintext。
5. 使用以下格式将连接 URI 字符串添加为纯文本。

```
YOUR_CONNECTION_URI_STRING
```

例如：

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

Warning

Apache Airflow 解析连接字符串中的每个值。您不得使用单引号或双引号，否则它会将连接解析为单个字符串。

- 对于加密密钥，从下拉列表中选择一个AWS KMS密钥选项。
- 在密钥名称的文本字段中输入以下格式的名称。

```
airflow/connections/YOUR_CONNECTION_NAME
```

例如：

```
airflow/connections/myconn
```

- 选择下一步。
- 在配置密钥页面的密钥名称和描述窗格上，执行以下操作。
 - 对于密钥名称，请提供您的密钥的名称。
 - (可选) 对Secrets (可选)，对密钥的描述，对密钥的描述

选择下一步。

- 在“配置轮换-可选”上，保留默认选项，然后选择“下一步”。
- 对于要添加的任何其他变量，请在 Secrets Manager 中重复这些步骤。
- 在“查看”页面上，查看您的密钥，然后选择“存储”。

示例代码

- 使用以下示例代码，了解如何使用本页上的 Apache Airflow 连接的密钥 (myconn) [在中使用密钥AWS Secrets Manager用于 Apache 气流连接 \(p. 189\)](#)。
- 使用以下示例代码，了解如何使用本页上的 Apache Airflow 变量 (test-variable) 的密钥 [在中使用密钥AWS Secrets Manager对于 Apache 气流变量 \(p. 187\)](#)。

资源

- 有关使用控制台和配置密钥管理器密钥的更多信息AWS CLI，请参阅AWS Secrets Manager用户指南中的[创建密钥](#)。
- 在“将 Apache Airflow 连接和变量移至”中，使用 Python 脚本将大量 A [pache Airflow 变量和连接迁移到 Secrets ManagerAWS Secrets Manager](#)。

接下来做什么？

- 在中学习如何生成令牌以访问 Apache Airflow UI [访问 Apache Airflow界面 \(p. 58\)](#)。

管理亚马逊 MWAA 环境

Amazon Managed Workflows for Apache Airflow (MWAA)，包含内置选项，用于配置对 Apache Airflow (MWAA) 它还包含用于配置环境大小、何时扩展工作人员的内置选项，以及允许您替代通常只能在中访问的 Apache Airflow 配置的 Apache Airflow 配置选项 `airflow.cfg`。本指南介绍如何在亚马逊 MWAA 控制台上使用这些配置。

主题

- [配置个 Amazon MWAA 环境类 \(p. 85\)](#)
- [配置亚马逊 MWAA 自动扩展 \(p. 87\)](#)
- [在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)
- [在亚马逊 MWAA 上使用启动脚本 \(p. 97\)](#)

配置个 Amazon MWAA 环境类

您为 Amazon MWAA 环境选择的环境类决定了 [Celery Executor](#) 运行的AWS托管AWS Fargate容器的大小，以及 Apache Airflow 调度程序在其中创建任务实例的 Amazon Aurora PostgreSQL 元数据数据库的大小。AWS本页描述了每个 Amazon MWAA 环境类以及在 Amazon MWAA 控制台上更新环境类的步骤。

小节目录

- [环境类别 \(p. 85\)](#)
- [环境功能 \(p. 86\)](#)
- [气流调度器 \(p. 87\)](#)

环境类别

下图显示了您可以在 Amazon MWAA 控制台上更新环境类的位置。

Apache Airflow v2

Environment class [Info](#)

Each Amazon MWAA environment includes the scheduler, web server, and 1 worker. Workers auto-scale up and down according to system load. You can monitor the load on your environment and modify its class at any time.

	DAG capacity*	Scheduler CPU	Worker CPU	Web server CPU
☒ mw1.small	Up to 50	1 vCPU	1 vCPU	0.5 vCPU
☐ mw1.medium	Up to 250	2 vCPU	2 vCPU	1 vCPU
☐ mw1.large	Up to 1000	4 vCPU	4 vCPU	2 vCPU

*under typical usage

Maximum worker count
The maximum number of workers your environment is permitted to scale up to.

Must be between 1 and 25

Scheduler count
The number of schedulers your environment will use.

Must be between 2 and 5

Apache Airflow v1

Environment class [Info](#)

Each Amazon MWAA environment includes the scheduler, web server, and 1 worker. Workers auto-scale up and down according to system load. You can monitor the load on your environment and modify its class at any time.

	DAG capacity*	Scheduler CPU	Worker CPU	Web server CPU
☒ mw1.small	Up to 50	1 vCPU	1 vCPU	0.5 vCPU
☐ mw1.medium	Up to 250	2 vCPU	2 vCPU	1 vCPU
☐ mw1.large	Up to 1000	4 vCPU	4 vCPU	2 vCPU

*under typical usage

Maximum worker count
The maximum number of workers your environment is permitted to scale up to.

Must be between 1 and 25

环境功能

以下部分包含每个环境类别的默认并发 Apache Airflow 任务、随机存取存储器 (RAM) 和虚拟集中处理单元 (vCPUs)。列出的并发任务假设任务并发不会超过环境中的 Apache Airflow Worker 容量。

mw1.small

- 5 个并发任务 (默认情况下)
- vCPUs
- 2 GB 内存

mw1.medium

- 10 个并发任务 (默认情况下)
- vCPUs
- 4 GB 内存

mw1.large

- 20 个并发任务 (默认情况下)
- vCPUs
- 8 GB RAM

您可以使用 `celery.worker_autoscale` 来增加每个工作人员的任务。要了解更多信息，请参阅 [the section called “高性能用例示例” \(p. 89\)](#)。

气流调度器

以下部分包含亚马逊 MWAA 控制台上可用的 Apache Airflow Scheduler 选项。

Apache Airflow v2

- v2-接受介于 2 到 5 之间。默认值为 2。

Apache Airflow v1

- v1-接受 1。

配置亚马逊 MWAA 自动扩展

自动缩放机制会自动增加 Apache Airflow 工作流的数量，以响应 Amazon Apache Airflow for AWS (MWAA) 环境中正在运行和排队的任务，并在没有更多排队或正在执行的任务时处置额外的工作人员。本页介绍如何通过使用 Amazon MWAA 控制台指定在您的环境中运行的 Apache Airflow 工作线程的最大数量来配置自动扩展。

Note

Amazon MWAA 使用 Apache Airflow 指标来确定何时需要额外的 [Celery Executor](#) 工作人员，并根据需要将 Fargate 工作人员的数量增加到指定的值 `max-workers`。当该数字为零时，Amazon MWAA 会移除额外的工作人员，缩减至该 `min-workers` 值。有关更多信息，请参阅以下 [the section called “工作原理” \(p. 88\)](#) 部分。

缩小规模时，可以安排新任务。此外，设置为删除的工作人员有可能在移除工作容器之前接管这些任务。这段时间可能持续两到五分钟，这要归因于多种因素：发送 Apache Airflow 指标所需的时间、检测到零任务的稳定状态的时间以及移除 Fargate 工作人员所需的时间。

如果您在持续工作负载时使用 Amazon MWAA，然后是一段没有工作负载的时期，则您将不受此限制的影响。但是，如果您的工作负载间歇性很强，反复出现高使用率，然后在大约五分钟内没有任务，则当在缩减规模的 worker 上运行的任务被删除并标记为失败时，您可能会受到此问题的影响。

如果您受到此限制的影响，我们建议您执行以下任一操作：

- 设置min-workers等于的容量足以满足您的平均工作负载，如果这种模式max-workers在 24 小时的大部分时间内持续存在，则更可取，因为在这种情况下，自动缩放的价值有限。
- 在这段间歇性活动期间，确保一个 DAG 中至少有一个任务（例如 a [DateTimeSensor](#)）在运行，以防止不必要的缩小规模。

小节目录

- [工作线程计数 \(p. 88\)](#)
- [工作原理 \(p. 88\)](#)
- [使用 Amazon MWAA 控制台 \(p. 89\)](#)
- [高性能用例示例 \(p. 89\)](#)
- [对卡在运行状态的任务进行故障排除 \(p. 90\)](#)
- [接下来做什么？ \(p. 90\)](#)

工作线程计数

下图显示了您可以在 Amazon MWAA 控制台上自定义最大工作线程数以配置自动扩展。

Environment class Info

Each Amazon MWAA environment includes the scheduler, web server, and 1 worker. Workers auto-scale up and down according to system load. You can monitor the load on your environment and modify its class at any time.

	DAG capacity*	Scheduler CPU	Worker CPU	Web server CPU
☒ mw1.small	Up to 50	1 vCPU	1 vCPU	0.5 vCPU
☐ mw1.medium	Up to 250	2 vCPU	2 vCPU	1 vCPU
☐ mw1.large	Up to 1000	4 vCPU	4 vCPU	2 vCPU

*under typical usage

Maximum worker count
 The maximum number of workers your environment is permitted to scale up to.

15

 Must be between 1 and 25

工作原理

Amazon MWAA 使用RunningTasks和QueuedTasks[指标 \(p. 266\)](#)，其中（正在运行的任务 + 排队的任务）/（每个工作人员 [\(p. 86\)](#) 的任务数）=（所需工作人员）。如果所需的工作人员数量大于当前的工作人员数量，Amazon MWAA 将向该值添加 Fargate 工作容器，直至达到指定的最大值max-workers。

当RunningTasks和QueuedTasks指标在两分钟内总和为零时，Amazon MWAA 请求 Fargate 将工作人员数量设置为环境的min-workers值。Amazon MWAA[stopTimeout](#) 为 Fargate 提供的值为 120 秒（目前的最大可用时间），以允许工作人员完成任何工作，之后将移除容器并删除所有剩余正在进行的工作。在大多数情况下，这是在队列中没有任务时发生的，但是在本页[前面部分 \(p. 87\)](#)提到的某些条件下，任务可能会在缩减规模时排队。

当您创建环境时，Amazon MWAA 会在不同可用区的两个私有子网中分别创建 AWS 管理的 Amazon Aurora PostgreSQL 元数据数据库和一个 Fargate 容器。例如，中的元数据数据库和容器us-east-1a以及us-east-1该区域us-east-1b可用区中的元数据数据库和容器。

- 亚马逊 MWAA 环境中的 Apache Airflow 工作人员使用 [Celery Executor](#) 排队并将任务分发给来自 Apache Airflow 平台的多个 Celery 员工。Celery Executor 在 AWS Fargate 容器中运行。如果一个可用区中的 Fargate 容器出现故障，Amazon MWAA 会切换到另一个可用区中的另一个容器来运行 Celery Executor，Apache Airflow 调度程序会在 Amazon Aurora PostgreSQL 元数据数据库中创建一个新的任务实例。
- 默认情况下，Amazon MWAA 将环境配置为并行（输入 `core.parallelism`）数百个任务和 `parallel`（输入 `core.dag_concurrency`）运行数百个任务。当任务排队时，Amazon MWAA 会添加工作人员以满足需求，直到达到您在最大工作人员人数中定义的人数。
- 例如，如果您指定的值为 10，则 Amazon MWAA 最多可增加 9 名员工以满足需求。这种自动缩放机制将继续运行额外的工作程序，直到没有更多任务要运行。如果没有更多的任务要运行，并且队列中也没有更多任务了，Amazon MWAA 就会处置多余的工作线程，仅留下程数量。

使用 Amazon MWAA 控制台

您可以在 Amazon MWAA 控制台上选择可在您的环境中同时运行的最大工作线程数。默认情况下，您可以指定最大值，最大值为 25。

线程计数

1. 在亚马逊 MWAA 控制台上打开 [环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 选择下一步。
5. 在“环境类”窗格上，在“最大工作器数”中输入一个值。
6. 选择保存。

Note

更改可能需要几分钟程计数。

高性能用例示例

下一节描述了可用于在环境中实现高性能和并行性的配置类型。

Apache

通常，在本地 Apache Airflow 平台中，您将在 `airflow.cfg` 文件中配置任务并行度、自动缩放和并发设置：

- `core.parallelism`— 可以在整个环境中 `parallel` 运行的最大任务实例数。
- `core.dag_concurrency`— DAG（非工作服务器）的最大并发量。
- `celery.worker_autoscale`— 可以在任何工作器上同时运行的最大和最小任务数。

例如，如果设置 `core.parallelism` 为 100 且 `core.dag_concurrency` 设置为 7，则只有当您有 2 个 DAG 时，您仍然能够同时运行总共的 14 任务。7 假定，即使总体并行度设置为（输入 `core.dag_concurrency`），每个 DAG 仍设置为仅同时运行七个任务 100（输入 `core.parallelism`）。

在 Amazon MWAA 环境中

在 Amazon MWAA 环境中，您可以使用 [在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)、和最大工作线程数自动扩展机制直接在 Amazon MWAA 控制台上配置这些设置。 [配置个 Amazon MWAA 环境](#)

[类 \(p. 85\)](#)虽然`core.dag_concurrency`在亚马逊 MWAA 控制台的下拉列表中不可用 Apache Airflow 配置选项，但您可以将其添加为自定义 [Apache Airflow 配置选项 \(p. 90\)](#)。

假设在创建环境时，您选择了以下设置：

1. `mw1.small` [环境类 \(p. 85\)](#)控制每个工作程序默认可以运行的最大并发任务数以及容器的 vCPU。
2. 最大**10**工作人员人数中的 Workers 的默认设置。
3. [Apache Airflow 配置选项 \(p. 90\)](#)`celery.worker_autoscale`，用于每个工作人员的5,5任务。

这意味着您可以在您的环境中运行 50 个并发任务。任何超过 50 的任务都将排入队列，等待正在运行的任务完成。

运行更多并发任务。您可以使用以下配置修改您的环境以同时运行更多任务：

1. 通过选择`mw1.medium`（默认情况下为 10 个并发任务）[环境类 \(p. 85\)](#)，增加每个工作线程可以运行的最大并发任务数和容器的 vCPU。
2. `celery.worker_autoscale`作为 [Apache Airflow 配置选项 \(p. 90\)](#)添加。
3. 增加最大工作人员人数。在此示例中，将最大工作线程数从增加10到20将使环境可以运行的并发任务数增加一倍。

指定最低员工人数。您还可以使用AWS Command Line Interface (AWS CLI) 指定在您的环境中运行的 Apache Airflow Worker 的最小和最大数量。例如：

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

要了解更多信息，请参阅中的[更新环境](#)命令AWS CLI。

对卡在运行状态的任务进行故障排除

在极少数情况下，Apache Airflow 可能会认为还有任务在运行。要解决此问题，你需要清除 Apache Airflow 用户界面中的搁浅任务。有关更多信息，请参阅[我看到我的任务卡住或没完成 \(p. 304\)](#)故障程主题。

接下来做什么？

- 详细了解我们推荐用于调整环境性能的最佳实践[亚马逊 MWAA 上的 Apache Airflow 的性能调整 \(p. 238\)](#)。

在亚马逊 MWAA 上使用 Apache Airflow 配置选项

Amazon Managed Workflows for Apache Airflow (MWAA) 环境。您可以从建议的下拉列表中进行选择，也可以在 Amazon MWAA 控制台上为您的 Apache Airflow 版本指定自定义配置选项。本页介绍可用的 Apache Airflow 配置选项，以及如何使用这些选项来覆盖环境中的 Apache Airflow 配置设置。

目录

- [先决条件 \(p. 91\)](#)
- [工作原理 \(p. 91\)](#)
- [在 Apache Airflow v2 中使用配置选项加载插件 \(p. 91\)](#)
- [配置桶概述 \(p. 91\)](#)
 - [Apache Airflow \(p. 92\)](#)
 - [Apache Airflow \(p. 92\)](#)

- [使用 Amazon Amazon MWAA 控制台 \(p. 92\)](#)
- [配置参考 \(p. 92\)](#)
 - [电子邮件配置 \(p. 92\)](#)
 - [任务配置 \(p. 93\)](#)
 - [调度器配置 \(p. 94\)](#)
 - [工作线程的配置 \(p. 95\)](#)
 - [Web 服务器桶配置 \(p. 95\)](#)
- [示例和示例代码 \(p. 96\)](#)
 - [示例 DAG \(p. 96\)](#)
 - [电子邮件通知设置示例 \(p. 92\)](#)
- [接下来做什么？ \(p. 97\)](#)

先决条件

您需要先完成以下内容才能完成此页面上的步骤。

- 权限 — 您的AWS账户必须已获得管理员对您环境的 [AmazonmwaaFullConsoleAccess \(p. 33\)](#) 访问控制政策的访问权限。此外，您的[执行角色 \(p. 44\)](#)必须允许您的 Amazon MWAA 环境访问您的环境所使用的AWS资源。
- 访问权限 — 如果您需要访问公共存储库才能直接在 Web 服务器上安装依赖项，则您的环境必须配置为具有公共网络 Web 服务器访问权限。有关更多信息，请参阅[the section called “Apache Airflow” \(p. 52\)](#)：
- Amazon S3 配置 — 用于存储 D [AG、中的plugins.zip自定义插件和 Python 依赖项的 Amazon S3 存储桶 \(p. 16\)](#)requirements.txt必须配置为禁止公共访问和启用版本控制。

工作原理

当您创建环境时，Amazon MWAA 会将您在 Amazon MWAA 控制台的 Airflow 配置选项中指定的配置设置作为环境变量附加到您的环境的AWS Fargate容器中。如果您在中使用同名设置，则您在airflow.cfg Amazon MWAA 控制台上指定的选项将覆盖中的值airflow.cfg。

虽然我们不会airflow.cfg在亚马逊 MWAA 环境的 Apache Airflow 用户界面中公开，但您可以直接在 Amazon MWAA 控制台上更改 Apache Airflow 配置选项，然后继续使用中的所有其他设置airflow.cfg。

在 Apache Airflow v2 中使用配置选项加载插件

默认情况下，在 Apache Airflow v2 中，使用该core.lazy_load_plugins : True设置将插件配置为“延迟”加载。如果您在 Apache Airflow v2 中使用自定义插件，则必须添加core.lazy_load_plugins : False为 Apache Airflow 配置选项，以便在每个 Airflow 进程开始时加载插件，以覆盖默认设置。

配置桶概述

当您在 Amazon MWAA 控制台上添加配置时，Amazon MWAA 将该配置写入环境变量。

- 列出的选项。您可以在下拉列表中选择适用于 Apache Airflow 版本的配置设置之一。例如，dag_concurrency:16。配置设置将转换为您环境的 Fargate 容器AIRFLOW__CORE__DAG_CONCURRENCY : 16
- 自定义选项。您还可以在下拉列表中指定 Apache Airflow 版本中未列出的 Airflow 配置选项。例如，foo.user:YOUR_USER_NAME。配置设置将转换为您环境的 Fargate 容器AIRFLOW__FOO__USER : YOUR_USER_NAME

Apache Airflow

下图显示了您可以在亚马逊 MWAA 控制台上自定义 Apache Airflow 配置选项的位置。

Airflow configuration options - optional [Info](#)
Modify the default settings for Airflow configuration options. You can select an option from the suggestion list or type one manually.

All Airflow configuration options are using default values.

Add custom configuration value

Apache Airflow

有关 Apache Airflow 支持的配置选项列表，请参阅 Apache Airflow 参考指南中的[配置参考](#)。要查看您在亚马逊 MWAA 上运行的 Apache Airflow 版本的选项，请从下拉列表中选择该版本。

使用 Amazon Amazon MWAA 控制台

以下过程将指导您完成向环境中添加 Airflow 配置选项的步骤。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 选择下一步。
5. 在 Airflow 配置选项窗格中选择添加自定义配置。
6. 从下拉列表中选择一个配置并输入一个值，或者键入自定义配置并输入一个值。
7. 为每个要添加的配置选择 Add 自定义配置。
8. 选择保存。

配置参考

以下部分包含亚马逊 MWAA 控制台下拉列表中可用的 Apache Airflow 配置列表。

电子邮件配置

以下列表显示了 Amazon MWAA 上可用的 Airflow 电子邮件通知配置选项。

我们建议使用端口 587 进行 SMTP 流量。默认情况下，会AWS阻塞所有 Amazon EC2 实例的端口 25 上的出站 SMTP 流量。如果您想在端口 25 上发送出站流量，可以[请求取消此限制](#)。

Apache Airflow v2

气流版本	气流桶选项	描述	示例值
v2	电子邮件.backend.email_backend	Apache Airflow 应用程序用于在 email_backend 中发送电子邮件通知 。	airflow.utils.email.send_email_smtp

气流版本	气流桶选项	描述	示例值
v2	smtp.smtp_host	用作 smtp_host 中电子邮件地址的出站服务器的名称。	localhost
v2	smtp.smtp_starttls	在 smtp_starttls 中， 传输层安全 (TLS) 用于加密通过互联网发送的电子邮件。	False
v2	smtp.smtp_ssl	安全套接字层 (SSL) 用于连接 smtp_ssl 中的服务器和电子邮件客户端。	True
v2	smtp.smtp_port	在 smtp_port 中指定给服务器的传输桶控制协议 (TCTC) 端口。	587
v2	smtp.smtp_mail_from	smtp_mail_from 中的出站电子邮件地址。	myemail@domain.com

Apache Airflow v1

气流版本	气流桶选项	描述	示例值
v1	电子邮件. email_backend	Apache Airflow 实用程序用于在 email_backend 中发送电子邮件通知。	airflow.utils.email.send_email_smtp
v1	smtp.smtp_host	用作 smtp_host 中电子邮件地址的出站服务器的名称。	localhost
v1	smtp.smtp_starttls	在 smtp_starttls 中， 传输层安全 (TLS) 用于加密通过互联网发送的电子邮件。	False
v1	smtp.smtp_ssl	安全套接字层 (SSL) 用于连接 smtp_ssl 中的服务器和电子邮件客户端。	True
v1	smtp.smtp_port	在 smtp_port 中指定给服务器的传输桶控制协议 (TCTC) 端口。	587
v1	smtp.smtp_mail_from	smtp_mail_from 中的出站电子邮件地址。	myemail@domain.com

任务配置

以下列表显示了 Amazon MWAA 上的 Airflow 任务下拉列表中的可用配置。

Apache Airflow v2

气流版本	气流桶选项	描述	示例值
v2	core.default_task_rets	在 de fault_task_retries 中重试 Apache Airflow 任务的次数 。	3
v2	核心。并行性	可以在整个环境中并行运行的最大任务实例数（ 并行度 ）。	40

Apache Airflow v1

气流版本	气流桶选项	描述	示例值
v1	core.default_task_rets	在 de fault_task_retries 中重试 Apache Airflow 任务的次数 。	3
v1	核心。并行性	可以在整个环境中并行运行的最大任务实例数（ 并行度 ）。	40

调度器配置

以下列表显示了 Amazon MWAA 下拉列表中可用的 Airflow 调度程序配置。

Apache Airflow v2

气流版本	气流桶选项	描述	示例值
v2	调度器.catchup_by_default_	告诉调度程序创建 DAG 运行以“catch”到 catchup_by_default 中的特定时间间隔。	False
v2	scheduler.scheduler_zombie_task_threshold	告诉调度程序是否将任务实例标记为失败，然后在 scheduler_zombie_task_threshold 中重新安排任务。	300

Apache Airflow v1

气流版本	气流桶选项	描述	示例值
v1	调度器.catchup_by_default_	告诉调度程序创建 DAG 运行以“catch”到 catchup_by_default 中的特定时间间隔。	False

气流版本	气流桶选项	描述	示例值
v1	scheduler.scheduler_zombie_task_threshold	告诉调度程序是否将任务实例标记为失败，然后在 scheduler_zombie_task_threshold 中重新安排任务。	300

工作线程的配置

以下列表显示了 Amazon MWAA 下拉列表中可用的 Airflow 工作人员配置。

Apache Airflow v2

气流版本	气流桶选项	描述	示例值
v2	celery.worker_autoScale	使用 worker autoscale 中的 Celery Executor 可以在任何工作器上同时运行的最大和最小任务数。值必须按以下顺序以逗号分隔：max_concurrency,min_concurrency。	16,12

Apache Airflow v1

气流版本	气流桶选项	描述	示例值
v1	celery.worker_autoScale	使用 worker autoscale 中的 Celery Executor 可以在任何工作器上同时运行的最大和最小任务数。值必须按以下顺序以逗号分隔：max_concurrency,min_concurrency。	16,12

Web 服务器桶配置

以下列表显示了亚马逊 MWAA 下拉列表中可用的 Airflow Web 服务器配置。

Apache Airflow v2

气流版本	气流桶选项	描述	示例值
v2	Web 服务器.default_ui_timezone	default_ui_timezone 中的默认 Apache Airflow UI 日期时间设置。 Note 设置该default_ui_timezone选	America/New_York

气流版本	气流桶选项	描述	示例值
		项不会更改 DAG 计划运行的时区。要更改 DAG 的时区。有关更多信息，请参阅 the section called “更改 DAG 的时区” (p. 198) ：	

Apache Airflow v1

气流版本	气流桶选项	描述	示例值
v1	Web 服务 器.default_ui_timezone	default it ui_timezone 中的默认 Apache Airflow UI 日期时间 设置。	America/New_York

示例和示例代码

示例 DAG

你可以使用以下 DAG 来打印你的 Aemail_backend pache Airflow 配置选项。要运行以响应 Amazon MWAA 事件，请将代码复制到 Amazon S3 存储桶上环境的 DAG 文件夹。

```
def print_var(**kwargs):
    email_backend = kwargs['conf'].get(section='email', key='email_backend')
    print("email_backend")
    return email_backend
with DAG(dag_id="email_backend_dag", schedule_interval="@once", default_args=default_args,
    catchup=False) as dag:
    email_backend_test = PythonOperator(
        task_id="email_backend_test",
        python_callable=print_var,
        provide_context=True
    )
```

电子邮件通知设置示例

以下 Apache Airflow 配置选项可用于使用应用程序密码的 Gmail.com 电子邮件帐户。有关更多信息，请参阅 Gmail 帮助参考指南[中的使用应用程序密码登录](#)。

Airflow configuration options - optional [Info](#)

Modify the default settings for Airflow configuration options. You can select an option from the suggestion list or type one manually.

Configuration option	Custom value	
smtp.smtp_host X	smtp.gmail.com	<button>Remove</button>
smtp.smtp_mail_from X	<your email>@gmail.com	<button>Remove</button>
smtp.smtp_password X	<your 16 digit app password>	<button>Remove</button>
smtp.smtp_port X	587	<button>Remove</button>
smtp.smtp_ssl X	False	<button>Remove</button>
smtp.smtp_starttls X	True	<button>Remove</button>
smtp.smtp_user X	<your email>@gmail.com	<button>Remove</button>
<button>Add custom configuration value</button>		

接下来做什么？

- 要了解如何将您的 Amazon S3 存储桶 [添加或更新 DAG \(p. 103\)](#)。

在亚马逊 MWAA 上使用启动脚本

启动脚本是您在环境的 Amazon S3 存储桶中托管的 shell (.sh) 脚本，类似于 DAG、要求和插件。在安装要求和初始化 Apache Airflow 流程之前，Amazon MWAA 在启动期间在每个单独的 Apache Airflow 组件（工作程序、调度程序和 Web 服务器）上运行此脚本。使用启动脚本执行以下操作：

- 安装运行时-安装工作流程和连接所需的 Linux 运行时。
- 配置环境变量-为每个 Apache Airflow 组件设置环境变量。重写 PATH、PYTHONPATH 和等常见变量 LD_LIBRARY_PATH。
- 管理密钥和令牌-将自定义存储库的访问令牌传递给安全密钥 requirements.txt 并配置安全密钥。

以下主题介绍如何配置启动脚本以安装 Linux 运行时、设置环境变量以及使用 CloudWatch 日志解决相关问题。

主题

- [配置启动脚本 \(p. 97\)](#)
- [使用启动脚本安装 Linux 运行时 \(p. 100\)](#)
- [使用启动脚本设置环境变量 \(p. 100\)](#)

配置启动脚本

要在现有 Amazon MWAA 环境中使用启动脚本，请将 .sh 文件上传到您的环境的 Amazon S3 存储桶。然后，要将脚本与环境关联，请在您的环境详细信息中指定以下内容：

- 脚本的 Amazon S3 URL 路径 — 存储桶中托管的脚本的相对路径，例如 s3://mwaa-environment/**startup.sh**

- 脚本的 Amazon S3 版本 ID — 您的 Amazon S3 存储桶中启动外壳脚本的版本。每次更新脚本时，都必须指定 Amazon S3 为文件分配的 [版本 ID](#)。版本 ID 是 Unicode、UTF-8 编码、URL 就绪、不透明的字符串，长度不超过 1,024 字节 3sL4kqtJlcpXroDTdmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo。

要完成本节中的步骤，请使用以下示例脚本。该脚本输出分配给的值 MWAA_AIRFLOW_COMPONENT。此环境变量标识了运行脚本的每个 Apache Airflow 组件。

复制代码并将其在本地另存为 startup.sh。

```
#!/bin/sh

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

接下来，将脚本上传到 Amazon S3 存储桶。

AWS Management Console

上传 shell 脚本（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 从 Buckets（桶）列表中，请选择与环境关联的存储桶的名称。
3. 在 Objects（对象）选项卡上，选择 Upload（上传）。
4. 在上传页面上，拖放您创建的 shell 脚本。
5. 请选择 Upload（上传）。

该脚本出现在对象列表中。Amazon S3 将为文件创建新版本 ID。如果您更新脚本并使用相同的文件名再次上传，则会为该文件分配新的版本 ID。

AWS CLI

创建和上传 shell 脚本 (CLI)

1. 打开新的命令提示符，然后运行 Amazon S3ls 命令以列出并识别与您的环境相关的存储桶。

```
$ aws s3 ls
```

2. 导航到保存 shell 脚本的文件夹。cp 在新的提示窗口中使用将脚本上传到您的存储桶。用 **#####** **#-s3-buck** et。

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

如果成功，Amazon S3 会输出对象的 URL 路径：

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

3. 使用以下命令检索脚本的最新版本 ID。

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

```
BbdVMmBRjttestta1EsVnbybZp1Wqh1J4
```

将脚本与环境关联时，可以指定此版本 ID。

现在，将脚本与您的环境相关联。

AWS Management Console

将脚本与环境 (控制台) 相关联 (控制台)

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择要更新的环境的行，然后选择编辑。
3. 在指定详细信息页面上，对于启动脚本文件-可选，输入脚本的 Amazon S3 URL，例如：`s3://your-mwaa-bucket/startup-sh.`。
4. 从下拉列表中选择最新版本，或浏览S3查找脚本。
5. 选择“下一步”，然后前往“查看并保存”页面。
6. 查看更改，然后选择“保存”。

环境更新可能需要 10 到 30 分钟。Amazon MWAA 会在环境中的每个组件重新启动时运行启动脚本。

AWS CLI

将脚本与环境关联 (CLI)

- 打开命令提示符并使用该脚本`update-environment`指定 Amazon S3 URL 和版本 ID。

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

如果成功，Amazon MWAirflow 将返回环境的 Amazon 资源名称 (ARN)：

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

环境更新可能需要 10 到 30 分钟。Amazon MWAA 会在环境中的每个组件重新启动时运行启动脚本。

最后，检索日志事件以验证脚本是否按预期运行。当您为每个 Apache Airflow 组件网络激活日志记录时，Amazon MWAA 会创建一个新的日志组和日志流。有关更多信息，请参阅 [Apache Airflow Airflo \(p. 258\)](#) w。

AWS Management Console

查看 Apache Airflow 日志流 (控制台)

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择您的环境。
3. 在“监控”窗格中，选择要查看其日志的日志组，例如 Airflow 调度程序日志组。
4. 在 CloudWatch 控制台中，从日志流列表中选择带有以下前缀的流：`startup_script_execution_ip`。
5. 在日志事件窗格上，您将看到打印值的命令输出MWAA_AIRFLOW_COMPONENT。例如，对于调度程序日志，您将执行以下操作：

```
Printing Apache Airflow component
scheduler
```

```
Finished running startup script. Execution time: 0.004s.  
Running verification  
Verification completed
```

您可以重复前面的步骤来查看工作和 Web 服务器日志。

使用启动脚本安装 Linux 运行时

使用启动脚本更新 Apache Airflow 组件的操作系统，并安装其他运行时库以用于您的工作流程。例如，运行以下脚本 `yum update` 以更新操作系统。

Note

用于运行 `sudo` 需要管理权限的操作。

```
#!/bin/sh  
  
echo "Updating operating system"  
sudo yum update -y
```

要在特定的 Apache Airflow 组件上安装运行时，请使用 `MWAA_AIRFLOW_COMPONENT` `if` 和 `fi` 条件语句。此示例运行单个命令将 `libaio` 库安装在调度程序和工作程序上，但不在 Web 服务器上。

Important

如果您配置了 [专用 Web 服务器 \(p. 56\)](#)，则必须使用以下条件或在本地提供所有安装文件，以避免安装超时。

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

您可以使用启动脚本来检查 Python 版本。

```
#!/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"
```

但是，您无法使用该脚本安装其他版本的 Python。Amazon MWAA 可防止您覆盖 Python 版本以确保与运行环境所需的依赖项兼容。

使用启动脚本设置环境变量

使用启动脚本设置环境变量和修改 Apache Airflow 配置。以下定义了一个新变量 `ENVIRONMENT_STAGE`。您可以在 DAG 或自定义模块中引用此变量。

```
#!/bin/sh  
  
export ENVIRONMENT_STAGE="development"  
echo "$ENVIRONMENT_STAGE"
```


使用启动脚本覆盖常见的 Apache Airflow 或系统变量。例如，您设置LD_LIBRARY_PATH为指示 Python 在您指定的路径中查找二进制文件。这使您可以使用[插件](#)为工作流程提供自定义二进制文件：

```
#!/bin/sh

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

预留环境变量

亚马逊 MWAA 保留了一组关键环境变量。如果您改写了保留变量，Amazon MWAA 会将其恢复为默认值。以下列出了保留变量：

- MWAA__AIRFLOW__COMPONENT— 用于使用以下值之一识别 Apache Airflow 组件：schedulerworker、或webserver。
- AIRFLOW__WEBSERVER__SECRET_KEY— 用于在 Apache Airflow Web 服务器中安全签署会话 cookie 的密钥。
- AIRFLOW__CORE__FERNET_KEY— 用于加密和解密存储在元数据数据库中的敏感数据的密钥，例如连接密码。
- AIRFLOW_HOME— Apache Airflow 主目录的路径，配置文件和 DAG 文件存储在本地。
- AIRFLOW__CELERY__BROKER_URL— 用于 Apache Airflow 调度程序与 Celery 工作节点之间通信的消息代理的 URL。
- AIRFLOW__CELERY__RESULT_BACKEND— 用于存储 Celery 任务结果的数据库的 URL。
- AIRFLOW__CORE__EXECUTOR— Apache Airflow 应该使用的执行器类。在亚马逊 MWAA 中，这是CeleryExecutor
- AIRFLOW__CORE__LOAD_EXAMPLES— 用于激活或停用示例 DAG 的加载。
- SQL_ALCHEMY_CONN— 用于在亚马逊 MWAA 中存储 Apache Airflow 元数据的 RDS for PostgreSQL 数据库的连接字符串。
- AIRFLOW__CORE__SQL_ALCHEMY_CONN— 使用目的与新的 Apache AirSQL_ALCHEMY_CONN low 命名规则相同，但遵循新的 Apache Airflow
- AIRFLOW__CELERY__DEFAULT_QUEUE— Apache Airflow 中 Celery 任务的默认队列。
- AIRFLOW__OPERATORS__DEFAULT_QUEUE— 使用特定 Apache Airflow 操作员执行任务的默认队列。
- AIRFLOW_VERSION— 安装在亚马逊 MWAA 环境中的 Apache Airflow 版本。
- AIRFLOW_CONN_AWS_DEFAULT— 用于与中的其他AWS服务集成的默认AWS凭据。
- AWS_DEFAULT_REGION— 设置与默认凭据一起使用的默认AWS区域，以便与其他AWS服务集成。
- AWS_REGION— 如果已定义此环境变量，它将覆盖环境变量AWS_DEFAULT_REGION和配置文件设置region 中的值。
- PYTHONUNBUFFERED— 用于发送stdout和stderr流式传输到容器日志。
- AIRFLOW__METRICS__STATSD_ALLOW_LIST— 用于配置以逗号分隔的前缀的允许列表，以发送以列表元素开头的指标。
- AIRFLOW__METRICS__STATSD_ON— 激活向发送指标StatsD。
- AIRFLOW__METRICS__STATSD_HOST— 用于连接到StatSD守护程序。
- AIRFLOW__METRICS__STATSD_PORT— 用于连接到StatSD守护程序。
- AIRFLOW__METRICS__STATSD_PREFIX— 用于连接到StatSD守护程序。
- AIRFLOW__CELERY__WORKER_AUTOSCALE— 设置最大和最小并发量。
- AIRFLOW__CORE__DAG_CONCURRENCY— 设置调度程序可以在一个 DAG 中同时运行的任务实例的数量。
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG—设置每个 DAG 的最大活动任务数。
- AIRFLOW__CORE__PARALLELISM— 定义可以同时运行的最大任务实例数。
- AIRFLOW__SCHEDULER__MAX_THREADS— 设置调度程序生成的最大线程数，用于调度 DAG。

- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT`— 定义工作人员在将消息重新传送给其他工作人员之前等待确认任务的秒数。
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION`— 为底层 Celery 传输设置AWS区域。
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES`— 为底层 Celery 传输设置队列。
- `AIRFLOW__WEBSERVER__BASE_URL`— 用于托管 Apache Airflow 用户界面的网络服务器的网址。

非预留环境变量

您可以使用启动脚本来覆盖未预留的环境变量。下面列出了其中某些常用变量：

- `PATH`— 指定操作系统在其中搜索可执行文件和脚本的目录列表。当命令在命令行中运行时，系统会检查目录`PATH`以查找和执行该命令。在 Apache Airflow 中创建自定义运算符或任务时，可能需要依赖外部脚本或可执行文件。如果包含这些文件的目录不在`PATH`变量中指定的目录中，则当系统找不到任务时，任务将无法运行。通过向中添添加相应的目录`PATH`，Apache Airflow 任务可以找到并运行所需的可执行文件。
- `PYTHONPATH`— Python 解释器使用它来确定要在哪些目录中搜索导入的模块和包。它是您可以添加到默认搜索路径的目录列表。这使解释器可以查找和加载未包含在标准库中或安装在系统目录中的 Python 库。使用此变量添加您的模块和自定义 Python 包，并将它们与 DAG 一起使用。
- `LD_LIBRARY_PATH`— Linux 中的动态链接器和加载器用来查找和加载共享库的环境变量。它指定了包含共享库的目录列表，在默认系统库目录之前搜索共享库。使用此变量指定您的自定义二进制文件。
- `CLASSPATH`— 由 Java 运行时环境 (JRE) 和 Java 开发工具包 (JDK) 用于在运行时查找和加载 Java 类、库和资源。它是一个包含已编译的 Java 代码的目录、JAR 文件和 ZIP 存档的列表。

在亚马逊上使用 DAG MWAA

要在 Amazon Apache Airflow 管理工作流 (MWAA) 环境上运行定向非循环图 (DAG)，您可以将文件复制到附加到环境的 Amazon S3 存储桶中，然后让 Amazon MWAA 知道您的 DAG 和支持文件在 Amazon MWAA 控制台上的位置。Amazon MWAA 负责在工作人员、计划程序和 Web 服务器之间同步 DAG。本指南介绍了如何添加或更新 DAG，以及如何在 Amazon MWAA 环境上安装自定义插件和 Python 依赖项。

主题

- [Amazon S3 存储桶概述 \(p. 103\)](#)
- [添加或更新 DAG \(p. 103\)](#)
- [安装自定义插件 \(p. 106\)](#)
- [安装 Python 的 \(p. 117\)](#)
- [删除 Amazon S3 上的文件 \(p. 122\)](#)

Amazon S3 存储桶概述

Amazon MWAA 环境的 Amazon S3 存储桶必须具有阻止公有访问。默认情况下，所有 Amazon S3 资源都是私有的，包括存储桶、对象和相关子资源（例如，生命周期配置）。

- 只有资源所有者，AWS 创建了存储桶的账户可以访问该资源。资源所有者（例如，您的管理员）可以通过编写访问控制策略授予他人访问权限。
- 您必须被授予访问权限才能添加 DAG 和自定义插件的访问控制策略 `plugins.zip`，以及中的 Python 依赖项 `requirements.txt` 到您的 Amazon S3 存储桶是 [亚马逊 WaafLL 控制台完全访问权限 \(p. 33\)](#)。

Amazon MWAA 环境的 Amazon S3 存储桶必须具有启用版本控制。启用 Amazon S3 存储桶版本控制后，无论何时创建新版本，都会创建一个新副本。

- 已为中的自定义插件启用了版本控制 `plugins.zip`，以及中的 Python 依赖关系 `requirements.txt` 在您的 Amazon S3 存储桶上。
- 您还必须指定 `plugins.zip` 和 `requirements.txt` 每次在 Amazon S3 存储桶上更新这些文件时，都会在 Amazon MWAA 控制台上进行。

添加或更新 DAG

有向无环图 (DAG) 在 Python 文件中定义，该文件将 DAG 的结构定义为代码。您可以使用 AWS CLI、或 Amazon S3 控制台将 DAG 上载到您的环境。本页介绍使用 Amazon S3 存储桶中的 `dags` 文件夹在适用于 Apache Airflow (MWAA) 的亚马逊托管工作流程 (MWAA) 环境上添加或更新 Apache Airflow DAG 的步骤。

小节目录

- [先决条件 \(p. 104\)](#)
- [工作原理 \(p. 104\)](#)
- [v2 中有什么变化 \(p. 104\)](#)
- [使用亚马逊 MWAA CLI 实用程序测试 DAG \(p. 105\)](#)
- [上载 DAG 代码到 Amazon S3 \(p. 105\)](#)

- [在亚马逊 MWAA 控制台上指定 DAG 文件夹的路径 \(第一次\) \(p. 106\)](#)
- [在 Apache Airflow UI 上查看更改 \(p. 106\)](#)
- [接下来做什么? \(p. 106\)](#)

先决条件

您需要先完成以下内容才能完成此页面上的步骤。

- 权限 — 您的AWS账户必须已获得管理员对您环境的 [AmazonmwaaFullConsoleAccess \(p. 33\)](#) 访问控制政策的访问权限。此外，您的[执行角色 \(p. 44\)](#)必须允许您的 Amazon MWAA 环境访问您的环境所使用的 AWS资源。
- 访问权限 — 如果您需要访问公共存储库才能直接在 Web 服务器上安装依赖项，则您的环境必须配置为具有公共网络 Web 服务器访问权限。有关更多信息，请参阅[the section called “Apache Airflow” \(p. 52\)](#)：
- Amazon S3 配置 — 用于存储 DAG、中的[plugins.zip自定义插件和 Python 依赖项的 Amazon S3 存储桶 \(p. 16\)](#)requirements.txt必须配置为禁止公共访问和启用版本控制。

工作原理

有向无环图 (DAG) 在单个 Python 文件中定义，该文件将 DAG 的结构定义为代码。它由以下内容组成：

- 一个 [DAG](#) 的定义。
- 描述如何运行 DAG 和要运行的[任务](#)的运@@ [算符](#)。
- 描述任务运行顺序的@@ [操作员关系](#)。

要在 Amazon MWAA 环境中运行 Apache Airflow 平台，您需要将 DAG 定义复制到存储桶中的dags文件夹。例如，存储桶中的 DAG 文件夹可能如下所示：

Example DAG 文件夹

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

Amazon MWAA 每 30 秒自动将新的和更改的对象从 Amazon S3 存储桶同步到 Amazon MWAA 调度程序和工作容器的文件/usr/local/airflow/dags夹，从而保留 Amazon S3 源的文件层次结构，无论文件类型如何。新 DAG 出现在你的 Apache Airflow 用户界面中的时间由以下因素控制[scheduler.dag_dir_list_interval \(p. 241\)](#)。对现有 DAG 的更改将在下一个 [DAG 处理循环 \(p. 239\)](#)中获取。

Note

您不需要在 DAG 文件夹中包含airflow.cfg配置文件。你可以从亚马逊 MWAA 控制台替换默认 Apache Airflow 配置。有关更多信息，请参阅[在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)：

v2 中有什么变化

- 新增：运算符、挂钩和执行器。在 Apache Airflow v1 和 Apache Airflow v2 之间，你的 DAG 中的导入语句以及你在亚马逊 MWAAplugins.zip 上指定的自定义插件发生了变化。例如，from airflow.contrib.hooks.aws_hook import AwsHook在 Apache Airflow v1中from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook 中，Apache Airflow v1 已更改为。要了解更多信息，请参阅 Apache Airflow [参考指南中的 Python API](#) 参考文档。

使用亚马逊 MWAA CLI 实用程序测试 DAG

- 命令行界面 (CLI) 实用程序会在本地复制 Amazon Managed Workflows for Apache Airflow (MWAA) 环境。
- CLI 在本地构建 Docker 容器映像，类似于亚马逊 MWAA 生产映像。这允许您在部署到 Amazon MWAA 之前运行本地 Apache Airflow 环境来开发和测试 DAG、自定义插件和依赖关系。
- 要运行 CLI，请参阅[aws-mwaa-local-runner](#)上的 GitHub。

上载 DAG 代码到 Amazon S3

您可以使用 Amazon S3 控制台或 AWS Command Line Interface (AWS CLI) 将 DAG 代码上载到 Amazon S3 存储桶。以下步骤假设您正在将代码 (.py) 上传到 Amazon S3 存储桶 dags 中命名的文件夹。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下条件：

- [AWS CLI— 安装版本 2](#)。
- [AWS CLI— 使用 aws configure 进行快速配置](#)。

要使用上传 AWS CLI

1. 使用以下命令列出您的所有 Amazon S3 存储桶。

```
aws s3 ls
```

2. 使用以下命令列出环境的 Amazon S3 存储桶中的文件和文件夹。

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

3. 以下命令将文件上传到 dag_def.py 文件 dags 夹。

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

如果您的 Amazon S3 存储桶中尚不存在名为 dags 的文件夹，则此命令会创建该 dags 文件夹并将名为 dag_def.py 的文件上传到新文件夹。

使用 Amazon S3 控制台

Amazon S3 控制台是基于 Web 的用户界面，允许您在 Amazon S3 存储桶中创建和管理。以下步骤假设您有一个名为 DAGs 文件夹 dags。

使用 Amazon S3 控制台上传

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在 S3 窗格中的 DAG 代码中选择 S3 存储段链接，在 Amazon S3 控制台上打开您的存储桶。
4. 选择 dags 文件夹。
5. 请选择 Upload (上传)。
6. 选择“添加文件”。

7. 选择您的本地副本dag_def.py，选择上传。

在亚马逊 MWAA 控制台上指定 DAG 文件夹的路径 (第一次)

以下步骤假设您指定的 Amazon S3 存储桶中名为的文件夹的路径dags。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择要在其中运行 DAG 的环境。
3. 选择 Edit (编辑)。
4. 在 Amazon S3 窗格中的 DAG 代码上，选择 DAG 文件夹字段旁边的浏览 S3。
5. 选择您的dags文件夹。
6. 选择 Choose (选择)。
7. 选择下一步，更新环境。

在 Apache Airflow UI 上查看更改

登录 Apache Airfl

你需要在AWS Identity and Access Management (IAM) 中拥有AWS账户的[Apache Airflow UI 访问政策 : AmazonmwaWebServerAccess \(p. 38\)](#)权限才能查看你的 Apache Airflow 用户界面。

访问 Apache Airflow UI

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择打开 Airflow UI。

接下来做什么？

- 使用 on 在本地测试您的 DAG、自定义插件和 Python 依赖关系 GitHub。 [aws-mwaa-local-runner](#)

安装自定义插件

适用于 Apache Airflow 的亚马逊托管工作流 (MWAA) 支持 Apache Airflow 的内置插件管理器，允许您使用自定义 Apache Airflow 运算符、挂钩、传感器或接口。本页介绍安装步骤[Apache 气流自定义插件](#)在您的亚马逊 MWAA 环境中使用plugins.zip文件。

目录

- [先决条件 \(p. 107\)](#)
- [工作方式 \(p. 107\)](#)
- [v2 中有什么变化 \(p. 107\)](#)
- [自定义插件概述 \(p. 108\)](#)
 - [自定义插件目录和大小限制 \(p. 108\)](#)
- [自定义插件示例 \(p. 108\)](#)

- [在 plugins.zip 中使用平面目录结构的示例 \(p. 108\)](#)
- [在 plugins.zip 中使用嵌套目录结构的示例 \(p. 110\)](#)
- [创建 plugins.zip 文件 \(p. 114\)](#)
 - [步骤一：使用亚马逊 MWAA CLI 实用程序测试自定义插件 \(p. 114\)](#)
 - [第二步：创建 plugins.zip 文件 \(p. 115\)](#)
- [上传plugins.zip到 Amazon S3 \(p. 115\)](#)
 - [使用 AWS CLI \(p. 115\)](#)
 - [使用 Amazon S3 控制台 \(p. 116\)](#)
- [在您的环境中安装自定义插件 \(p. 116\)](#)
 - [指定路径plugins.zip在亚马逊 MWAA 控制台上 \(第一次\) \(p. 116\)](#)
 - [指定plugins.zip亚马逊 MWAA 控制台上的版本 \(p. 116\)](#)
- [plugins.zip 的使用案例示例 \(p. 116\)](#)
- [接下来做什么？ \(p. 117\)](#)

先决条件

在完成此页面上的步骤之前，您需要具备以下内容。

- Permissions (权限)— 您的AWS账户必须已获得管理员授予的访问权限[亚马逊 MWAAFullConsoleAccess \(p. 33\)](#)您的环境的访问控制策略。此外，您的亚马逊 MWAA 环境必须获得您的[执行角色 \(p. 44\)](#)访问AWS您的环境使用的资源。
- 访问— 如果您需要访问公共存储库以直接在 Web 服务器上安装依赖项，则您的环境必须配置为公共网络Web 服务器访问。有关更多信息，请参阅 [the section called “Apache Airflow” \(p. 52\)](#)。
- Amazon S3 配置— 该[Amazon S3 存储桶 \(p. 16\)](#)用来存储你的 DAG，自定义插件plugins.zip，以及 Python 依赖关系在requirements.txt必须配置公有访问被阻止和版本控制已启用。

工作方式

要在您的环境中运行自定义插件，必须执行以下三项操作：

1. 创建plugins.zip本地文件。
2. 上传本地plugins.zip文件Amazon S3。
3. 在中指定此文件的版本插件文件” 字段，位于亚马逊 MWAA 控制台上。

Note

如果是第一次上传一个plugins.zip存储桶中，您还需要在 Amazon S3 控制台上指定文件的路径。您只需要完成此步骤一次。

v2 中有什么变化

- 新：运算符、钩子和执行器. 你的 DAG 中的导入语句，以及你在plugins.zip在亚马逊上，MWAA 在 Apache Airflow v1 和 Apache Airflow v2 之间发生了变化。例如，`from airflow.contrib.hooks.aws_hook import AwsHook`在 Apache Airflow v1 中已更改为`from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook`在 Apache Airflow v2。要了解更多信息，请参阅[Python API 参考](#)中的Apache气流引用指南。
- 新：插件中的导入. 使用导入插件中添加的运算符、传感器、挂钩`airflow.{operators,sensors,hooks}.<plugin_name>`不再支持。这些扩展应作为常规 Python 模块导入。

在 v2 及更高版本中，推荐的方法是将它们放在 DAGs 目录中，然后创建并使用 .airflowignore 文件以排除它们作为 DAG 解析。要了解更多信息，请参阅[模块管理](#)和[创建自定义运算符](#)中的 Apache 气流引用指南。

自定义插件概述

Apache Airflow 的内置插件管理器可以将外部功能集成到其核心中，只需将文件拖放到 \$AIRFLOW_HOME/plugins 文件夹。它允许您使用自定义 Apache Airflow 运算符、挂钩、传感器或接口。以下部分提供了本地开发环境中的平面和嵌套目录结构的示例，以及生成的 import 语句，这些语句决定了 plugins.zip 中的目录结构。

自定义插件目录和大小限制

Apache Airflow 计划程序和工作线程在启动时寻找自定义插件。AWS 为您的环境管理的 Fargate 容器位于 /usr/local/airflow/plugins/。

- 目录结构. 目录结构 (位于 /) 基于您的 plugins.zip 文件。例如，如果您的 plugins.zip 包含 operators 目录作为顶级目录，则该目录将被提取到 /usr/local/airflow/plugins/operators 在您的环境中。
- 大小限制. 我们建议 plugins.zip 文件小于 1 GB。a 的大小越大 plugins.zip 文件，则在环境中启动的时间越长。尽管 Amazon MWAA (Amazon MWAA) 不会限制 plugins.zip 文件，如果依赖项无法在十分钟内安装，Fargate 服务将超时并尝试将环境回滚到稳定状态。

Note

对于使用 Apache Airflow v1.10.12 或 v2.0.2 的环境，亚马逊 MWAA 会限制 Apache Airflow Web 服务器上的出站流量，并且不允许您直接在 Web 服务器上安装插件或 Python 依赖项。从 Apache Airflow v2.2.2 开始，亚马逊 MWAA 可以直接在 Web 服务器上安装插件和依赖项。

自定义插件示例

以下部分使用了 Apache 气流引用指南以显示如何构建本地开发环境。

在 plugins.zip 中使用平面目录结构的示例

Apache Airflow v2

以下示例显示了一个 plugins.zip 文件具有 Apache Airflow v2 的平面目录结构。

Example 平面目录 PythonVirtualenvOperator plugins.zip

以下示例显示 plugins.zip 文件的顶级树 PythonVirtualenvOperator 中的自定义插件 [为 Apache Airflow 创建自定义插件 PythonVirtualenvOperator \(p. 207\)](#)。

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

Example plugins/virtual_python_plugin.py

以下示例显示 PythonVirtualenvOperator 自定义插件。

```
"""
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

以下示例显示了一个plugins.zip文件，其中包含 Apache Airflow v1 的平面目录结构。

Example 平面目录 PythonVirtualenvOperator plugins.zip

以下示例显示 plugins.zip 文件的顶级树 PythonVirtualenvOperator 中的自定义插件为 [Apache Airflow 创建自定义插件 PythonVirtualenvOperator \(p. 207\)](#)。

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

Example plugins/virtual_python_plugin.py

以下示例显示 PythonVirtualenvOperator 自定义插件。

```
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'
```

在 plugins.zip 中使用嵌套目录结构的示例

Apache Airflow v2

以下示例显示了一个plugins.zip带有单独目录的文件hooks、operators，还有一个sensorsApache Airflow v2 的目录。

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
```

以下示例显示 DAG (DAG) 中的导入语句。[DAG 文件夹](#)，使用自定义插件。

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]
```

以下示例显示了自定义插件文件中需要的每个 import 语句。

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
```

按中的步骤操作。[使用亚马逊 MWAA CLI 实用程序测试自定义插件 \(p. 114\)](#)，然后[创建 plugins.zip 文件 \(p. 115\)](#)压缩内容之内您的plugins目录。例如，cd plugins。

Apache Airflow v1

以下示例显示了plugins.zip带有单独目录的文件hooks、operators，还有一个sensorsApache Airflow v1.10.12 的目录。

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
```

以下示例显示 DAG (DAG) 中的导入语句。[DAG 文件夹](#)，使用自定义插件。

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]
```

以下示例显示了自定义插件文件中需要的每个 import 语句。

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
```

按中的步骤操作。[使用亚马逊 MWAA CLI 实用程序测试自定义插件 \(p. 114\)](#)，然后[创建 plugins.zip 文件 \(p. 115\)](#)压缩内容之内您的plugins目录。例如，cd plugins。

创建 plugins.zip 文件

以下步骤描述了我们建议在本地创建 plugins.zip 文件的步骤。

步骤一：使用亚马逊 MWAA CLI 实用程序测试自定义插件

- 命令行界面 (CLI) 实用程序在本地复制了 Amazon Mankow (MWAirflow) 环境。
- CLI 会在本地构建一个 Docker 容器映像，类似于 Amazon MWAA 生产映像。这允许您在部署到 Amazon MWAA 之前运行本地 Apache Airflow 环境来开发和测试 DAG、自定义插件和依赖项。

- 要运行 CLI，请参阅[aws-mwaa-local-runner](#)上 GitHub。

第二步：创建 plugins.zip 文件

您可以使用内置的 ZIP 存档实用程序或任何其他 ZIP 实用程序（例如[7zip](#)）创建 .zip 文件。

Note

Windows 操作系统的内置 zip 实用程序可能会在创建 .zip 文件时添加子文件夹。我们建议您在上传到 Amazon S3 存储桶之前先验证 plugins.zip 文件的内容，以确保没有添加其他目录。

1. 将目录更改为本地的 Airflow 插件目录。例如：

```
myproject$ cd plugins
```

2. 运行以下命令以确保内容具有可执行权限（仅限 macOS 和 Linux）。

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

3. Zip之内您的pluginsfolder。

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

上传plugins.zip到 Amazon S3

您可以使用 Amazon S3 控制台或AWS Command Line Interface(AWS CLI) 上传一个plugins.zip文件 Amazon S3。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要满足以下条件：

- [AWS CLI- 安装版本 2.](#)
- [AWS CLI- 快速配置，使用aws configure.](#)

使用上传AWS CLI

1. 在命令提示符中，导航到您的plugins.zip文件已存储。例如：

```
cd plugins
```

2. 使用以下命令列出您的所有 Amazon S3 存储桶。

```
aws s3 ls
```

3. 使用以下命令列出您环境的 Amazon S3 存储桶中的文件和文件夹。

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

4. 使用以下命令上传plugins.zip文件Amazon S3 桶。

```
aws s3 cp plugins.zip s3://YOUR_S3_BUCKET_NAME/plugins.zip
```

使用 Amazon S3 控制台

Amazon S3 控制台是基于 Web 的用户界面，以方便您在 Amazon S3 控制台中创建和管理资源。

使用 Amazon S3 控制台上传文件

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. 在 S3 bucket 中的链接 S3 中的 DAG 代码窗格以在 Amazon S3 控制台上打开存储桶。
4. 请选择 Upload (上传)。
5. 选择添加文件。
6. 选择您的本地副本 plugins.zip，选择上传。

在您的环境中安装自定义插件

本节介绍如何安装您上传到 Amazon S3 存储桶的自定义插件，方法是指定 plugins.zip 文件的路径，并在每次更新 zip 文件时指定 plugins.zip 文件的版本。

指定路径 plugins.zip 在亚马逊 MWAA 控制台上 (第一次)

如果是第一次上传一个 plugins.zip 存储桶中，您还需要在 Amazon S3 控制台上指定文件的路径。您只需要完成此步骤一次。

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. 选择 Edit (编辑)。
4. 在存储库的 Amazon S3 中的 DAG 代码窗格中，选择浏览 S3 旁边的插件文件-可选字段中返回的子位置类型。
5. 在 S3 桶中指定 plugins.zip 的路径。
6. 选择 Choose (选择)。
7. 选择下一步、更新环境。

指定 plugins.zip 在亚马逊 MWAA 控制台上的版本

您需要指定 plugins.zip 每次您上传您的 Amazon MWAA 控制台上的文件 plugins.zip 到您的 Amazon S3 存储桶中。

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. 选择 Edit (编辑)。
4. 在存储库的 Amazon S3 中的 DAG 代码窗格中，选择 plugins.zip 版本。
5. 选择 Next (下一步)。

plugins.zip 的使用案例示例

- 了解如何在您的环境中创建自定义插件 [使用 Apache Hive 和 Hadoop 的自定义插件 \(p. 204\)](#)。
- 了解如何在您的环境中创建自定义插件 [要修补的自定义插件 PythonVirtualenvOperator \(p. 207\)](#)。

- 了解如何在创建自定义插件[使用 Oracle 自定义插件 \(p. 192\)](#).
- 了解如何在创建自定义插件[the section called “更改 DAG 的时区” \(p. 198\)](#).

接下来做什么？

- 在本地测试你的 DAG、自定义插件和 Python 依赖关系[aws-mwaa-local-runner](#)上 GitHub.

安装 Python 的

Python 依赖项是指在亚马逊 Apache Airflow 托管工作流程 (MWAA) 环境上的 Apache Airflow 版本的 Apache Airflow 基础安装中未包含的任何软件包或发行版。本页介绍使用 Amazon S3 存储桶中的 `requirements.txt` 文件在亚马逊 MWAA 环境上安装 Apache Airflow Python 依赖项的步骤。

目录

- [先决条件 \(p. 117\)](#)
- [工作原理 \(p. 117\)](#)
- [Python 依赖关系 \(p. 118\)](#)
 - [Python 依赖项的位置和大小限制 \(p. 118\)](#)
- [正在创建 requirements.txt 文件 \(p. 118\)](#)
 - [第一步：使用亚马逊 MWAA CLI 实用程序测试 Python 依赖关系 \(p. 118\)](#)
 - [步骤二：创建 requirements.txt \(p. 118\)](#)
- [上传 requirements.txt 到 Amazon S3 \(p. 120\)](#)
 - [使用 AWS CLI \(p. 120\)](#)
 - [使用 Amazon S3 控制台 \(p. 121\)](#)
- [在您的环境中安装 Python 依赖项 \(p. 121\)](#)
 - [在亚马逊 MWAA 控制台 requirements.txt 上指定路径 \(第一次\) \(p. 121\)](#)
 - [在亚马逊 MWAA 控制台上指定 requirements.txt 版本 \(p. 121\)](#)
- [查看您的日志 requirements.txt \(p. 122\)](#)
- [接下来做什么？ \(p. 122\)](#)

先决条件

您需要先完成以下内容才能完成此页面上的步骤。

- 权限 — 您的 AWS 账户必须已获得管理员对您环境的 [AmazonMwaaFullConsoleAccess \(p. 33\)](#) 访问控制策略的访问权限。此外，您的[执行角色 \(p. 44\)](#)必须允许您的 Amazon MWAA 环境访问您的环境所使用的 AWS 资源。
- 访问权限 — 如果您需要访问公共存储库才能直接在 Web 服务器上安装依赖项，则您的环境必须配置为具有公共网络 Web 服务器访问权限。有关更多信息，请参阅[the section called “Apache Airflow” \(p. 52\)](#)：
- Amazon S3 配置 — 用于存储 [DAG、中的 plugins.zip 自定义插件和 Python 依赖项的 Amazon S3 存储桶 \(p. 16\)](#) `requirements.txt` 必须配置为禁止公共访问和启用版本控制。

工作原理

在 Amazon MWAA 上，您可以安装所有 Python 依赖项，方法是将 `requirements.txt` 文件上传到您的 Amazon S3 存储桶，然后在每次更新文件时在 Amazon MWAA 控制台上指定该文件的版本。Amazon

MWAA 运行是 `pip3 install -r requirements.txt` 为了在 Apache Airflow 调度程序和每个工作程序上安装 Python 依赖项。

要在你的环境上运行 Python 依赖关系，你必须做三件事：

1. 在本地创建 `requirements.txt` 文件。
2. 将本地文件上传 `requirements.txt` 到 Amazon S3 存储桶。
3. 在 Amazon MWAA 控制台的要求文件字段中指定此文件的版本。

Note

如果这是您第一次创建并将其上传 `requirements.txt` 到 Amazon S3 存储桶，则还需要在 Amazon MWAA 控制台上指定文件的路径。您只需要完成此步骤一次。

Python 依赖关系

你可以在你的环境中安装来自 Python Package 索引 (PyPi.org)、Python 轮子 (.whl) 或托管在私有 PyPi / PEP-503 兼容存储库上的 Python 依赖项的 Apache Airflow extras 和其他 Python 依赖项。

Python 依赖项的位置和大小限制

Apache Airflow S scheduler 和 Workers 在启动期间在您的环境 AWS 托管的 Fargate 容器上查找自定义插件，网址为 `/usr/local/airflow/plugins`。

- 大小限制。我们建议使用引用合并大小小于 1 GB 的库的 `requirements.txt` 文件。Amazon MWAA 需要安装的库越多，环境中的启动时间就越长。尽管 Amazon MWAA 没有明确限制已安装库的大小，但如果无法在十分钟内安装依赖项，Fargate 服务将超时并尝试将环境回滚到稳定状态。

Note

从 Apache Airflow v2.2.2 开始，亚马逊 MWAA 将在 Airflow Web 服务器上安装 Python 要求、提供程序包和自定义插件。

正在创建 requirements.txt 文件

以下步骤描述了我们建议在本地创建 `requirements.txt` 文件的步骤。

第一步：使用亚马逊 MWAA CLI 实用程序测试 Python 依赖关系

- 命令行界面 (CI) 工具会在本地复制 Amazon Managed Workflows for Apache Airflow (MWAA) 环境。
- CLI 在本地构建 Docker 容器映像，类似于亚马逊 MWAA 生产映像。这允许您在部署到 Amazon MWAA 之前运行本地 Apache Airflow 环境来开发和测试 DAG、自定义插件和依赖关系。
- 要运行 CLI，请参阅 [aws-mwaa-local-runner](#) 上的 GitHub。

步骤二：创建 requirements.txt

下一节介绍如何在 `requirements.txt` 文件中指定 Python Package [索引中的 Python](#) 依赖关系。

Apache Airflow v2

1. 在本地测试。在创建 `requirements.txt` 文件之前，以迭代方式添加其他库以找到软件包及其版本的正确组合。要运行亚马逊 MWAA CLI 实用程序，请参阅 [aws-mwaa-local-runner](#) 上的 GitHub。

2. 查看 Apache Airflow 套餐额外内容。要查看亚马逊 MWAA 上为 Apache Airflow v2 安装的软件包列表，请参阅 GitHub 网站requirements.txt上的[亚马逊 MWAA 本地运行器](#)。
3. 添加约束声明。在文件顶部添加 Apache Airflow v2 环境的限制requirements.txt文件。Apache Airflow 限制文件指定了 Apache Airflow 发布时可用的提供者版本。

从 Apache Airflow v2.5.1 开始，您的需求文件必须包含--constraint声明。如果您不提供限制条件，Amazon MWAA 将为您指定一个限制条件，以确保您的要求中列出的软件包与您的环境兼容。

Note

Apache Airflow v2.4.3 及更高版本使用 Python 版本 3.10，而较早的 Apache Airflow 版本使用 Python

在以下示例中，替换为环境的版本号以及{Airflow-version}{Python-version}与您的环境兼容的 Python 版本。

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

如果限制文件确定该xyz==1.0软件包与您环境中的其他软件包不兼容，则pip3 install将无法阻止将不兼容的库安装到您的环境中。如果任何软件包的安装失败，则可以在日志的相应日志流中查看每个 Apache Airflow 组件（调度程序、工作程序和 Web 服务器）的错误日 CloudWatch 志。有关日志类型的更多信息，请参阅[the section called “查看气流日志” \(p. 257\)](#)。

4. Apache Airflow 添加[软件包的额外内容](#)和版本(==)。这有助于防止在您的环境中安装名称相同但版本不同的软件包。

```
apache-airflow[package-extra]==2.5.1
```

5. Python 库。在requirements.txt文件中添加软件包名称和版本(==)。这有助于防止future 自动应用来自 [PyPi.org](#) 的重大更新。

```
library == version
```

Example Boto3 和 psycopg2-binary

提供此示例是为了演示。boto 和 psycopg2-二进制库包含在 Apache Airflow v2 基础安装中，无需在requirements.txt文件中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的软件包没有版本，Amazon MWAA 会从 [PyPi.org](#) 安装该软件包的最新版本。此版本可能与您的其他软件包冲突requirements.txt。

Apache Airflow v1

1. 在本地测试。在创建requirements.txt文件之前，以迭代方式添加其他库以找到软件包及其版本的正确组合。要运行亚马逊 MWAA CLI 实用程序，请参阅[aws-mwaa-local-runner](#)上的 GitHub。
2. 查看 Airflow 套餐的额外内容。查看 Apache Airflow v1.10.12 的可用软件包列表，[网址为 https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt](#)。
3. 添加约束文件。将 Apache Airflow v1.10.12 的限制文件添加到requirements.txt文件顶部。如果限制文件确定该xyz==1.0软件包与您环境中的其他软件包不兼容，则pip3 install将无法阻止将不兼容的库安装到您的环境中。

```
--constraint "https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt"
```

4. Amaze Airflow vor 1.10.12 软件包 添加 A [irflow 套餐额外内容](#) 和 Apache Airflow v1.10.12 版本 (==)。这有助于防止在您的环境中安装名称相同但版本不同的软件包。

```
apache-airflow[package]==1.10.12
```

Example 安全外壳 (SSH)

以下示例requirements.txt文件为 Apache Airflow v1.10.12 安装 SSH。

```
apache-airflow[ssh]==1.10.12
```

5. Python 库。在requirements.txt文件中添加软件包名称和版本 (==)。这有助于防止future 自动应用来自 [PyPi.org](#) 的重大更新。

```
library == version
```

Example Boto3

以下示例requirements.txt文件为 Apache Airflow v1.10.12 安装了 Boto3 库。

```
boto3 == 1.17.4
```

如果指定的软件包没有版本，Amazon MWAA 会从 [PyPi.org](#) 安装该软件包的最新版本。此版本可能与您的其他软件包冲突requirements.txt。

上传requirements.txt到 Amazon S3

您可以使用 Amazon S3 控制台或AWS Command Line Interface (AWS CLI) 将文件上传到 Amazon S3控制台，或 () 将requirements.txt文件上传Amazon S3 存储桶。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下条件：

- [AWS CLI— 安装版本 2。](#)
- [AWS CLI— 使用aws configure. 进行快速配置。](#)

要使用上传AWS CLI

1. 使用以下命令列出您的所有 Amazon S3 存储桶。

```
aws s3 ls
```

2. 使用以下命令可列出您的环境 Amazon S3 存储桶中的文件和文件夹。

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

3. 以下命令会将requirements.txt文件上传Amazon S3 存储桶。

```
aws s3 cp requirements.txt s3://YOUR_S3_BUCKET_NAME/requirements.txt
```

使用 Amazon S3 控制台

Amazon S3 控制台是基于 Web 的用户界面，允许您创建和管理 Amazon S3 存储桶中的资源。

使用 Amazon S3 控制台上载的内容

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在 S3 窗格中的 DAG 代码中选择 S3 存储段链接，在 Amazon S3 控制台上打开您的存储桶。
4. 请选择 Upload (上传)。
5. 选择“添加文件”。
6. 选择您的本地副本 requirements.txt，选择上传。

在您的环境中安装 Python 依赖项

本节介绍如何通过指定 requirements.txt 文件的路径以及每次更新 requirements.txt 文件的版本来安装您上传到 Amazon S3 存储桶的依赖项。

在亚马逊 MWAA 控制台 requirements.txt 上指定路径 (第一次)

如果这是您第一次创建并将其上传 requirements.txt 到 Amazon S3 存储桶，则还需要在 Amazon MWAA 控制台上指定文件的路径。您只需要完成此步骤一次。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 在 Amazon S3 窗格的 DAG 代码上，选择“需求文件-可选”字段旁边的“浏览 S3”。
5. 在 Amazon S3 存储桶中选择 requirements.txt 文件。
6. 选择 Choose (选择)。
7. 选择下一步，更新环境。

在环境完成更新后，您可以立即开始使用新软件包。

在亚马逊 MWAA 控制台上指定 requirements.txt 版本

每次在 Amazon S3 存储桶中上传 requirements.txt 文件的新版本时，都需要 requirements.txt 在 Amazon MWAA 控制台上指定文件版本。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 在 Amazon S3 窗格中的 DAG 代码上，在下拉列表中选择 requirements.txt 版本。
5. 选择下一步，更新环境。

在环境完成更新后，您可以立即开始使用新软件包。

查看您的日志requirements.txt

您可以查看 Apache Airflow 日志，以便调度程序安排您的工作流程和解析您的dags文件夹。以下步骤描述了如何在 Amazon MWAA 控制台上打开调度程序的日志组，以及如何在日志控制台上查看 Apache Airflow CloudWatch 日志。

查看日志requirements.txt

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“监视”窗格中选择 Airflow 调度程序日志组。
4. 在requirements_install_ip日志流中选择日志。
5. 您应该在上看到安装在环境中的软件包列表/usr/local/airflow/.local/bin。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgm5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 查看软件包列表，以及这些软件包在安装过程中是否遇到任何错误。如果出现问题，您可能会看到类似以下内容的错误：

```
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))
```

接下来做什么？

- 使用 on 在本地测试您的 DAG、自定义插件和 Python 依赖关系 GitHub。 [aws-mwaa-local-runner](#)

删除 Amazon S3 上的文件

本页介绍版本控制在适用于 Apache Airflow 的亚马逊托管工作流 (MWAA) 环境的 Amazon S3 存储桶中的工作原理，以及删除 DAG 的步骤、plugins.zip，或者requirements.txt文件。

目录

- [先决条件 \(p. 123\)](#)
- [版本控制概述 \(p. 123\)](#)
- [工作方式 \(p. 123\)](#)
- [删除 Amazon S3 上的 DAG \(p. 123\)](#)
- [从环境中移除“当前” requirements.txt 或 plugins.zip \(p. 124\)](#)
- [删除“非当前”\(以前的\) requirements.txt 或 plugins.zip 版本 \(p. 124\)](#)
- [使用生命周期删除“非当前”\(以前的\) 版本并自动删除标记 \(p. 124\)](#)
- [删除 requirements.txt “非当前” 版本并自动删除标记的生命周期策略示例 \(p. 124\)](#)
- [接下来做什么？ \(p. 125\)](#)

先决条件

在完成此页面上的步骤之前，您需要具备以下内容。

- Permissions (权限)— 您的AWS账户必须已获得管理员授予的访问权限 [AmazonMWAAFullConsoleAccess \(p. 33\)](#) 您的环境的访问控制策略。此外，您的亚马逊 MWAA 环境必须获得您的 [执行角色 \(p. 44\)](#) 访问AWS您的环境使用的资源。
- 访问— 如果您需要访问公共存储库以直接在 Web 服务器上安装依赖项，则您的环境必须配置为公共网络Web 服务器访问。有关更多信息，请参阅 [the section called “Apache Airflow” \(p. 52\)](#)。
- Amazon S3 配置— 该 [Amazon S3 存储桶 \(p. 16\)](#) 用来存储你的 DAG，自定义插件plugins.zip，以及 Python 依赖关系在requirements.txt必须配置公有访问被阻止和版本控制已启用。

版本控制概述

这些区域有：requirements.txt和plugins.zip您的 Amazon S3 存储桶中的版本。如果为对象启用了 Amazon S3 存储桶版本控制，并且从 Amazon S3 存储桶中删除了工件（例如 plugins.zip），则该文件不会被完全删除。每当在 Amazon S3 上删除项目时，都会创建该文件的新副本，该副本是 404（对象未找到）错误/0k 文件，显示“我不在这里”。Amazon S3 称之为删除标记。与任何其他对象一样，删除标记是文件的“空”版本，具有键名（键）和版本 ID。

我们建议您定期删除文件版本并删除标记，以降低 Amazon S3 存储桶的存储成本。要完全删除“非当前”（以前的）文件版本，必须删除这些文件的版本，然后删除删除标记对于版本。

工作方式

Amazon MWAA 每三十秒对您的 Amazon S3 存储桶运行一次同步操作。这会导致 Amazon S3 存储桶中的任何 DAG 删除都将同步到 Fargate 容器的气流映像。

适用于plugins.zip和requirements.txt文件，只有在 Amazon MWAA 使用自定义插件和 Python 依赖项为 Fargate 容器构建新的 Airflow 映像时，环境更新后才会发生更改。如果删除当前的任一版本的requirements.txt要么plugins.zip文件，然后更新您的环境而不为已删除的文件提供新版本，则更新将失败并显示错误消息，例如“无法读取版本{version}的文件{file}”。

删除 Amazon S3 上的 DAG

一个 DAG 文件 (.py) 未受版本控制，可以在 Amazon S3 控制台上将其删除。以下步骤描述了如何删除 Amazon S3 存储桶上的 DAG。

删除 DAG

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. 选择S3 bucket中的链接S3 中的 DAG 代码窗格，在 Amazon S3 控制台上打开存储桶。
4. 选择 dags 文件夹。
5. 选择 DAG，Delete。
6. UNDER删除对象？，键入delete。
7. 请选择 Delete objects（删除对象）。

Note

Apache 气流会保留历史 DAG 运行。在 Apache Airflow 中运行 DAG 后，无论文件状态如何，它都会保留在 Ariflow DAG 列表中，直到您在 Apache Airflow 中将其删除。要在 Apache Airflow 中删除 DAG，请选择链接column。

从环境中移除“当前” requirements.txt 或 plugins.zip

目前，没有办法在添加 plugins.zip 或 requirements.txt 后将其从环境中删除，但我们正在努力解决这个问题。在此期间，一种解决方法是分别指向空文本或 zip 文件。

删除“非当前”（以前的） requirements.txt 或 plugins.zip 版本

这些区域有：requirements.txt和plugins.zip您的 Amazon S3 存储桶中的文件版本控制在 Amazon MWAA 上。如果您想完全删除 Amazon S3 存储桶上的这些文件，则必须检索数据元的当前版本 (121212)（例如 plugins.zip），删除版本，然后删除删除标记针对文件版本。

您也可以在 Amazon S3 控制台上删除“非当前”（以前的）文件版本；但是，您仍然需要删除删除标记使用以下选项之一。

- 要检索对象版本，请参阅[从启用了版本控制的存储桶中检索对象版本](#)在 Amazon S3 指南中).
- 要删除对象版本，请参阅[从启用了版本控制的存储桶中删除对象版本](#)在 Amazon S3 指南中).
- 要删除标记，请参阅[删除“删除标记”](#)在 Amazon S3 指南中).

使用生命周期删除“非当前”（以前的）版本并自动删除标记

您可以为 Amazon S3 存储桶配置生命周期策略，以便在一定天数后删除 Amazon S3 存储桶中的 plugins.zip 和 requirements.txt 文件的“非当前”（先前）版本，或者删除过期对象的删除标记。

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. UNDERAmazon S3 中的 DAG 代码，请选择您的 Amazon S3 存储桶。
4. 选择创建生命周期规则。

删除 requirements.txt “非当前”版本并自动删除标记的生命周期策略示例

以下示例演示如何创建生命周期规则，该规则将在三十天后永久删除 requirements.txt 文件的“非当前”版本及其删除标记。

1. 打开[环境页面](#)在亚马逊 MWAA 控制台上。
2. 选择环境。
3. UNDERAmazon S3 中的 DAG 代码，请选择您的 Amazon S3 存储桶。
4. 选择创建生命周期规则。
5. In生命周期规则名称，键入Delete previous requirements.txt versions and delete markers after thirty days.
6. In前缀、要求。
7. In生命周期规则操作，选择永久删除对象的先前版本和删除过期的删除标记或未完成的分段上传。
8. In对象成为早期版本后的天数，键入30.
9. In过期的对象删除标记，选择删除过期对象删除标记，对象在 30 天后被永久删除。

接下来做什么？

- 了解有关 Amazon S3 删除标记的更多信息 [管理删除标记](#).
- 了解有关 Amazon S3 生命周期的更多信息 [即将过期的对象](#).

联网

本指南描述了亚马逊 MWAA 环境需要的亚马逊 VPC 网络设置。

小节目录

- [关于 Amazon MWAA 上的联网 \(p. 126\)](#)
- [您在亚马逊 MWAA 上的 VPC 中的安全性 \(p. 129\)](#)
- [在 Amazon MWAA 上管理 VPC 终端节点的访问 \(p. 132\)](#)
- [使用私有路由在 Amazon VPC 中创建所需的 VPC 服务终端节点 \(p. 136\)](#)

关于 Amazon MWAA 上的联网

Amazon VPC 是链接到您的虚拟网络。AWSaccount. 它通过提供对虚拟基础设施和网络流量分段的精细控制，为您提供云安全性和动态扩展的能力。本页面介绍了 Amazon VPC 基础设施公共路由要么私人路由支持 Amazon Managed Workflow Apache Airflow (MWAA) 环境所需的。

目录

- [条款 \(p. 126\)](#)
- [什么受支持 \(p. 126\)](#)
- [VPC 基础设施概述 \(p. 127\)](#)
 - [通过互联网进行公共路由 \(p. 127\)](#)
 - [没有互联网接入的私人路 \(p. 127\)](#)
- [亚马逊 VPC 和 Apache Airflow 访问模式的示例用例 \(p. 128\)](#)
 - [允许访问互联网-新的 Amazon VPC 网络 \(p. 128\)](#)
 - [不允许访问互联网-新的 Amazon VPC 网络 \(p. 129\)](#)
 - [不允许访问互联网-现有的 Amazon VPC 网络 \(p. 129\)](#)

条款

公共路由

可访问 Internet 的 Amazon VPC 网络。

私人路由

Amazon VPC 网络没有访问 Internet。

什么受支持

下表描述了 Amazon MWAA 支持的 Amazon VPC 类型。

Amazon VPC 类型	支持	
尝试创建环境的账户所有的 Amazon VPC。	是	
共享的亚马逊 VPC，其中多个 AWS 创建账户 AWS 资源的费用。	否	

VPC 基础设施概述

当您创建 Amazon MWAA 环境时，Amazon MWAA 会根据您为环境选择的 Apache Airflow 访问模式为您的环境创建一到两个 VPC 终端节点。这些终端节点显示为您的 Amazon VPC 中使用私有 IP 的弹性网络接口 (ENI)。创建这些端点后，发往这些 IP 的所有流量都将私下或公开路由到相应的 AWS 您的环境使用的服务。

以下部分介绍公开路由流量所需的 Amazon VPC 基础设施通过互联网或私下在您的亚马逊 VPC 内。

通过互联网进行公共路由

本节介绍了具有公共路由的环境的 Amazon VPC 基础设施。您将需要以下 VPC 基础设施：

- 一个 VPC 安全组。VPC 安全组充当虚拟防火墙，用于控制实例上的入站（入站）和出站（出站）网络流量。
 - 最多可以指定 5 个安全组。
 - 安全组必须指定自引用入站规则。
 - 安全组必须为所有流量指定出站规则（0.0.0.0/0）。
 - 安全组必须允许自引用规则中的所有流量。例如 [（推荐）所有访问自引用安全组示例 \(p. 131\)](#)。
 - 安全组可以可选通过指定 HTTPS 端口范围的端口范围来进一步限制流量 443 和 TCP 端口范围 5432。例如，[（可选）限制对端口 5432 的入站访问的示例安全组 \(p. 131\)](#) 和 [（可选）限制对端口 443 的入站访问的示例安全组 \(p. 131\)](#)。
- 两个公有子网。公有子网是指与包含指向 Internet 网关的路由的路由表关联的子网。
 - 需要两个公有子网。这允许 Amazon MWAA 在其他可用区域中为您的环境构建新容器映像（如果一个容器出现故障）。
 - 这些子网必须位于不同的可用区中。例如，us-east-1a 和 us-east-1b。
 - 子网必须路由到具有弹性 IP 地址 (EIP) 的 NAT 网关（或 NAT 实例）。
 - 子网必须有一个路由表，该路由表将 Internet 范围的流量定向到 Internet 网关。
- 两个私有子网。私有子网是一个子网不与包含指向 Internet 网关的路由的路由表关联。
 - 需要两个私有子网。这允许 Amazon MWAA 在其他可用区域中为您的环境构建新容器映像（如果一个容器出现故障）。
 - 这些子网必须位于不同的可用区中。例如，us-east-1a 和 us-east-1b。
 - 这些子网必须有一个到 NAT 设备（网关或实例）的路由表。
 - 这些子网不能路由到 Internet 网关。
- 网络访问控制列表 (ACL)。NACL 在子网级别管理（通过允许或拒绝规则）入站和出站流量。
 - NACL 必须有入站规则，允许所有流量（0.0.0.0/0）。
 - NACL 必须有拒绝所有流量的出站规则（0.0.0.0/0）。
 - 例如 [（推荐）ACL 示例 \(p. 130\)](#)。
- 两个 NAT 网关（或 NAT 实例）。NAT 设备将来自私有子网中实例的流量转发到 Internet 或其他 AWS 服务，然后将响应路由回给实例。
 - NAT 设备必须连接到公有子网。（每个公有子网一个 NAT 设备。）
 - NAT 设备必须将弹性 IPv4 地址 (EIP) 附加到每个公有子网。
- Internet 网关。互联网网关将 Amazon VPC 连接到 Internet 和其他 AWS 服务。
 - 必须将互联网网关连接到亚马逊 VPC。

没有互联网接入的私人路

本节介绍 VPC 了使用私人路由。您将需要以下 VPC 基础设施：

- 一个 VPC 安全组。VPC 安全组充当虚拟防火墙，用于控制实例上的入站（入站）和出站（出站）网络流量。

- 最多可以指定 5 个安全组。
- 安全组必须指定自引用入站规则。
- 安全组必须为所有流量指定出站规则 (0.0.0.0/0)。
- 安全组必须允许自引用规则中的所有流量。例如 [\(推荐 \) 所有访问自引用安全组示例 \(p. 131\)](#)。
- 安全组可以可选通过指定 HTTPS 端口范围的端口范围来进一步限制流量 443 和 TCP 端口范围 5432。例如，[\(可选 \) 限制对端口 5432 的入站访问的示例安全组 \(p. 131\)](#) 和 [\(可选 \) 限制对端口 443 的入站访问的示例安全组 \(p. 131\)](#)。
- 两个私有子网。私有子网是一个子网不与包含指向 Internet 网关的路由的路由的路由表关联。
 - 需要两个私有子网。这允许 Amazon MWAA 在其他可用区域中为您的环境构建新容器映像 (如果一个容器出现故障)。
 - 这些子网必须位于不同的可用区中。例如，us-east-1a 和 us-east-1b。
 - 子网必须有一个到达 VPC 终端节点的路由表。
 - 这些子网不能有一个到 NAT 设备 (网关或实例) 的路由表，也不 Internet 网关。
- 网络访问控制列表 (ACL)。NACL 在子网级别管理 (通过允许或拒绝规则) 入站和出站流量。
 - NACL 必须有入站规则，允许所有流量 (0.0.0.0/0)。
 - NACL 必须有拒绝所有流量的出站规则 (0.0.0.0/0)。
 - 例如 [\(推荐 \) ACL 示例 \(p. 130\)](#)。
- 本地路由表。本地路由表是 VPC 内通信的默认路由。
 - 本地路由表必须与您的私有子网关联。
 - 本地路由表必须支持 VPC 中的实例与您自己的网络进行通信。例如，如果您正在使用 AWS Client VPN 访问 Apache Airflow 的 VPC 接口终端节点 Web 服务器，路由表必须路由到 VPC 终端节点。
- VPC 终端节点为每个 AWS 您的环境使用的服务，以及 Apache Airflow VPC 终端节点 AWS 区域和 Amazon VPC 作为您的 Amazon MWAA 环境。
 - 适用于每个 VPC 终端节点 AWS 环境和 VPC 终端节点用于 Apache Airflow 的服务。例如 [\(必需 \) VPC 终端节点 \(p. 138\)](#)。
 - VPC 终端节点必须启用私有 DNS。
 - VPC 终端节点必须与环境的两个私有子网关联。
 - VPC 终端节点必须与环境的安全组关联。
 - 应将每个终端节点的 VPC 终端节点策略配置为允许访问 AWS 环境使用的服务。例如 [\(推荐 \) 允许所有访问的 VPC 终端节点策略示例 \(p. 132\)](#)。
 - Amazon S3 的 VPC 终端节点策略应配置为允许存储桶访问。例如 [\(推荐 \) 允许存储桶访问的 Amazon S3 网关终端节点策略示例 \(p. 132\)](#)。

亚马逊 VPC 和 Apache Airflow 访问模式的示例用例

本节介绍了亚马逊 VPC 和 Apache Airflow 中网络访问的不同使用案例。Web 服务器您应该在亚马逊 MWAA 控制台上选择访问模式。

允许访问互联网-新的 Amazon VPC 网络

如果您的组织允许访问 VPC 中的互联网，和你希望用户访问你的 Apache Airflow Web 服务器通过互联网：

1. 创建 Amazon VPC 网络具有 Internet 访问。
2. 使用创建环境公共网 Apache Airflow 的访问模式 Web 服务器。
3. 我们建议什么：我们建议使用 AWS CloudFormation 快速入门模板，可同时创建 Amazon VPC 基础设施、Amazon S3 存储桶和 Amazon MWAA 环境。要了解更多信息，请参阅 [Workflows for Apache Airflow \(MWAA\) \(p. 5\)](#)。

如果您的组织允许访问 VPC 中的互联网，和你想限制 Apache Airflow Web 服务器访问 VPC 内的用户：

- [安全性概述 \(p. 130\)](#)
- [网络访问控制列表 \(ACL\) \(p. 130\)](#)
 - [\(推荐 \) ACL 示例 \(p. 130\)](#)
- [VPC 安全组 \(p. 131\)](#)
 - [\(推荐 \) 所有访问自引用安全组示例 \(p. 131\)](#)
 - [\(可选 \) 限制对端口 5432 的入站访问的示例安全组 \(p. 131\)](#)
 - [\(可选 \) 限制对端口 443 的入站访问的示例安全组 \(p. 131\)](#)
- [VPC 终端节点策略 \(仅限私有路由 \) \(p. 132\)](#)
 - [\(推荐 \) 允许所有访问的 VPC 终端节点策略示例 \(p. 132\)](#)
 - [\(推荐 \) 允许存储桶访问的 Amazon S3 网关终端节点策略示例 \(p. 132\)](#)

条款

公共路由

可以访问互联网的Amazon VPC 网络。

私有路由

亚马逊 VPC 网络没有访问互联网。

安全性概述

安全组和访问控制列表 (ACL) 提供了使用您指定的规则控制通过 Amazon VPC 中的子网和实例的网络流量的方法。

- 进出子网的网络流量可由访问控制列表 (ACL) 控制。您只需要一个 ACL，同一个 ACL 可以在多个环境中使用。
- 进出实例的网络流量可以由 Amazon VPC 安全组控制。每个环境可以使用一到五个安全组。
- 进出实例的网络流量也可以由 VPC 终端节点策略控制。如果您的组织不允许在 Amazon VPC 内访问互联网，并且您正在使用 Amazon VPC 网络私有路由，则需要 VPC 终端节点策略[AWSVPC 终端节点和 Apache AirVPC low \(p. 138\)](#)。

网络访问控制列表 (ACL)

一个[网络访问控制列表 \(ACL\)](#)可以在 (通过允许或拒绝规则) 管理入站和出站流量子网级别。ACL 是无状态的，这意味着必须单独明确地指定入站和出站规则。它用于指定 VPC 网络中允许进出实例的网络流量类型。

每个 Amazon VPC 都有允许所有入站和出站流量的默认 ACL。您可以编辑默认 ACL 规则，也可以创建自定义 ACL 并将其附加到子网。一个子网在任何时候只能连接一个 ACL，但一个 ACL 可以连接到多个子网。

(推荐) ACL 示例

以下示例显示该语法入站和往外地的ACL 规则可以用于 Amazon VPC 的 Amazon VPC公共路由要么私有路由。

规则编号	类型	协议	端口范围	源	允许/拒绝
100	所有 IPv4 流量	全部	All	0.0.0.0/0	Allow (允许)
*	所有 IPv4 流量	全部	All	0.0.0.0/0	拒绝

VPC 安全组

一个 [VPC 安全组](#) 充当虚拟防火墙，控制网络流量实例级别。安全组是有状态的，这意味着当允许入站连接时，允许它回复。它用于指定允许从 VPC 网络中的实例进入的网络流量类型。

每个 Amazon VPC 都有一个默认的安全组。默认情况下，它没有入站规则。它具有允许所有出站流量的出站规则。您可以编辑默认安全组规则，也可以创建自定义安全组并将其附加到您的 Amazon VPC。在 Amazon MWAA 上，您需要配置入站和出站规则，以便在 NAT 网关上引导流量。

(推荐) 所有访问自引用安全组示例

以下示例显示该语法入站安全组规则，允许使用 Amazon VPC 的 Amazon VPC 的所有流量公共路由要么私有路由。本示例中的安全组是一个自引用规则。

类型	协议	源类型	源		
所有流量	All	全部	sg-0909e8e81919/ my-mwaa-vpc- security-组中		

以下示例显示该语法往外地的安全组规则。

类型	协议	源类型	源		
所有流量	All	All	0.0.0.0/0		

(可选) 限制对端口 5432 的入站访问的示例安全组

以下示例显示该语法入站安全组规则，允许您的环境的 Amazon Aurora PostgreSQL 元数据数据库 (由 Amazon MWAA 拥有) 的端口 5432 上的所有 HTTPS 流量。

Note

如果您选择使用此规则限制流量，则需要添加另一条规则以允许端口 443 上的 TCP 流量。

类型	协议	端口范围	源类型	源	
自定义 TCP	TCP	5432	自定义	sg-0909e8e81919/ my-mwaa-vpc- security-组中	

(可选) 限制对端口 443 的入站访问的示例安全组

以下示例显示该语法入站允许端口 443 上的所有 TCP 流量用于 Apache 气流的安全组规则Web 服务器。

类型	协议	端口范围	源类型	源	
HTTPS	TCP	443	自定义	sg-0909e8e81919/ my-mwaa-vpc- security-组中	

VPC 终端节点策略 (仅限私有路由)

一个[VPC 终端节点AWS PrivateLink](#) 策略控制对AWS私有子网中的服务。VPC 终端节点策略是一种 IAM 资源策略，您可将它们附加到您的 VPC 网关或接口终端节点。本部分介绍每个 VPC 终端节点的 VPC 终端节点策略所需的权限。

我们建议对您创建的每个 VPC 终端节点使用 VPC 接口终端节点策略，该策略允许对所有 VPC 终端节点进行完全访问AWS服务，并且将您的执行角色专门用于AWS权限。

(推荐) 允许所有访问的 VPC 终端节点策略示例

以下示例显示了 Amazon VPC 的 VPC 接口终端节点策略私有路由。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

(推荐) 允许存储桶访问的 Amazon S3 网关终端节点策略示例

以下示例展示一个 VPC 网关终端节点策略提供对 Amazon ECR 操作所需的 Amazon ECR 操作所需的 Amazon ECR 操作所需的 Amazon S3 存储桶的访问权限私有路由。除了存储 DAG 和支持文件的存储桶外，这也是检索您的 Amazon ECR 映像所必需的。

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

在 Amazon MWAA 上管理 VPC 终端节点的访问

VPC 终端节点 (AWS PrivateLink) 使您能够将 VPC 私密地连接到托管在上的服务AWS无需互联网网关、NAT 设备、VPN 或防火墙代理。这些终端节点都是水平可扩展且高度可用的虚拟设备，可以在 VPC 中的实例与AWS服务。本页面介绍了 Amazon MWAA 创建的 VPC 终端节点，以及如何访问 Apache Airflow 的 VPC 终端节点Web 服务器如果你选择了私有网络Amazon Managed Workflows in Apache Airflow (MWAA) 上的访问模式。

目录

- [定价 \(p. 133\)](#)
- [VPC 终端节点概述 \(p. 133\)](#)
 - [公共网络访问模式 \(p. 133\)](#)

- [专用网络访问模式 \(p. 133\)](#)
- [允许使用其他AWS服务 \(p. 133\)](#)
- [查看 VPC 终端节点 \(p. 134\)](#)
 - [在 Amazon VPC 控制台中查看 VPC 终端节点 \(p. 134\)](#)
 - [识别 Apache Airflow Web 服务器及其 VPC 终端节点的私有 IP 地址 \(p. 134\)](#)
- [访问 Apache Airflow Web 服务器的 VPC 终端节点 \(专用网络访问 \) \(p. 135\)](#)
 - [使用 AWS Client VPN \(p. 135\)](#)
 - [使用 Linux 堡垒主机 \(p. 135\)](#)
 - [使用 Load Balancer \(高级 \) \(p. 135\)](#)

定价

- [AWS PrivateLink 定价](#)

VPC 终端节点概述

当您创建 Amazon MWAA 环境时，Amazon MWAA 会为您的环境创建一到两个 VPC 终端节点。这些终端节点将显示为您的 Amazon VPC 中的私有 IP 地址的弹性网络接口 (ENI)。创建这些端点后，发往这些 IP 的所有流量都将私下或公开路由到相应的AWS您的环境使用的服务。

公共网络访问模式

如果你选择了公网Apache Airflow 的访问模式Web 服务器，网络流量已公开路由通过互联网。

- Amazon MWAA 为您的 Amazon Aurora PostgreSQL 元数据数据库创建 VPC 接口终端节点。在映射到私有子网的可用区中创建此终端节点，独立于其他子网。AWS账户。
- 然后，Amazon MWAA 将私有子网的 IP 地址绑定到接口终端节点。这旨在支持从 Amazon VPC 的每个可用区绑定单个 IP 的最佳实践。

专用网络访问模式

如果你选择了私有网络Apache Airflow 的访问模式Web 服务器，网络流量是私下路由的在您的亚马逊 VPC 内。

- 亚马逊 MWAA 为您的 Apache Airflow 创建 VPC 接口终端节点Web 服务器，以及您的 Amazon Aurora PostgreSQL 元数据数据库的接口终端节点。这些终端节点是在映射到私有子网的可用区中创建的，独立于其他AWS账户。
- 然后，Amazon MWAA 将私有子网的 IP 地址绑定到接口终端节点。这旨在支持从 Amazon VPC 的每个可用区绑定单个 IP 的最佳实践。

允许使用其他AWS服务

接口终端节点在中为您的环境使用执行角色AWS Identity and Access Management(IAM) 来管理AWS您的环境使用的资源。更多AWS已为环境启用服务，每项服务都需要您使用环境的执行角色配置权限。要添加权限，请参阅[Amazon MWAA 执行角色 \(p. 44\)](#)。

如果你选择了私有网络Apache Airflow 的访问模式Web 服务器，您还必须在 VPC 终端节点策略中允许每个终端节点的权限。要了解更多信息，请参阅 [the section called “VPC 终端节点策略 \(仅限私有路由 \)” \(p. 132\)](#)。

查看 VPC 终端节点

本节介绍如何查看 Amazon MWAA 创建的 VPC 终端节点，以及如何识别 Apache Airflow VPC 终端节点的私有 IP 地址。

在 Amazon VPC 控制台中查看 VPC 终端节点

以下部分显示了查看 Amazon MWAA 创建的 VPC 终端节点的步骤，以及您在使用时可能创建的任何 VPC 终端节点的步骤。私人路由对于您的 Amazon VPC。

查看 VPC 终端节点

1. 打开[终端节点页](#)在 Amazon VPC 控制台上。
2. 使用AWS区域选择器用于选择您的地区。
3. 您应该看到由 Amazon MWAA 创建的 VPC 接口终端节点，以及您在使用时可能创建的任何 VPC 终端节点私人路由在您的 Amazon VPC 中。

要了解有关 Amazon VPC 所需的 VPC 服务终端节点的更多信息私人路由，请参阅[使用私有路由在 Amazon VPC 中创建所需的 VPC 服务终端节点 \(p. 136\)](#)。

识别 Apache Airflow Web 服务器及其 VPC 终端节点的私有 IP 地址

以下步骤介绍了如何检索 Apache Airflow Web 服务器及其 VPC 接口终端节点的主机名以及它们的私有 IP 地址。

1. 请使用以下AWS Command Line Interface(AWS CLI) 命令来检索 Apache Airflow 的主机名Web 服务器。

```
aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query  
'Environment.WebserverUrl'
```

您应该可以看到类似于如下响应的内容：

```
"99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
```

2. 运行挖在上一个命令的响应中返回的主机名上的命令。例如：

```
dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-  
west-2.airflow.amazonaws.com
```

您应该可以看到类似于如下响应的内容：

```
vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

3. 请使用以下AWS Command Line Interface(AWS CLI) 命令来检索之前命令响应中返回的 VPC 终端节点 DNS 名称。例如：

```
aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-  
svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

您应该可以看到类似于如下响应的内容：

```
"DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com",
```

4. 运行nslookup要么挖命令在您的 Apache Airflow 主机名及其 VPC 终端节点 DNS 名称上检索 IP 地址。例如：

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

您应该可以看到类似于如下响应的内容：

```
10.199.11.111  
10.999.11.33
```

访问 Apache Airflow Web 服务器的 VPC 终端节点 (专用网络访问)

如果你选择了私有网络Apache Airflow 的访问模式Web 服务器，您需要创建一个机制来访问 Apache Airflow 的 VPC 接口终端节点Web 服务器. 对于这些资源，您必须使用与 Amazon MWAA 环境相同的 Amazon VPC、VPC 安全组和私有子网。

使用 AWS Client VPN

AWS Client VPN 是一种基于客户端的托管 VPN 服务，让您能够安全地 AWS 资源和本地网络中的资源。它使用 OpenVPN 客户端从任何位置提供安全的 TLS 连接。

我们建议按照亚马逊 MWAA 教程来配置 Client VPN：[教程：使用配置专用网络访问权限AWS Client VPN \(p. 155\)](#)。

使用 Linux 堡垒主机

堡垒主机是一种服务器，其目的是提供从外部网络（例如从计算机通过 Internet）访问专用网络的服务。Linux 实例位于公有子网中，它们设置的安全组允许从附加到运行堡垒主机的底层 Amazon EC2 实例的安全组进行 SSH 访问。

我们建议按照亚马逊 MWAA 教程来配置 Linux 堡垒主机：[教程：使用 Linux 堡垒主机配置专用网络访问权限 \(p. 162\)](#)。

使用 Load Balancer (高级)

以下部分显示了您需要应用到[应用程序负载均衡器](#)或者[网络负载均衡器](#)。

1. 目标组. 您需要使用指向 Apache Airflow 的私有 IP 地址的目标组Web 服务器，及其 VPC 接口终端节点。我们建议将两个私有 IP 地址指定为注册目标，因为只使用一个可以降低可用性。要识别私有 IP 地址，请参阅[the section called “识别 Apache Airflow Web 服务器及其 VPC 终端节点的私有 IP 地址” \(p. 134\)](#)在此页上。
2. 状态代码. 我们建议使用200和302目标群组设置中的状态代码。否则，如果 Apache Airflow 的 VPC 终端节点，则可能会将目标标记为运行状况不佳Web 服务器响应302 Redirect错误消息。
3. HTTPS 侦听器. 你需要为 Apache Airflow 指定目标端口Web 服务器. 例如：

协议	端口
TCP	443

4. ACM 新域. 如果要在中关联 SSL/TLS 证书AWS Certificate Manager，您需要为负载均衡器的 HTTPS 侦听器创建新域。
5. ACM 证书区域. 如果要在中关联 SSL/TLS 证书AWS Certificate Manager，您需要将其上传到同一个 AWS区域作为你的环境。例如：
 - Example 要上传证书的区域

```
aws acm import-certificate --certificate fileb://Certificate.pem --certificate-chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --region us-west-2
```

使用私有路由在 Amazon VPC 中创建所需的 VPC 服务终端节点

无法访问互联网的现有 Amazon VPC 网络需要额外的 VPC 服务终端节点 (AWS PrivateLink) 才能使用 Amazon Managed Workflows for Apache Airflows for Apache Air 本页介绍了 Amazon MWAA 使用的AWS 服务所需的 VPC 终端节点、Apache Airflow 所需的 VPC 终端节点，以及如何通过私有路由创建 VPC 终端节点并将其连接到现有 Amazon VPC。

目录

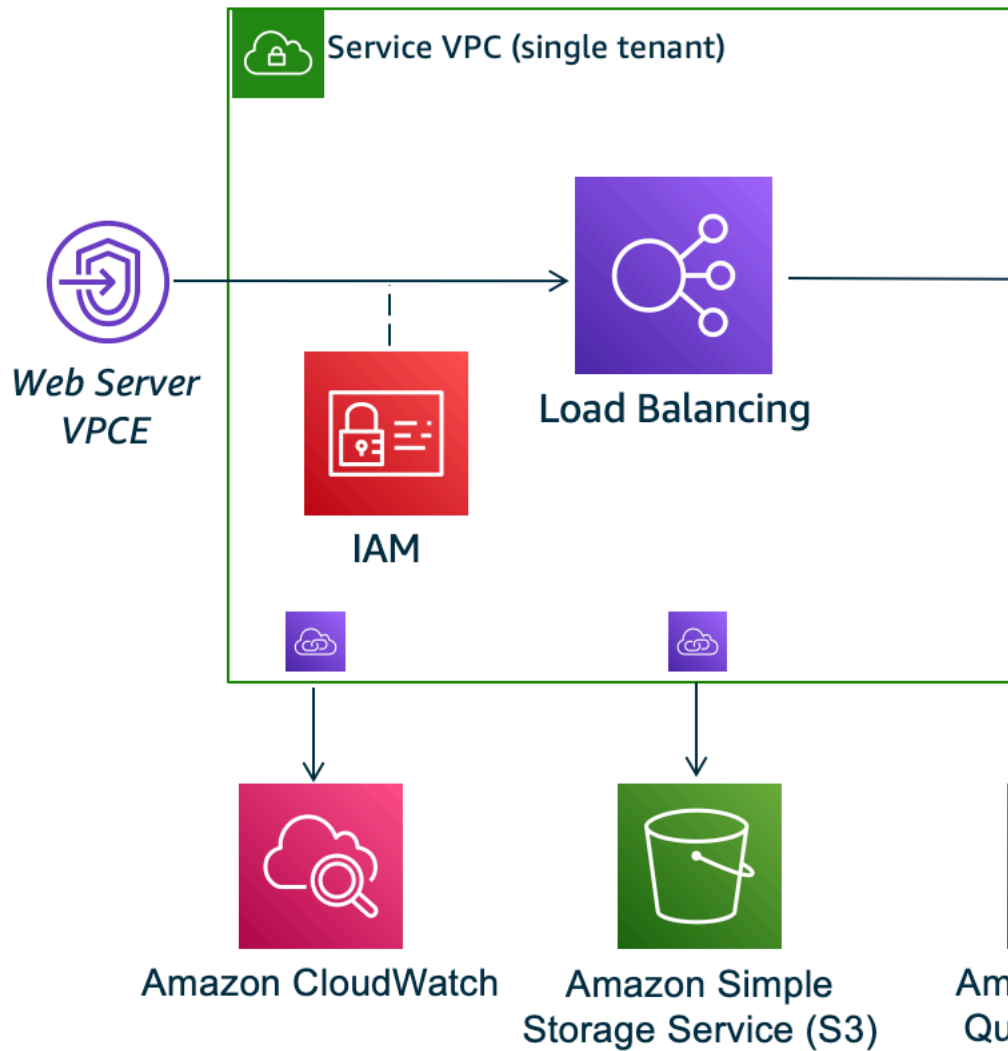
- [定价 \(p. 136\)](#)
- [专用网络和私有路由 \(p. 137\)](#)
- [\(必需 \) VPC 终端节点 \(p. 138\)](#)
- [连接所需的 VPC 终端节点 \(p. 138\)](#)
 - [AWS服务所需的 VPC 终端节点 \(p. 138\)](#)
 - [Apache Airflows flows \(p. 140\)](#)
- [\(可选 \) 为您的 Amazon S3 VPC 接口终端节点启用私有 IP 地址 \(p. 141\)](#)
 - [使用 Route 53 \(p. 141\)](#)
 - [具有自定义 DNS 的 VPC \(p. 141\)](#)

定价

- [AWS PrivateLink 定价](#)

专用网络和私有路由

Private Web Server Opt



私有网络访问模式将对 Apache Airflow UI 的访问权限限制在您的 Amazon VPC 中，这些用户已被授予[对您环境的 IAM 策略的 \(p. 32\)](#)访问权限。

创建具有私有 Web 服务器访问权限的环境时，必须将所有依赖项打包到 Python 轮子存档 (.whl) 中，然后在 .whl 中引用 requirements.txt。有关使用 wheel 打包和安装依赖项的说明，请参阅[使用 Python 轮子管理依赖关系 \(p. 250\)](#)。

下图显示了在 Amazon MWAA 控制台上可以找到私有网络选项的位置。

Web server access

☒ Private network (Recommended)

Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.

☐ Public network (No additional setup)

Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

- 私有路由。[无法访问互联网的 Amazon VPC \(p. 126\)](#) 会限制 VPC 内的网络流量。此页面假设您的 Amazon VPC 无法访问互联网，并且要求您的环境使用的每项 AWS 服务使用 VPC 终端节点，Apache Airflow 的 VPC 终端节点位于与您的亚马逊 MWAA 环境相同的 AWS 区域和亚马逊 VPC。

(必需) VPC 终端节点

以下部分显示了无法访问互联网的 Amazon VPC 所需的 VPC 终端节点。它列出了亚马逊 MWAA 使用的每项 AWS 服务的 VPC 终端节点，包括 Apache Airflow 所需的 VPC 终端节点。

```
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
```

连接所需的 VPC 终端节点

本节介绍使用私有路由为 Amazon VPC 连接所需的 VPC 终端节点的步骤。

AWS 服务所需的 VPC 终端节点

以下部分显示了将环境所用 AWS 服务的 VPC 终端节点连接到现有 Amazon VPC 的步骤。

将 VPC 终端节点连接到您的私有子网

1. [在 Amazon VPC 控制台上打开 Endpoints \(](#)
2. 使用 AWS 地区选择器选择您的区域。
3. 为 Amazon S3 创建终端节点：
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入 `.s3`，然后按键盘上的 Enter。
 - c. 我们建议选择为网关类型列出的服务端点。

例如，`com.amazonaws.us-west-2.s3 amazon Gateway`
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区的两个私有子网，并通过选择启用 DNS 名称来启用该私有 DNS。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
4. 为 Amazon ECR 创建第一个终端节点：

- a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.ecr.dkr**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
5. 为 Amazon ECR 创建第二个终端节点：
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.ecr.api**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
6. 为 CloudWatch 日志创建终端节点：
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.logs**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
7. 创建用于 CloudWatch 监控的端点：
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.monitoring**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
8. 为 Amazon SQS 创建终端节点：
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.sqs**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组³⁹
 - g. 在“策略”中选择“完全访问权限”。

- h. 选择 Create endpoint (创建端点)。
9. 为以下对象创建终端节点AWS KMS :
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.kms**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。

Apache Airflows flows

以下部分显示了将 Apache Airflow 的 VPC 终端节点连接到现有 Amazon VPC 的步骤。

将 VPC 终端节点连接到您的私有子网

1. [在 Amazon VPC 控制台上打开 Endpoints \(](#)
2. 使用AWS地区选择器选择您的区域。
3. 为 Apache Airflow API 创建端点 :
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.airflow.api**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
4. 为 Apache Airflow 环境创建第一个端点 :
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.airflow.env**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。
 - h. 选择 Create endpoint (创建端点)。
5. 为 Apache Airflow 操作创建第二个端点 :
 - a. 选择 Create Endpoint (创建端点)。
 - b. 在“按属性筛选”或“按关键字搜索”文本字段中，键入：**.airflow.ops**，然后按键盘上的 Enter。
 - c. 选择服务端点。
 - d. 在 VPC 中选择您环境的亚马逊 VPC。
 - e. 确保选择了位于不同可用区域的两个私有子网，并启用 Enable DNS 名称。
 - f. 选择您的环境的 Amazon VPC 安全组。
 - g. 在“策略”中选择“完全访问权限”。

- h. 选择 Create endpoint (创建端点) 。

(可选) 为您的 Amazon S3 VPC 接口终端节点启用私有 IP 地址

Amazon S3 接口终端节点不支持私有 DNS。S3 终端节点请求仍解析为公有 IP 地址。要将 S3 地址解析为私有 IP 地址，您需要[在 Route 53 中为 S3 区域终端节点添加一个私有托管区域](#)。

使用 Route 53

本节介绍使用 Route 53 为 S3 接口终端节点启用私有 IP 地址的步骤。

1. 为 Amazon S3 VPC 接口终端节点 (例如 s3.eu-west-1.amazonaws.com) 创建私有托管区域，并将其与您的 Amazon VPC 关联 VPC。
2. 为 Amazon S3 VPC 接口终端节点 (例如 s3.eu-west-1.amazonaws.com) 创建解析为您的 VPC 接口终端节点 A 记录 ALAmazon S3 VPC 接口终端节点。
3. 为您的 Amazon S3 接口终端节点 (例如 *.s3.eu-west-1.amazonaws.com) 创建解析为 VPC 接口终端节点 DNS 名称的别名 A 通配符记录。

具有自定义 DNS 的 VPC

如果您的 Amazon VPC 使用自定义 DNS 路由，则需要通过创建 CNAME 记录在 DNS 解析器 (不是 Route 53，通常是运行 DNS 服务器的 EC2 实例) 中进行更改。例如：

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value: *.vpce-0f67d23e37648915c-e2q2e2j3.s3.eu-west-1.vpce.amazonaws.com
```

Amazon Managed Workflows of Apache Airflow (MWAA)

本指南包括使用 and 配置 Amazon Managed Workflows of Apache Airflow (MWAA) 环境的分步教程。

主题

- [教程：aws-mwaa-local-runner 在持续交付 \(CD\) 管道中配置 \(p. 142\)](#)
- [教程：使用配置专用网络访问权限AWS Client VPN \(p. 155\)](#)
- [教程：使用 Linux 堡垒主机配置专用网络访问权限 \(p. 162\)](#)
- [教程：限制亚马逊 MWAA 用户对 DAG 子集的访问权限 \(p. 169\)](#)

教程：aws-mwaa-local-runner 在持续交付 (CD) 管道中配置

本教程指导您完成 GitHub 使用适用于 Apache Airflow 的 Amazon 托管工作流程 (MWAA) 构建持续交付 (CD) 管道的过程，在本地测试您的 [aws-mwaa-local-runner](#) Apache Airflow 代码。它使用AWS CloudFormation模板来创建您需要的AWS资源。您将了解如何：

- (可选) 创建 Amazon VPC 以使用本教程的AWS资源。
- 通过分叉存储库[aws-mwaa-local-runner](#)在您的 GitHub 账户中创建副本。
- 创建、标记的 Docker 镜像并将其推送[aws-mwaa-local-runner](#)到私有 Amazon ECR 存储库。
- 创建 Amazon Elastic Container Service (Amazon ECS) 集群和AWS Fargate服务[aws-mwaa-local-runner](#)。

目录

- [关于工作 GitHub 流程 \(p. 143\)](#)
- [先决条件 \(p. 143\)](#)
 - [Docker \(p. 143\)](#)
 - [AWS CLI \(p. 143\)](#)
- [第一步：\(可选 \) 创建 Amazon VPC \(p. 143\)](#)
- [第二步：构建 Docker 镜像 \(p. 146\)](#)
- [第三步：在亚马逊 ECR 中创建 Docker 镜像注册表 \(p. 147\)](#)
- [第四步：对 Docker 客户端进行身份验证 \(p. 148\)](#)
- [第五步：推送 Docker 镜像 \(p. 148\)](#)
- [第六步：创建 Amazon ECS 集群 \(p. 150\)](#)
- [第七步：将任务定义推送到 GitHub \(p. 151\)](#)
- [第八步：将您AWS和亚马逊 ECR 信息添加到 GitHub \(p. 152\)](#)
- [第九步：设置入 GitHub 门工作流程 \(p. 152\)](#)
- [第十步：测试工作 GitHub 流程 \(p. 154\)](#)
- [清除 \(p. 154\)](#)
- [接下来做什么？ \(p. 155\)](#)

关于工作 GitHub 流程

实际上，我们希望只有在测试成功的情况下，我们的 GitHub 工作流程才能为您的本地运行器构建新的 Docker 容器映像并将其部署到 Amazon ECR 和 Amazon ECS。GitHub 本教程中的 GitHub 工作流程使用以下 GitHub 操作。

- 用于[检出本地运行器存储库main分支的操作](#)/ 签出工作流程。
- 使用AWS凭证密钥[configure-aws-credentials](#)的工作流程。
- 登录容器镜像并将容器映像部署到 Amazon ECR [amazon-ecr-login](#)的工作流程。
- [amazon-ecs-render-task-definition](#) 工作流程，用于将容器镜像 URI 插入到 Amazon ECS 任务定义 JSON 文件中，从而创建新的任务定义文件。
- 用于注册 Amazon ECS 任务[amazon-ecs-deploy-task-definition](#)并将其部署到 [Amazon ECS Fargate 服务的-definition](#) 工作流程。

先决条件

本小节介绍完成本教程中的步骤所需的工具和资源。

Docker

- 为操作系统@@ [安装 Docker](#)。

AWS CLI

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。要完成本页上的步骤，您需要以下条件：

- [AWS CLI— 安装版本 2](#)。
- [AWS CLI— 使用aws configure. 进行快速配置](#)。

第一步：（可选）创建 Amazon VPC

您需要一个连接了互联网网关的 Amazon VPC、一个连接了 NAT 网关的私有子网以及一个分配了自动分配公有 IP 地址配置的公有子网。如果您不想修改用于 Amazon MWAA 环境的 Amazon VPC，请使用本节中的 AWS CloudFormation模板创建一个包含这些组件的新 VPC。

Note

完成此步骤后，保持AWS CloudFormation控制台处于打开状态，以复制“输出”选项卡中的详细信息。

1. 打开 [AWS CloudFormation 控制台](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择创建堆栈，模板已准备就绪，然后上传模板文件。
4. 使用mwaa_local_runner_vpc_public.yaml模板创建堆栈。我们建议提供堆栈名称mwaa-cicd-vpc。您也可以[下载该模板](#)。

Description: This template deploys a VPC, with a pair of public and private subnets across two Availability Zones (AZ). 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: A name that is prefixed to resource names
    Type: String
    Default: mwaa-cicd-

  VpcCIDR:
    Description: The IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.0.0.0/16

  PublicSubnet1CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the first
    Availability Zone
    Type: String
    Default: 10.0.0.0/24

  PrivateSubnet1CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the first
    Availability Zone
    Type: String
    Default: 10.0.1.0/24

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)

  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)

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

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

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

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

SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "mwa-cicd"
    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 ] ]

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

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

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

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

5. 创建 Amazon VPC 后，选择 Outputs，然后复制您的 Amazon VPC 的所有标识符。

第二步：构建 Docker 镜像

以下步骤显示了如何对[aws-mwaa-local-runner](#)存储库进行分支GitHub。默认情况下，它使用本地运行器的main分支，这是 Apache Airflow 的最新版本。可用的最新版本是 Apache Airflow v2.5.1。分叉是存储库的副本。分叉存储库允许您自由试用 DAG、Python 依赖项和自定义插件并将其推送到存储库，而不会影响原始项目。

1. 打开 [aws-mwaa-local-runner](#)，选择 Fork。
2. 打开命令提示符并在本地创建项目目录，然后将分叉的存储库克隆到此目录中。例如：

```
mkdir mwaa-cicd
cd mwaa-cicd
git clone git@github.com:your-git-user-name/aws-mwaa-local-runner.git
cd aws-mwaa-local-runner
```

3. 可选-macOS。打开 [Docker 桌面](#) 程序。
4. 返回命令提示符窗口，在aws-mwaa-local-runner目录中构建 Docker 容器镜像。

```
./mwaa-local-env build-image
```

5. 启动本地 Apache Airflow 环境。

```
./mwaa-local-env start
```

6. [通过 http://localhost:8080/](http://localhost:8080/) 登录 Apache Airflow：

- a. 用户名：管理员

b. 密码：测试

7. 在你分叉存储库并确认 Apache Airflow 正在本地运行后，使用键盘上的 C trl + C 来停止容器。

第三步：在亚马逊 ECR 中创建 Docker 镜像注册表

您需要在AWS Identity and Access Management (IAM) 中为用户、群组或角色提供特定权限才能完成本教程中的步骤。本部分包含创建 Amazon ECR 存储库并附加所需最低权限的AWS CloudFormation模板。默认情况下，该模板为AWS账户中的任何用户提供访问权限。您可以将示例 ARN 替换为 IAM 中需要访问您的 CD 管道中AWS资源的任何现有角色、用户或群组。

Note

完成此步骤后，保持AWS CloudFormation控制台处于打开状态，以复制“输出”选项卡中的详细信息。

1. 打开 [AWS CloudFormation 控制台](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择创建堆栈，模板已准备就绪，然后上传模板文件。
4. 使用mwaa_local_runner_ecr_repo.yaml模板创建堆栈。我们建议提供堆栈名称mwaa-cicd-ecr。您也可以[下载该模板](#)。

```
AWS::TemplateFormatVersion: '2010-09-09'
Description: Creates an ECR repository with the permissions needed to push and pull
images. It gives any user in the account access.
Parameters:
  RepoName:
    Type: String
    Default: mwaa-cicd
    Description: The name of the ECR repository.

Resources:
  Repository:
    Type: AWS::ECR::Repository
    Properties:
      RepositoryName: !Ref RepoName
      RepositoryPolicyText:
        Version: "2012-10-17"
        Statement:
          - Sid: AllowPushPull
            Effect: Allow
            Principal:
              AWS:
                - !Sub "arn:aws:iam::${AWS::AccountId}:root"
            Action:
              - "ecr:GetAuthorizationToken"
              - "ecr:GetRepositoryPolicy"
              - "ecr:GetDownloadUrlForLayer"
              - "ecr:BatchGetImage"
              - "ecr:BatchCheckLayerAvailability"
              - "ecr:PutImage"
              - "ecr:ListImages"
              - "ecr:DescribeImages"
              - "ecr:DescribeRepositories"
              - "ecr:InitiateLayerUpload"
              - "ecr:UploadLayerPart"
              - "ecr:CompleteLayerUpload"

Outputs:
  EcrRepoName:
    Value: !Ref RepoName
```

```
EcrRepositoryUri:  
Value: !GetAtt Repository.RepositoryUri
```

5. 创建存储库后，选择“输出”，然后将 Amazon ECR URI 复制到 EcrRepositoryUri。您也可以使用中的 desc [rube-re](#) positoriAWS CLI es 命令在中检索存储库的 URIrepositoryUri。

```
aws ecr describe-repositories --repository-names mwaa-cicd
```

第四步：对 Docker 客户端进行身份验证

以下步骤说明了如何将您的 Amazon ECR 证书添加到 Docker CLI。这将检索身份验证令牌并将您的 Docker 客户端身份验证到您的 Amazon ECR 私有注册表。

1. 在命令提示符下，将目录更改为本地运行器目录的根目录。例如：

```
cd mwaa-cicd/aws-mwaa-local-runner
```

2. 运行 Docker list image 命令，记下TAG和中amazon/mwaa-local值的名称IMAGE ID。

```
docker image ls
```

3. 运行[get-login-password](#)命令进行身份验证。这授予 Docker CLI 访问您的AWS账户的权限。将中的region值替换为您在前面步骤中指定的AWS区域。例如：

```
aws ecr get-login-password --region YOUR_REGION | docker login --username AWS --  
password-stdin YOUR_ACCOUNT_ID.dkr.ecr.YOUR_REGION.amazonaws.com
```

输出应与以下内容类似：

```
Login Succeeded
```

第五步：推送 Docker 镜像

本部分将引导您完成标记 Docker 镜像并将其推送到 Amazon ECR 的步骤。

标记镜像

1. 如果您还没有进入命令提示符，请将目录更改为本地运行器目录的根目录。例如：

```
cd mwaa-cicd/aws-mwaa-local-runner
```

2. 使用存储库 URL 运行带有前面步骤中的amazon/mwaa-local:2.0标签的 Docker 标签命令。用你的URI 替换样本值。例如：

```
docker image tag amazon/mwaa-  
local:2.0 YOUR_ACCOUNT_ID.dkr.ecr.YOUR_REGION.amazonaws.com/mwaa-cicd:2.0
```

响应不包含任何输出。

3. 验证您的存储库的 Docker 标签。

```
docker image ls
```


输出应与以下内容类似：

REPOSITORY	CREATED	SIZE	TAG	IMAGE ID
YOUR_ACCOUNT.dkr.ecr.us-east-1.amazonaws.com/mwaa-cicd	5f6473ce154e	About an hour ago	1.33GB	2.0
amazon/mwaa-local	5f6473ce154e	About an hour ago	1.33GB	2.0
postgres	17ec9988ae21	2 days ago	72.8MB	10-alpine

4. 运行 `Docker push` 命令将映像推送到存储库。用你的 URI 替换样本值。例如：

```
docker image push YOUR_ACCOUNT_ID.dkr.ecr.YOUR_REGION.amazonaws.com/mwaa-cicd:2.0
```

5. 输出应与以下内容类似：

```
5f70bf18a086: Pushed
299dc11cc9fc: Pushed
999fe11097c8: Pushed
99a9119976aa: Pushed
99c8a1192b0a: Pushed
ccccac999f7f: Pushed
499481199891: Pushed
ca9911963715: Pushed
cf299ec9e179: Pushed
964c9939e7a2: Pushed
3d99d0d9910f: Pushed
919bfe99a642: Pushed
ae99f399c8e7: Pushed
latest: digest: sha256:ee0008e584b009be056ef1c0d46f00438b316111c80009141114e45b115d9a52
size: 3033
```

6. 运行 `aws ecr describe-images` 命令查看注册表摘要。您还可以在亚马逊 ECR 控制台的存储[库页面](#)中访问 Amazon ECR 存储库摘要。

```
aws ecr describe-images --repository-name mwaa-cicd
```

输出应与以下内容类似：

```
{
  "imageDetails": [
    {
      "registryId": "YOUR_ACCOUNT_ID",
      "repositoryName": "mwaa-cicd",
      "imageDigest":
        "sha256:ee0008e584b009be056ef1c0d46f00438b316111c80009141114e45b115d9a52",
      "imageTags": [
        "latest"
      ],
      "imageSizeInBytes": 504790381,
      "imagePushedAt": "2021-08-26T15:39:43-06:00",
      "imageManifestMediaType": "application/
vnd.docker.distribution.manifest.v2+json",
      "artifactMediaType": "application/vnd.docker.container.image.v1+json"
    }
  ]
}
```

第六步：创建 Amazon ECS 集群

需要任务定义才能在 Amazon ECS 集群中运行 Docker 容器。以下AWS CloudFormation模板使用本地运行器的 Docker 映像部署了 Amazon ECS Fargate 集群。它包含 Fargate 的任务定义和 Amazon ECS 所需的执行角色策略。

尽管您可以在这些步骤中选择与 Amazon MWAA 环境相同的 Amazon VPC、公有子网和安全组，但此模板中的描述假定您使用的是前面步骤中创建的 Amazon VPC 组件。

1. 打开 [AWS CloudFormation 控制台](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择创建堆栈，模板已准备就绪，然后上传模板文件。
4. 使用mwaa_local_runner_ecs_cluster.yaml模板创建堆栈。将中的Image值替换为您的 Amazon ECR 存储库 URI，包括标签。我们建议提供堆栈名称mwaa-cicd-ecs。您也可以[下载该模板](#)。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: This template deploys an ECS Fargate cluster with an Amazon Linux
image.
Parameters:
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: Select your VPC. If you are using the VPC from the CD tutorial,
select the mwaa-cicd VPC.
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Select your subnet. If you are using the subnet from the CD
tutorial, select mwaa-cicd-Public Subnet (AZ1).
  SecurityGroups:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Select your security group. If you are using the security group from
the CD tutorial, select the mwaa-cicd security group.
Resources:
  Cluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Sub "${AWS::StackName}"

  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: !Sub "${AWS::StackName}"
      Cpu: 2048
      Memory: 4096
      NetworkMode: awsvpc
```

```
ExecutionRoleArn: !Ref ExecutionRole
ContainerDefinitions:
  - Name: !Sub "${AWS::StackName}"
    Image: YOUR_ACCOUNT_ID.dkr.ecr.YOUR_REGION.amazonaws.com/mwaa-cicd:2.0
    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: !Sub "${AWS::StackName}"
    Cluster: !Ref Cluster
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: 1
    LaunchType: FARGATE
    PlatformVersion: 1.3.0
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        Subnets: !Ref SubnetIds
        SecurityGroups: !Ref SecurityGroups
```

第七步：将任务定义推送到 GitHub

以下步骤显示如何将 Amazon ECS 集群的任务定义添加到分叉 GitHub 存储库中。

将任务定义保存到本地

1. 打开 [Amazon ECS 控制台](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择“任务定义”mwaa-cicd-ecs，然后选择您的任务定义。
4. 请选择 JSON 选项卡。
5. 在本地复制并另存为task-def.json。

将任务定义推送到 GitHub

1. 将任务定义复制到分叉存储库的根目录中。例如，在 macOS 的终端中，请执行以下操作：

```
mv ~/Downloads/task-def.json ~/mwaa-cicd/aws-mwaa-local-runner/task-def.json
```

2. 将目录更改为本地运行器目录的根目录。例如：

```
cd mwaa-cicd/aws-mwaa-local-runner
```

3. 验证您的分叉存储库的上游。

```
git remote -vv
```

输出应与以下内容类似：

```
origin git@github.com:your-git-user-name/aws-mwaa-local-runner.git (fetch)
origin git@github.com:your-git-user-name/aws-mwaa-local-runner.git (push)
```

4. 获取 git 分支的状态。

```
git status
```

5. 将任务定义推送到存储库。

```
git add task-def.json
git commit -m "Create initial commit"
git push -u origin main
```

第八步：将您AWS和亚马逊 ECR 信息添加到 GitHub

1. 在浏览器中打开分叉存储库。例如，<https://github.com/your-git-user-name/aws-mwaa-local-runner/>。
2. 选择“设置”、“密钥”、“新存储库密钥”。
3. 添加以下键/值对：
 - a. AWS_ACCESS_KEY_ID在“名称”中添加您的访问密钥 ID 的值，然后选择“添加密钥”。
 - b. 添加名称，AWS_SECRET_ACCESS_KEY在值中添加您的密钥的值，然后选择添加密钥。
 - c. 可选-添加名称，AWS_SESSION_TOKEN在值中添加会话令牌的值，然后选择添加密钥。
 - d. 添加“名称”，ECR_REPOSITORY_URI在“值”中添加 Amazon ECR 存储库 URI 的值，然后选择添加密钥，不包括标签。例如：

```
YOUR_ACCOUNT.dkr.ecr.us-east-1.amazonaws.com/mwaa-cicd
```

第九步：设置入 GitHub 门工作流程

工作流程是由一个或多个任务组成的可配置的自动化流程。以下步骤使用[amazon-ecr-login](#) GitHub 操作工作流程。在您的分叉存储库中更改文件后，工作流程会登录到您的 Amazon ECR 注册表以提取图像并将其推送到您的存储库。

1. 在浏览器中打开分叉存储库。例如，<https://github.com/your-git-user-name/aws-mwaa-local-runner/>。
2. 选择“操作”，然后为“部署到 Amazon ECS”工作流程设置此工作流程。
3. 复制以下模板并粘贴到文本字段中。将值替换为aws-region。您也可以[下载该模板](#)。

```
name: Deploy to Amazon ECS

on:
  push:
    branches:
      - main

env:
  AWS_REGION: us-east-1           # set this to your AWS region, e.g. us-west-2
  ECR_REPOSITORY: mwaa-cicd       # the Amazon ECR repository name
  ECS_SERVICE: mwaa-cicd-ecs      # the Amazon ECS service name
```

```
ECS_CLUSTER: maa-cicd-ecs          # the name of the Amazon ECS cluster name
ECS_TASK_DEFINITION: task-def.json # the path to the Amazon ECS task definition
CONTAINER_NAME: maa-cicd-ecs       # the name of the Amazon ECS container
IMAGE_TAG: '2.0'                  # the Amazon ECR container image tag
LOCAL_TAG: amazon/maa-local

defaults:
  run:
    shell: bash

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest # the GitHub Action VM Runner to execute the workflow
    permissions:
      packages: write
      contents: read

    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          # aws-session-token: ${ secrets.AWS_SESSION_TOKEN } # add if you use an AWS
session token
          aws-region: ${ env.AWS_REGION }

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v1

      - name: Build, tag, and push image to Amazon ECR
        id: build-image
        env:
          ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
          IMAGE_TAG: ${ env.IMAGE_TAG }
          LOCAL_TAG: ${ env.LOCAL_TAG }
        run: |
          # Build the docker container and push it to ECR so that it can be deployed to
ECS.
          ./maa-local-env build-image
          docker tag $LOCAL_TAG:$IMAGE_TAG ${ secrets.ECR_REPOSITORY_URI }:$IMAGE_TAG
          docker push ${ secrets.ECR_REPOSITORY_URI }:$IMAGE_TAG
          echo "::set-output name=image::${ secrets.ECR_REPOSITORY_URI }:$IMAGE_TAG"

      - name: Fill in the new image ID in the Amazon ECS task definition
        id: task-def
        uses: aws-actions/amazon-ecs-render-task-definition@v1
        with:
          task-definition: ${ env.ECS_TASK_DEFINITION }
          container-name: ${ env.CONTAINER_NAME }
          image: ${ steps.build-image.outputs.image }

      - name: Deploy Amazon ECS task definition
        uses: aws-actions/amazon-ecs-deploy-task-definition@v1
        with:
          task-definition: ${ steps.task-def.outputs.task-definition }
          service: ${ env.ECS_SERVICE }
          cluster: ${ env.ECS_CLUSTER }
          wait-for-service-stability: true
```

4. 选择“开始提交”，然后在“描述”字段中添加 ECS 工作流程。

5. 选择“直接提交到主分支，提交新文件”。

第十步：测试工作 GitHub 流程

要测试工作流程，您可以将 Amazon ECS 任务定义或 GitHub 工作流程文件（中 `aws.yaml`）的更新推送到中的本地运行器存储库 GitHub。以下步骤显示如何推送任务定义的更新以运行工作流程。

1. 在命令提示符下，将目录更改为本地运行器目录的根目录。例如：

```
cd mwa-cicd/aws-mwa-local-runner
```

2. 提取最新更改。

```
git pull
```

3. 打开任务定义并在 `task-def.json` 文件中稍作更改，例如在最后一个方括号 (}) 之后添加一个空格。

```
open task-def.json
```

4. 获取分支的状态。

```
git status
```

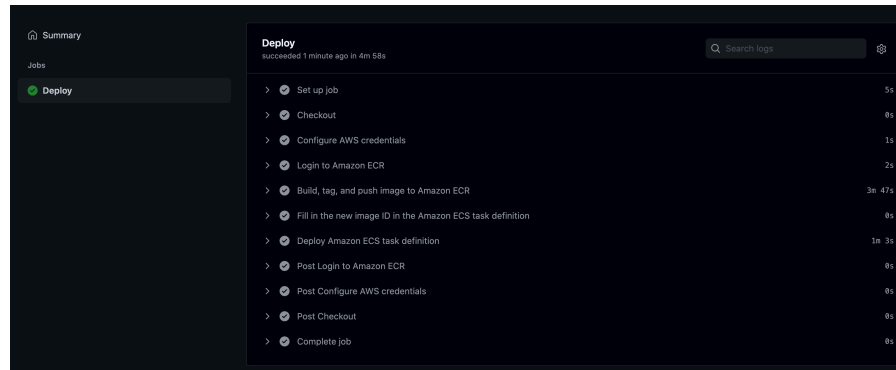
输出应与以下内容类似：

```
modified:   task-def.json
```

5. 将任务定义推送到存储库。

```
git add task-def.json
git commit -m "Add modified task definition"
git push -u origin main
```

6. 打开“操作”选项卡，在存储库中部署到 Amazon ECS。您应看到工作流程正在运行：



清除

您可以使用以下步骤删除您在本教程中创建的任何AWS资源。

1. 打开 [AWS CloudFormation 控制台](#)。
2. 选择要删除的堆栈，选择删除，然后选择删除堆栈。

接下来做什么？

- 在 <https://github.com/aws-actions> 浏览所有可用AWS的 A GitHub ctions。

教程：使用配置专用网络访问权限AWS Client VPN

本教程将引导您完成为适用于 Apache Airflow (MWAA) 的亚马逊托管工作流程 (MWAA) 环境创建从计算机到 Apache Airflow Web 服务器的 VPN 隧道的步骤。要通过 VPN 隧道连接到互联网，您首先需要创建 AWS Client VPN终端节点。设置后，Client VPN 端点充当 VPN 服务器，允许从您的计算机安全连接到 VPC 中的资源。然后，您将使用 for Desktop 从计算机连接到Client VPN。AWS Client VPN

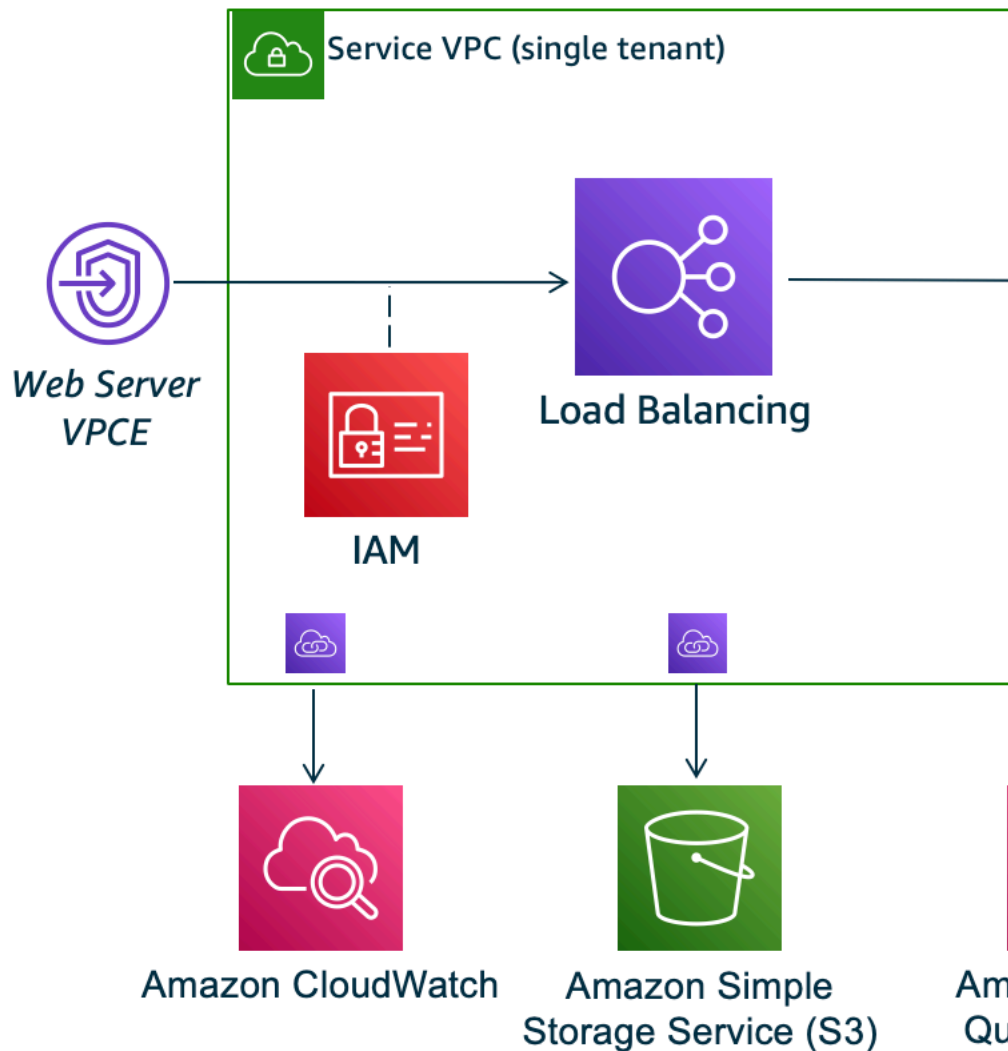
小节目录

- [私有网络 \(p. 155\)](#)
- [使用案例 \(p. 157\)](#)
- [开始前的准备工作 \(p. 157\)](#)
- [目标 \(p. 157\)](#)
- [\(可选 \) 第一步：确定您的 VPC、CIDR 规则和 VPC 安全性 \(p. 157\)](#)
- [步骤二：创建服务器和客户端证书 \(p. 158\)](#)
- [第三步：将AWS CloudFormation模板保存到本地 \(p. 159\)](#)
- [第四步：创建Client VPNAWS CloudFormation 堆栈 \(p. 160\)](#)
- [第五步：将子网关联到您的Client VPN \(p. 160\)](#)
- [第六步：向您的Client VPN 添加授权入口规则 \(p. 160\)](#)
- [步骤七：下载 Client VPN 终端节点配置文件 \(p. 161\)](#)
- [第八步：Connect 到AWS Client VPN \(p. 162\)](#)
- [接下来做什么？ \(p. 162\)](#)

私有网络

本教程假设你已经为 Apache Airflow Web 服务器选择了专用网络访问模式。

Private Web Server Opt



私有网络访问模式将对 Apache Airflow UI 的访问权限限制在您的 Amazon VPC 中，这些用户已被授予[对您环境的 IAM 策略的 \(p. 32\)](#)访问权限。

当您创建具有私有 Web 服务器访问权限的环境时，必须将所有依赖项打包到 Python wheel 存档中 (.whl)，然后在 .whl 中引用 requirements.txt。有关使用 wheel 打包和安装依赖项的说明，请参阅[使用 Python 轮子管理依赖关系 \(p. 250\)](#)。

下图显示了在 Amazon MWAA 控制台上可以找到私有网络选项的位置。

Web server access

☒ **Private network (Recommended)**

Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.

☐ **Public network (No additional setup)**

Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

使用案例

您可以在创建 Amazon MWAA 环境之前或之后使用本教程。您必须使用与您的环境相同的 Amazon VPC、VPC 安全组和私有子网。如果您在创建 Amazon MWAA 环境后使用本教程，则完成这些步骤后，您可以返回亚马逊 MWAA 控制台并将您的 Apache Airflow Web 服务器访问模式更改为专用网络。

开始前的准备工作

1. 检查用户权限。确保您在 AWS Identity and Access Management (IAM) 中的账户有足够的权限来创建和管理 VPC 资源。
2. 使用您的亚马逊 MWAA VPC。本教程假设您正在将 Client VPN 关联到现有 VPC。Amazon VPC 必须与 Amazon MWAA 环境位于同一 AWS 区域，并且有两个私有子网。如果您尚未创建 Amazon VPC，请使用其中的 AWS CloudFormation 模板 [选项三：创建没有互联网访问权限的 Amazon VPC 网络 \(p. 23\)](#)。

目标

在本教程中，您将执行以下操作：

1. 使用现有 Amazon VPC 的 AWS CloudFormation 模板创建 AWS Client VPN 终端节点。
2. 生成服务器和客户端证书和密钥，然后将服务器证书和密钥上传到 AWS Certificate Manager Amazon MWAA 环境所在的 AWS 区域。
3. 下载并修改 Client VPN 的 Client VPN 端点配置文件，然后使用该文件创建 VPN 配置文件以使用桌面版 Client VPN 进行连接。

(可选) 第一步：确定您的 VPC、CIDR 规则和 VPC 安全性

以下部分介绍如何为您的 Amazon VPC、VPC 安全组查找 ID，以及确定在后续步骤中创建 Client VPN 所需的 CIDR 规则的方法。

确定您的 CIDR 规则

以下部分说明如何识别 CIDR 规则，创建 Client VPN 需要这些规则。

识别您的 Client VPN 的 CIDR

1. 打开 [Amazon VPC 控制台的 Amazon VPC \(端点 \) 页面](#)。
2. 使用导航栏中的区域选择器选择与 Amazon MWAA 环境相同的 AWS 区域。
3. 选择您的 Amazon VPC。
4. 假设您的私有子网的 CIDR 是：
 - 私有子网 1 : 10.192.10.0/24

- 私有子网 2：10.192.11.0/24

如果您的 Amazon VPC 的 CIDR 是 10.192.0.0/16，那么您为 Client VPN 指定的客户端 IPv4 CIDR 将是 10.192.0.0/22。

5. 保存此 CIDR 值以及您的 VPC ID 的值以供后续步骤使用。

识别您的 VPC 和安全组

以下部分介绍如何查找您的 Amazon VPC 和安全组的 ID，创建 Client VPN 需要这些 ID。

Note

您可能正在使用一个以上的安全组。您需要在后续步骤中指定您的 VPC 的所有安全组。

识别安全组

1. 打开 Amazon VPC 控制台上的[安全组页面](#)。
2. 使用导航栏中的区域选择器选择 AWS 区域。
3. 在 VPC ID 中查找 Amazon VPC，并确定与 VPC 相关的安全组。
4. 保存您的安全组和 VPC 的 ID 以供后续步骤使用。

步骤二：创建服务器和客户端证书

客户端 VPN 终端节点仅支持 1024 位和 2048 位 RSA 密钥大小。以下部分说明了如何使用 OpenVPN easy-rsa 生成服务器和客户端证书和密钥，然后使用 AWS Command Line Interface (AWS CLI) 将证书上传到 ACM。

创建客户证书

1. 按照以下快速步骤，通过 `in C` [lient 身份验证和授权：相互身份验证，创建证书](#) 并将其上传到 ACM。AWS CLI
2. 在这些步骤中，您必须在上传服务器和客户端证书时在 AWS CLI 命令中指定与 Amazon MWAA 环境相同的 AWS 区域。以下是如何在这些命令中指定区域的一些示例：

- a. Example 服务器证书区域

```
aws acm import-certificate --certificate fileb://server.crt --private-key fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
```

- b. Example 客户端证书区域

```
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. 完成这些步骤后，保存服务器证书和客户端证书 ARN 的 AWS CLI 响应中返回的值。您将在 AWS CloudFormation 模板中指定这些 ARN 来创建 Client VPN。
3. 在这些步骤中，客户端证书和私钥将保存到您的计算机中。以下示例说明了在哪里可以找到这些证书：
 - a. Example 在 macOS 上

在 macOS 上，内容保存在 `/Users/youruser/custom_folder`。如果你列出了这个目录的所有 (`ls -a`) 内容，你应该会看到与以下内容类似的内容：

```
.  
..  
ca.crt  
client1.domain.tld.crt  
client1.domain.tld.key  
server.crt  
server.key
```

- b. 完成这些步骤后，保存内容或记下客户端证书的位置，并在中client1.domain.tld.crt记下私有密钥的位置client1.domain.tld.key。您将把这些值添加到您的Client VPN 的配置文件中。

第三步：将AWS CloudFormation模板保存到本地

以下部分包含用于创建Client VPN 的AWS CloudFormation模板。您必须指定与您的亚马逊 MWAA 环境相同的 Amazon VPC、VPC 安全组和私有子网。

- 复制以下模板的内容并将其作为保存到本地mwaa_vpn_client.yaml的。您也可以[下载该模板](#)。

替换以下值：

- **YOUR_CLIENT_ROOT_CERTIFICATE_ARN**— 中的 client1.domain.tld 证书的 ARNClientRootCertificateChainArn。
- **YOUR_SERVER_CERTIFICATE_ARN**— 中的服务器证书的 ARNServerCertificateArn。
- 中的客户端 IPv4 CIDR 规则ClientCidrBlock。提供10.192.0.0/22的 CIDR 规则。
- 您的亚马逊 VPC ID 位于VpcId。提供的 VPCvpc-010101010101 为。
- 您的 VPC 安全组 ID 在中SecurityGroupIds。提供了sg-0101010101一个安全组。

```
AWS::CloudFormation::Template  
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

如果您在环境中使用多个安全组，则可以按以下格式指定多个安全组：

```
SecurityGroupIds:
  - sg-0112233445566778b
  - sg-0223344556677889f
```

第四步：创建Client VPNAWS CloudFormation 堆栈

创建 AWS Client VPN

1. 打开 [AWS CloudFormation 控制台](#)。
2. 选择模板已准备就绪，上传模板文件。
3. 选择“选择文件”，然后选择您的mwaa_vpn_client.yaml文件。
- 4.
5. 选择“下一步，下一步”。
6. 选择确认，然后选择创建堆栈。

第五步：将子网关联到您的Client VPN

将私有子网关联到AWS Client VPN

1. 打开 [亚马逊 VPC 控制台](#)。
2. 选择 Client VPN 端点页面。
3. 选择您的Client VPN，然后选择“关联”选项卡，“关联”。
4. 在下拉列表中选择以下内容：
 - 您在 VPC 中的亚马逊 VPC。
 - 选择要关联的子网中的一个私有子网。
5. 选择 Associate。

Note

VPC 和子网与Client VPN 关联需要几分钟时间。

第六步：向您的Client VPN 添加授权入口规则

您需要使用您的 VPC 的 CIDR 规则向您的Client VPN 添加授权入口规则。如果您想授权来自 Active Directory 组或基于 SAML 的身份提供商 (IdP) 的特定用户或群组，请参阅Client VPN 指南中的[授权规则](#)。

将 CIDR 添加到AWS Client VPN

1. 打开 [亚马逊 VPC 控制台](#)。
2. 选择 Client VPN 端点页面。
3. 选择您的Client VPN，然后选择“授权”选项卡，“授权入口”。
4. 指定以下内容：
 - 要启用目标网络中的 Amazon VPC 的 CIDR 规则。例如：

```
10.192.0.0/16
```

- 选择 Endpoints (授予访问权限) 中的允许所有用户访问。
 - 在描述中输入描述性名称。
5. 选择“添加授权规则”。

Note

根据您的 Amazon VPC 的网络组件，您可能还需要将此授权入口规则添加到您的网络访问控制列表 (NACL)。

步骤七：下载 Client VPN 终端节点配置文件

下载配置文件

1. 按照以下快速步骤在下载 Client VPN [端点配置文件中下载 Client VPN 配置文件](#)。
2. 在这些步骤中，要求您在 Client VPN 终端节点 DNS 名称前加一个字符串。示例如下：

- Example 终端节点 DNS 名称

如果您的 Client VPN 终端节点 DNS 名称如下所示：

```
remote cvpn-endpoint-0909091212aeee1.prod.clientvpn.us-west-1.amazonaws.com 443
```

你可以添加一个字符串来标识你的 Client VPN 端点，如下所示：

```
remote mwaavpn.cvpn-endpoint-0909091212aeee1.prod.clientvpn.us-west-1.amazonaws.com 443
```

3. 在这些步骤中，要求您在一组新标记之间添加客户端证书的内容，而在一组新<cert></cert>标记之间添加私有密钥的<key></key>内容。示例如下：
 - a. 打开命令提示符并将目录更改为客户端证书和私钥的位置。
 - b. Example macOS client1.domain.tld.crt

要在 macOS 上显示 client1.domain.tld.crt 文件内容，可以使用 cat client1.domain.tld.crt。

从终端复制值并 downloaded-client-config.ovpn 像这样粘贴：

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

- c. Example macOS client1.domain.tld.key

要显示的内容 client1.domain.tld.key，可以使用 cat client1.domain.tld.key。

从终端复制值并 downloaded-client-config.ovpn 像这样粘贴：

```
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>
```

第八步：Connect 到AWS Client VPN

AWS Client VPN的客户端免费提供。您可以将计算机直接连接到，AWS Client VPN以获得 end-to-end VPN 体验。

连接到Client VPN

1. 下载并安装[AWS Client VPN桌面版](#)。
2. 打开 AWS Client VPN。
3. 在 VPN 客户端菜单中选择文件、托管配置文件。
4. 选择“添加个人资料”，然后选择downloaded-client-config.ovpn。
5. 在“显示名称”中输入描述性名称。
6. 选择“添加个人资料”，“完成”。
7. 选择 Connect (连接)。

连接到Client VPN 后，您需要断开与其他 VPN 的连接才能查看 Amazon VPC 中的任何资源。

Note

您可能需要退出客户端，然后重新启动，然后才能建立连接。

接下来做什么？

- 了解如何在中创建 Amazon MWAA 环境[开始使用 Amazon Managed Workflows of Apache Airflow \(MWAA\) \(p. 15\)](#)。您必须在与Client VPN 相同的AWS区域中创建环境，并使用与Client VPN 相同的VPC、私有子网和安全组。

教程：使用 Linux 堡垒主机配置专用网络访问权限

本教程将引导您完成在 Apache Airflow (MWAA) 环境中创建从计算机到 Apache Airflow Web 服务器的 SSH 隧道的步骤。它假设您已经创建了亚马逊 MWAA 环境。设置后，Linux 堡垒主机将充当跳转服务器，允许从您的计算机安全连接到 VPC 中的资源。然后，您将使用 SOCKS 代理管理附加组件控制浏览器中的代理设置，以访问 Apache Airflow 用户界面。

小节目录

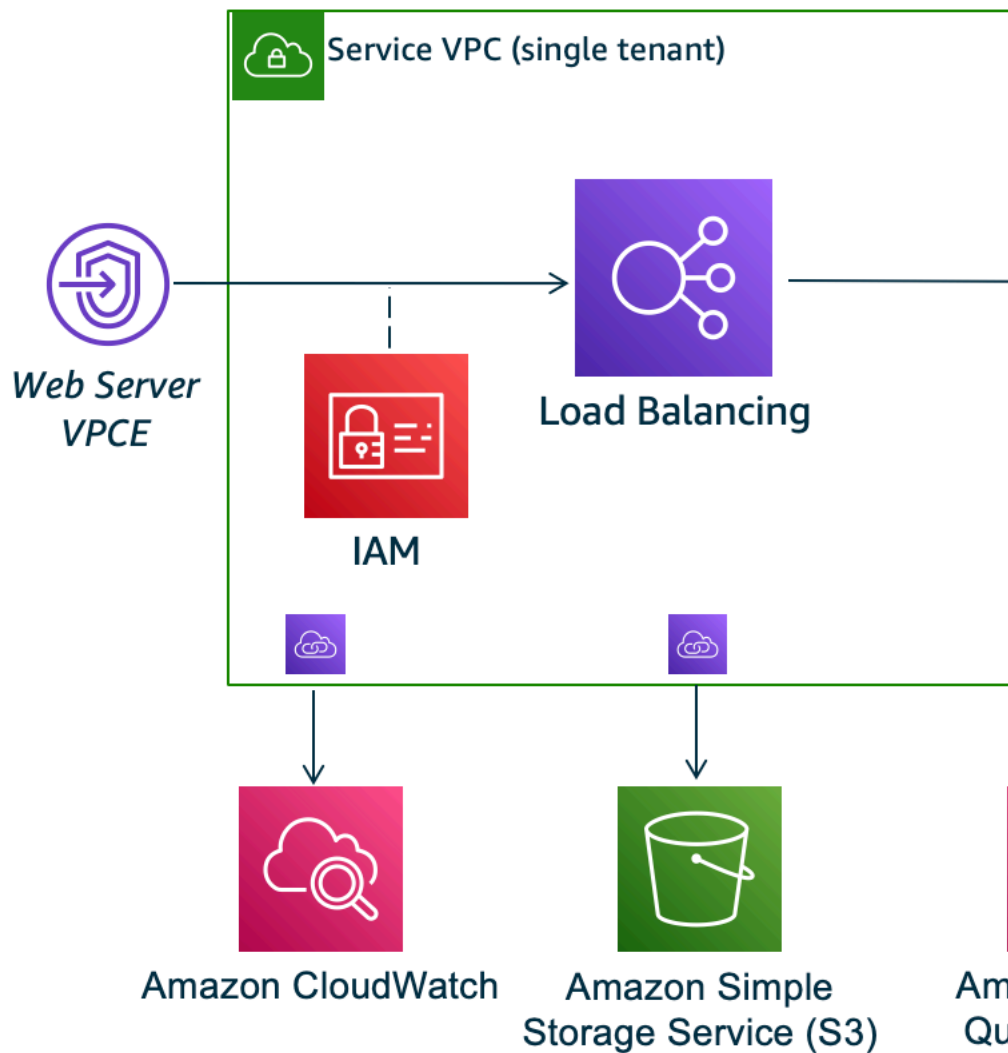
- [私有网络 \(p. 163\)](#)
- [使用案例 \(p. 164\)](#)
- [开始前的准备工作 \(p. 164\)](#)
- [目标 \(p. 164\)](#)
- [步骤一：创建堡垒实例 \(p. 164\)](#)
- [步骤二：创建 ssh 隧道 \(p. 165\)](#)

- [第三步：将堡垒安全组配置为入站规则 \(p. 166\)](#)
- [第四步：复制 Apache Airflow 网址 \(p. 166\)](#)
- [第五步：配置代理设置 \(p. 166\)](#)
- [第六步：打开 Apache Airflow 用户界面 \(p. 168\)](#)
- [接下来做什么？ \(p. 168\)](#)

私有网络

本教程假设你已经为 Apache Airflow Web 服务器选择了专用网络访问模式。

Private Web Server Opt



私有网络访问模式将对 Apache Airflow UI 的访问权限限制在您的 Amazon VPC 中，这些用户已被授予[对您的环境的 IAM 策略的 \(p. 32\)](#)访问权限。

创建具有私有 Web 服务器访问权限的环境时，必须将所有依赖项打包到 Python 轮子存档 (.whl) 中，然后在 .whl 中引用 requirements.txt。有关使用 wheel 打包和安装依赖项的说明，请参阅[使用 Python 轮子管理依赖关系 \(p. 250\)](#)。

下图显示了在 Amazon MWAA 控制台上可以找到私有网络选项的位置。

Web server access

- ☒ Private network (Recommended)
Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.
- ☐ Public network (No additional setup)
Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

使用案例

您可以在创建 Amazon MWAA 环境后使用本教程。您必须使用与您的环境相同的 Amazon VPC、VPC 安全组和公有子网。

开始前的准备工作

1. 检查用户权限。确保您在 AWS Identity and Access Management (IAM) 中的账户有足够的权限来创建和管理 VPC 资源。
2. 使用您的亚马逊 MWAA VPC。本教程假定您正在将堡垒主机关联到现有 VPC。Amazon VPC 必须与您的 Amazon MWAA 环境位于同一区域，并且有两个私有子网，如中所定义[创建 VPC 网络 \(p. 18\)](#)。
3. 创建 SSH 密钥。您需要在与亚马逊 MWAA 环境相同的区域创建 Amazon EC2 SSH 密钥 (.pem) 才能连接到虚拟服务器。如果您没有 SSH 密钥，请参阅 Amazon EC2 用户指南 (适用于 Linux 实例) 中的[创建或导入 key pair](#)。

目标

在本教程中，您将执行以下操作：

1. 使用[现有 VPC 的 AWS CloudFormation 模板](#)创建 Linux 堡垒主机实例。
2. 使用端口上的入口规则向堡垒实例的安全组授权入站流量 22。
3. 授权从 Amazon MWAA 环境的安全组到堡垒实例的安全组的入站流量。
4. 创建通往堡垒实例的 SSH 隧道。
5. 安装并配置 Firefox 浏览器的 FoxyProxy 插件以查看 Apache Airflow 用户界面。

步骤一：创建堡垒实例

以下部分介绍在 AWS CloudFormation 控制台上使用[现有 VPC 的 AWS CloudFormation 模板](#)创建 Linux 堡垒实例的步骤。

创建 Linux 堡垒主机

1. 在 AWS CloudFormation 控制台上打开“[部署快速入门](#)”页面。
2. 使用导航栏中的区域选择器选择与您的 Amazon MWAA 环境相同的 AWS 区域。

3. 选择 Next (下一步)。
4. 在堆栈名称文本字段中键入名称，例如mwaa-linux-bastion。
5. 在参数、网络配置窗格上，选择以下选项：
 - a. 选择您的亚马逊 MWAA 环境的 VPC ID。
 - b. 选择您的亚马逊 MWAA 环境的公有子网 1 ID。
 - c. 选择您的亚马逊 MWAA 环境的公有子网 2 ID。
 - d. 在允许的堡垒外部访问 CIDR 中输入尽可能窄的地址范围（例如，内部 CIDR 范围）。

Note

识别范围的最简单方法是使用与公有子网相同的 CIDR 范围。例如，[创建 VPC 网络页面上的AWS CloudFormation模板中的公有子网 \(p. 18\)](#)是10.192.10.0/24和10.192.11.0/24。

6. 在 Amazon EC2 配置窗格上，选择以下选项：
 - a. 在密钥对名称的下拉列表中选择您的 SSH 密钥。
 - b. 在堡垒主机名中输入名称。
 - c.
 - d. 为 TCP 转发选择真。

Warning

在此步骤中，必须将 TCP 转发设置为 true。否则，您就能在下步骤中创建 SSH 隧道。

- e.
7. 选择“下一步，下一步”。
8. 选择确认，然后选择创建堆栈。

要了解有关 Linux 堡垒主机架构的更多信息，请参阅[AWS云上的 Linux 堡垒主机：架构](#)。

步骤二：创建 ssh 隧道

以下步骤介绍如何创建 Linux 堡垒的 SSH 隧道。SSH 隧道接收来自您的本地 IP 地址发往 Linux 堡垒的请求，这就是true在前面的步骤中将 linux 堡垒的 TCP 转发设置为的原因。

macOS/Linux

通过命令行创建隧道

1. 打开 Amazon EC2 控制台上的[实例](#)页面。
2. 选择一个实例。
3. 复制公有 IPv4 DNS 中的地址。例如，ec2-4-82-142-1.compute-1.amazonaws.com。
4. 在命令提示符下，导航到 SSH 密钥的存储目录。
5. 运行以下命令以使用 ssh 连接到堡垒实例。将示例值替换为中的 SSH 密钥名称mykeypair.pem。

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

使用 PuTTY 创建隧道

1. 打开 Amazon EC2 控制台上的[实例](#)页面。

2. 选择一个实例。
3. 复制公有 IPv4 DNS 中的地址。例如，`ec2-4-82-142-1.compute-1.amazonaws.com`。
4. 打开 [PuTTY](#)，选择会话。
5. 在主机名中输入主机名为 `ec2-user@YOUR_PUBLIC_IPV4_DNS`，将端口输入为 22。
6. 展开 SSH 选项卡，选择身份验证。在用于身份验证的私钥文件中，选择您的本地“ppk”文件。
7. 在 SSH 下，选择“隧道”选项卡，然后选择“动态”和“自动”选项。
8. 在源端口中，添加 8157 端口（或任何其他未使用的端口），然后将目标端口留空。选择 Add（添加）。
9. 选择会话选项卡并输入会话名称。例如 SSH Tunnel。
10. 选择“保存，打开”。

Note

您可能需要为公钥输入密码短语。

Note

如果您收到 `Permission denied (publickey)` 错误，我们建议使用 [AWS Support-TroubleshootSSH](#) 工具，然后选择 Run this Automation（控制台）对您的 SSH 设置进行故障排除。

第三步：将堡垒安全组配置为入站规则

允许访问服务器和从服务器定期访问互联网，但必须为这些服务器附加一个特殊的维护安全组。以下步骤描述如何将堡垒安全组配置为环境的 VPC 安全组的入站流量来源。

1. 在亚马逊 MWAA 控制台上打开 [环境页面](#)。
2. 选择一个环境。
3. 在网络窗格上，选择 VPC 安全组。
4. 选择 Edit inbound rules（编辑入站规则）。
5. 选择 Add rule。
6. 在 Source 下拉列表中选择您的 VPC 安全组 ID。
7. 将其余选项留空，或设置为其默认值。
8. 选择 Save rules（保存规则）。

第四步：复制 Apache Airflow 网址

以下步骤描述了如何打开亚马逊 MWAA 控制台并将网址复制到 Apache Airflow 用户界面。

1. 在亚马逊 MWAA 控制台上打开 [环境页面](#)。
2. 选择一个环境。
3. 复制 Airflow 用户界面中的 URL 以执行后续步骤。

第五步：配置代理设置

如果您使用带有动态端口转发的 SSH 隧道，则必须使用 SOCKS 代理管理附加组件来控制浏览器中的代理设置。例如，您可以使用 Chromium 的 `--proxy-server` 功能启动浏览器会话，也可以在 Mozilla Firefox 浏览器中使用 FoxyProxy 扩展程序。

选项一：使用本地端口转发设置 SSH 隧道

如果您不想使用 SOCKS 代理，可以使用本地端口转发设置一个 SSH 隧道。以下示例命令通过在本地端口 8157 上转发流量来访问 Amazon EC2 ResourceManagerWeb 界面。

1. 打开新的命令提示符窗口。
2. 键入以下命令以打开 SSH 隧道。

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-  
vpce.YOUR_REGION.airflow.amazonaws.com:443  
ubuntu@YOUR_PUBLIC_IPV4_DNS.YOUR_REGION.compute.amazonaws.com
```

-L 表示使用本地端口转发，由此，您就能指定一个本地端口，用于将数据转发到节点本地 Web 服务器上标识的远程端口。

3. `http://localhost:8157/` 在浏览器中键入。

Note

您可能需要使用 `https://localhost:8157/`。

选项二：通过命令行使用代理

大多数 Web 浏览器都允许您通过命令行或配置参数配置代理。例如，使用 Chromium，您可以使用以下命令启动浏览器：

```
chromium --proxy-server="socks5://localhost:8157"
```

这将启动一个浏览器会话，该会话使用您在之前的步骤中创建的 ssh 隧道来代理其请求。您可以按如下方式打开您的私有亚马逊 MWAA 环境网址（使用 `https://`）：

```
https://YOUR_VPC_ENDPOINT_ID-vpce.YOUR_REGION.airflow.amazonaws.com/home.
```

选项三：在 Mozilla Firefox 上使用的 FoxyProxy 代理

以下示例演示了 Mozilla Firefox 的 FoxyProxy 标准（版本 7.5.1）配置。FoxyProxy 提供了一套代理管理工具。它允许您使用代理服务器来访问与 Apache Airflow 用户界面所使用的域对应模式相匹配的 URL。

1. 在中 FireFox，打开 [FoxyProxy 标准](#) 扩展页面。
2. 选择“添加到火狐浏览器”。
3. 选择 Add（添加）。
4. 在浏览器工具栏中选择 FoxyProxy 图标，选择“选项”。
5. 复制以下代码并在本地另存为 `mwaa-proxy.json`。将 `YOUR_HOST_NAME` 中的示例值替换为你的 Apache Airflow URL。

```
{  
  "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. 在“从 FoxyProxy 6.0+ 导入设置”窗格中，选择“导入设置”并选择mwaa-proxy.json文件。
7. 选择 OK (确定)。

第六步：打开 Apache Airflow 用户界面

以下步骤介绍如何打开 Apache Airflow 用户界面。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择打开 Airflow UI。

接下来做什么？

- 了解如何在通往堡垒主机的 SSH 隧道上运行 Airflow CLI 命令[Apache Airflow Air \(p. 66\)](#)。
- 了解如何将 DAG 代码上载到 Amazon S3 存储桶[添加或更新 DAG \(p. 103\)](#)。

教程：限制亚马逊 MWAA 用户对 DAG 子集的访问权限

Amazon MWAA 通过将您的 IAM 委托人映射到 Apache Airflow 的一个或多个[默认角色](#)来管理对您的环境的访问权限。以下教程展示了如何限制个人 Amazon MWAA 用户仅查看特定的 DAG 或一组 DAG 并与之交互。

Note

只要可以担任 IAM 角色，就可以使用联合访问完成本教程中的步骤。

主题

- [先决条件 \(p. 169\)](#)
- [第一步：使用默认 APublic Apache Airflow 角色向您的 IAM 委托人提供亚马逊 MWAA Web 服务器访问权限。 \(p. 169\)](#)
- [步骤二：创建新的 Adit Aribules \(p. 170\)](#)
- [第三步：将您创建的角色分配给您的亚马逊 MWAA 用户 \(p. 170\)](#)
- [后续步骤 \(p. 171\)](#)
- [相关资源 \(p. 171\)](#)

先决条件

要完成本教程中的步骤，您需要以下条件：

- 具有[多个 DAG 的亚马逊 MWAA 环境 \(p. 15\)](#)
- Admin具有[AdministratorAccess](#)权限的 IAM 委托人和作为委托人的 IAM 用户，您可以限制 DAG 访问权限。MWAAUser有关管理员角色的更多信息，请参阅 IAM 用户指南中的[管理员工作职能](#)

Note

不要将权限策略直接附加到您的 IAM 用户。我们建议设置用户可以代入的 IAM 角色，以临时访问您的 Amazon MWAA 资源。

- [AWS Command Line Interface版本 2](#) 已安装。

第一步：使用默认 APublic Apache Airflow 角色向您的 IAM 委托人提供亚马逊 MWAA Web 服务器访问权限。

使用授予权限AWS Management Console

1. 使用Admin角色登录您的AWS账户并打开 [IAM 控制台](#)。
2. 在左侧导航窗格中，选择用户，然后从用户表中选择您的Amazon MWAA IM 用户。
3. 在用户详细信息页面的摘要下，选择权限选项卡，然后选择权限策略以展开卡片并选择添加权限。
4. 在授予权限部分中，选择直接附加现有策略，然后选择创建策略以创建和附加您自己的自定义权限策略。
5. 在创建策略页面上，选择 JSON，然后将以下 JSON 权限策略复制并粘贴到策略编辑器中。该策略向具有默认Public Apache Airflow 角色的用户授予 Web 服务器访问权限。

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

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "airflow:CreateWebLoginToken",
    "Resource": [
      "arn:aws:airflow:YOUR_REGION:YOUR_ACCOUNT_ID:role/YOUR_ENVIRONMENT_NAME/
Public"
    ]
  }
]
```

步骤二：创建新的 Adit Aribules

使用 Adit Aribules UI 创建新角色

1. 使用您的管理员 IAM 角色，打开[亚马逊 MWAA 控制台](#)并启动您环境的 Apache Airflow 用户界面。
2. 在顶部的导航窗格中，将鼠标悬停在“安全”上打开下拉列表，然后选择“列出角色”以查看默认 Apache Airflow 角色。
3. 从角色列表中选择“用户”，然后在页面顶部选择“操作”以打开下拉列表。选择“复制角色”，然后确认“确定”

Note

复制 Ops 或 Viewer 角色以分别授予更多或更少的访问权限。

4. 在表中找到您创建的新角色，然后选择编辑记录。
5. 在，执行以下操作：
 - 对于名称，在文本字段中键入角色的新名称。例如，**Restricted**。
 - 对于权限列表，移除can read on DAGs和can edit on DAGs，然后为要提供访问权限的 DAG 集添加读写权限。例如，对于 DAGexample_dag.py，添加**can read on DAG:example_dag**和**can edit on DAG:example_dag**。

选择 Save (保存)。现在，您应该拥有一个新角色来限制对您的 Amazon MWAA 环境中可用的 DAG 子集的访问权限。现在，您可以将此角色分配给任何现有的 Apache Airflow 用户。

第三步：将您创建的角色分配给您的亚马逊 MWAA 用户

分配新角色

1. 使用访问凭证MWAAUser，运行以下 CLI 命令来检索环境的 Web 服务器 URL。

```
$ aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq  
' .Environment.WebserverUrl '
```

如果成功，将会看到以下输出：

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. MWAAUser登录后AWS Management Console，打开一个新的浏览器窗口并访问以下 URL。
替Webserver-URL换为您的信息。

```
https://<Webserver-URL>/home
```

如果成功，您将看到一个Forbidden错误页面，因为MWAAUser尚未获得访问 Apache Airflow UI 的权限。

3. Admin登录后AWS Management Console，再次打开亚马逊 MWAA 控制台并启动您环境的 Apache Airflow 用户界面。
4. 在 UI 仪表板中，展开“安全”下拉列表，这次选择“列出用户”。
5. 在用户表中，找到新的 Apache Airflow 用户并选择“编辑记录”。用户的名字将按照以下模式与您的 IAM 用户名相匹配：user/*mwaa-user*。
6. 在编辑用户页面的角色部分中，添加您创建的新自定义角色，然后选择保存。

Note

姓氏字段是必填字段，但空格可以满足要求。

IAMPublic 委托人授予访问 Apache Airflow 用户界面的权限，而新角色则提供查看其 DAG 所需的额外权限。MWAAUser

Important

使用 Apache Airflow UI 添加的 5 个未经 IAM 授权的 5 个默认角色（例如Admin）中的任何一个都将在下次用户登录时删除。

后续步骤

- 要了解有关管理您的 Amazon MWAA 环境访问的更多信息，并查看您可以为环境用户使用的 JSON IAM 策略示例，请参阅[the section called “访问Amazon MWAA 环境” \(p. 32\)](#)

相关资源

- [访问控制](#) (Apache Airflow 文档) — 在 Apache Airflow 文档网站上了解有关默认 Apache Airflow 角色的更多信息。

Managed Workflows for Airflow (MWAA)

本指南包含代码示例，包括 DAG 和自定义插件，您可以在 Amazon Managed Workflows for Apache Airflow (MWAA) 环境。有关使用 Apache Airflow flowAWS服务，请参阅[example_dags](#)Apache Airflow 中的目录 GitHub 存储库。

示例

- [使用 DAG 在 CLI 中导入变量 \(p. 172\)](#)
- [使用创建 SSH 连接SSHOperator \(p. 174\)](#)
- [在中使用密钥AWS Secrets Manager用于 Apache Airflow 雪花连接 \(p. 177\)](#)
- [使用 DAG 写入自定义指标 CloudWatch \(p. 179\)](#)
- [在亚马逊 MWAA 环境中清理 Aurora PostgreSQL \(p. 181\)](#)
- [将环境元数据导出到 Amazon S3 上的 CSV 文件 \(p. 185\)](#)
- [在中使用密钥AWS Secrets Manager对于 Apache 气流变量 \(p. 187\)](#)
- [在中使用密钥AWS Secrets Manager用于 Apache 气流连接 \(p. 189\)](#)
- [使用 Oracle 创建自定义插件 \(p. 192\)](#)
- [创建生成运行时环境变量的自定义插件 \(p. 196\)](#)
- [在亚马逊 MWAA 上更改 DAG 的时区 \(p. 198\)](#)
- [Refresh CodeArtifact 代币 \(p. 200\)](#)
- [使用 Apache Hive 和 Hadoop 创建自定义插件 \(p. 204\)](#)
- [为 Apache Airflow 创建自定义插件 PythonVirtualenvOperator \(p. 207\)](#)
- [使用 Lambda 函数调用 DAG \(p. 211\)](#)
- [在不同的亚马逊 MWAA 环境中调用 DAG \(p. 213\)](#)
- [通过 Amazon MWAA fAmazon RDS for Microsoft SQL Server \(p. 215\)](#)
- [将亚马逊 MWAA 与亚马逊 EMR 结合使用 \(p. 218\)](#)
- [将适用于 Amazon MWAA \(p. 220\)](#)
- [使用连接到 Amazon ECSECSOperator \(p. 228\)](#)
- [在亚马逊 MWAA 上使用 dbt \(p. 234\)](#)
- [AWS博客和教程 \(p. 237\)](#)

使用 DAG 在 CLI 中导入变量

以下示例代码使用 Amazon Managed Workflows flows flows flow (MWAA) 导入变量。

主题

- [版本 \(p. 173\)](#)
- [先决条件 \(p. 173\)](#)
- [权限 \(p. 173\)](#)
- [附属物 \(p. 173\)](#)
- [代码示例 \(p. 173\)](#)
- [接下来做什么？ \(p. 174\)](#)

版本

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

- 无需额外权限即可使用此页面上的代码示例。

权限

你的AWS账户需要访问AmazonMWAAirflowCliAccess策略。要了解更多信息，请参阅 [Apache Airflow CLI 政策：AmazonmAirflowCliAccess \(p. 32\)](#)。

附属物

- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

以下示例代码需要三个输入：您的亚马逊 MWAA 环境名称（在mwaa_env），AWS您环境的区域（在aws_region），以及包含要导入的变量的本地文件（在var_file）。

```
import boto3
import json
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwaa_env=''
aws_region=''
var_file=''

try:
    opts, args = getopt.getopt(argv, 'e:v:r:', ['environment', 'variable-file', 'region'])
    #if len(opts) == 0 and len(argv) > 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 MAA environment -v variable file
location and filename -r aws region')
    print("Unexpected error:", sys.exc_info()[0])
    sys.exit(2)
```

接下来做什么？

- 学习如何将此示例中的 DAG 代码上传到dagsAmazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#).

使用创建 SSH 连接SSHOperator

以下示例描述如何使用SSHOperator在定向无环图 (MDW) 中连接到 Amazon EC2 实例。您可以使用类似的方法连接到任何具有 SSH 访问权限的远程实例。

在以下示例中，您上传了一个 SSH 密钥 (.pem) 到您的环境dagsAmazon S3 上的目录。然后，您可以使用以下命令安装必要的依赖项requirements.txt然后在用户界面中创建一个新的 Apache Airflow 连接。最后，您编写一个 DAG 来创建与远程实例的 SSH 连接。

主题

- [版本 \(p. 175\)](#)
- [先决条件 \(p. 175\)](#)
- [权限 \(p. 175\)](#)
- [要求 \(p. 175\)](#)
- [将您的密钥复制到 Amazon S3 \(p. 175\)](#)
- [创建新的 Apache Airflow 连接 \(p. 175\)](#)
- [代码示例 \(p. 176\)](#)

版本

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下命令：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MMWPlow \(p. 15\)](#).
- 一个 SSH 密钥。代码示例假定您有一个 Amazon EC2 实例和 .pem 位于您的 Amazon MWA 环境所在的同一区域。如果您没有密钥，请参阅[创建或导入密钥对](#)在里面适用 Linux 实例的 Amazon EC2 用户指南。

权限

- 无需额外权限即可使用此页面上的代码示例。

要求

将以下参数添加到 requirements.txt 安装 apache-airflow-providers-ssh 您的 Web 服务器上的程序包。环境更新并且 Amazon MWA 成功安装依赖项后，您将能够看到一个新的 SSH Apache Airflow UI 中的连接类型。

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.2.2/constraints-3.7.txt
apache-airflow-providers-ssh
```

Note

-c 定义了 URL 中的约束条件 requirements.txt。这可以确保 Amazon MWA 为您的环境安装正确的软件包版本。

将您的密钥复制到 Amazon S3

请使用以下命令 AWS Command Line Interface 命令复制您的 .pem 环境的关键 dags Amazon S3 中的目录。

```
$ aws s3 cp your-secret-key.pem s3://your-bucket/dags/
```

亚马逊 MWA 将内容复制到 dags，包括 .pem 密钥，到本地 /usr/local/airflow/dags/ 目录，通过这样做，Apache Airflow 可以访问密钥。

创建新的 Apache Airflow 连接

使用 Apache Airflow 用户界面创建新的 SSH 连接

1. 打开[环境页面](#)在亚马逊 MWA 控制台上。
2. 从环境列表中，选择打开 Airflow 适用于您的环境。
3. 在 Apache Airflow 用户界面页面上，选择管理员从顶部导航栏展开下拉列表，然后选择连接。
4. 在列出连接页面，选择+，或添加新记录按钮添加新连接。
5. 在添加连接页面，添加以下信息：

- a. 对于连接编号，ENTER **ssh_new**.
- b. 对于连接类型，选择SSH从下拉列表中。

Note

如果SSH列表中没有连接类型，亚马逊 MWAA 尚未安装所需的连接类型 `apache-airflow-providers-ssh` 程序包。更新您的 `requirements.txt` 文件以包括此程序包，然后重试。

- c. 对于Host，输入您要连接的 Amazon EC2 实例的 IP 地址。例如，**12.345.67.89**。
- d. 对于用户名，ENTER **ec2-user** 如果您连接到 Amazon EC2 实例。您的用户名可能会有所不同，具体取决于您希望 Apache Airflow 连接到的远程实例的类型。
- e. 对于额外的，以 JSON 格式输入以下键/值对：

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

这个密钥值对指示 Apache Airflow 在本地寻找密钥/dags目录。

代码示例

以下 DAG 使用 `SSHOperator` 连接您的目标 Amazon EC2 实例，然后运行 `hostname` 用于打印实例名称的 Linux 命令。您可以修改 DAG 以在远程实例上运行任何命令或脚本。

1. 打开终端，然后导航到存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为 `ssh.py`。

```
from airflow.decorators import dag
from datetime import datetime
from airflow.providers.ssh.operators.ssh import SSHOperator

@dag(
    dag_id="ssh_operator_example",
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    catchup=False,
)
def ssh_dag():
    task_1=SSHOperator(
        task_id="ssh_task",
        ssh_conn_id='ssh_new',
        command='hostname',
    )

my_ssh_dag = ssh_dag()
```

3. 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，您将在任务日志中看到类似以下内容的输出 `ssh_task` 在里面 `ssh_operator_example` DAG:

```
[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
```

在中使用密钥AWS Secrets Manager用于 Apache Airflow 雪花连接

以下示例调用AWS Secrets Manager在 Amazon Managed Workflows (MWAA) 上获取 Apache Airflow (MWAA) 的密钥。它假设您已完成中的步骤[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。

主题

- [版本 \(p. 177\)](#)
- [先决条件 \(p. 177\)](#)
- [权限 \(p. 177\)](#)
- [要求 \(p. 178\)](#)
- [代码示例 \(p. 178\)](#)
- [接下来做什么？ \(p. 178\)](#)

版本

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#)。

先决条件

要使用此页面上的示例代码，您需要以下内容：

- Secrets Manager 后端作为 Apache Airflow 配置选项，如[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。
- Secrets Manager 中的 Apache Airflow 连接字符串，如所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。

权限

- Secrets Manager 权限，如中所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。

要求

要使用此页面上的示例代码，请将以下依赖项添加到您的`requirements.txt`。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。

```
apache-airflow-providers-snowflake==1.3.0
```

代码示例

以下步骤介绍如何创建调用 Secrets Manager 获取密钥的 DAG 代码。

1. 在命令提示符下，导航到 DAG 代码的存储目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存在`snowflake_connection.py`。

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

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

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

snowflake_query = [
    """use warehouse "MY_WAREHOUSE";""",
    """select * from "SNOWFLAKE_SAMPLE_DATA"."WEATHER"."WEATHER_14_TOTAL" limit
100;""",
]

with DAG(dag_id='snowflake_test', schedule_interval=None, catchup=False,
start_date=days_ago(1)) as dag:
    snowflake_select = SnowflakeOperator(
        task_id="snowflake_select",
        sql=snowflake_query,
        snowflake_conn_id="snowflake_conn",
    )
```

接下来做什么？

- 学习如何将此示例中的 DAG 代码上传到dagsAmazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#)。

使用 DAG 写入自定义指标 CloudWatch

您可以使用以下代码示例编写运行的有向无环图 (DAG) PythonOperator 检索 Amazon MWAA 环境的操作系统级指标。然后，DAG 将数据作为自定义指标发布到亚马逊 CloudWatch。

自定义操作系统级别指标可让您进一步了解您的环境工作人员如何利用虚拟内存和 CPU 等资源。您可以使用此信息来选择[环境类 \(p. 85\)](#)最适合你的工作量。

主题

- [版本 \(p. 179\)](#)
- [先决条件 \(p. 179\)](#)
- [权限 \(p. 179\)](#)
- [附属物 \(p. 179\)](#)
- [代码示例 \(p. 179\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在[Python 3.7](#)。

先决条件

要使用此页面上的代码示例，您需要以下内容：

- 网络 ACL 和安全组都允许（因此可到达您的实例）的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#)。

权限

- 无需额外权限即可使用此页面上的代码示例。

附属物

- 无需额外的依赖关系即可使用此页面上的代码示例。

代码示例

1. 在命令提示符下，导航到存储 DAG 代码的文件夹。例如：

```
cd dags
```

2. 复制以下代码示例的内容，并且使用本地将其保存为 dag-custom-metrics.py。Replace（替换）MWAA-ENV-NAME 使用您的环境名称。

```
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. 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果 DAG 成功运行，您应该会在 Apache Airflow 日志中看到类似以下内容的内容：

```
[2022-08-16, 10:54:46 UTC] [{logging_mixin.py:109}] INFO - cpu_stats
scputstats(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
```

在亚马逊 MWAA 环境中清理 Aurora PostgreSQL

亚马逊 Apache Airflow 管理工作流程 (MWAA) 使用 Aurora PostgreSQL 数据库作为 Apache Airflow 元数据库，DAG 运行和存储任务实例。以下示例代码会定期清除您的亚马逊 MWAA 环境的专用 Aurora PostgreSQL 数据库中的条目。

主题

- [版本 \(p. 182\)](#)
- [先决条件 \(p. 182\)](#)
- [附属物 \(p. 182\)](#)
- [代码示例 \(p. 182\)](#)

版本

- 此页面上的示例代码可用于Apache Airflow在[Python 3.7](#).
- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#).

附属物

- 要在 Apache Airflow v1 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v1 基础安装](#)在您的环境中。
- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

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, settings

from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.jobs.base_job import BaseJob
from airflow.models import DAG, DagModel, DagRun, ImportError, Log, SlaMiss,
    RenderedTaskInstanceFields, TaskFail, TaskInstance, TaskReschedule, Variable, XCom

from datetime import timedelta
import os

DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
```

```

        'email_on_failure': False,
        'email_on_retry': False,
        'retries': 0,
    }

    DEFAULT_MAX_AGE_IN_DAYS = 31

    OBJECTS_TO_CLEAN = [[BaseJob, BaseJob.latest_heartbeat],
        [DagModel, DagModel.last_parsed_time],
        [DagRun, DagRun.execution_date],
        [ImportError, ImportError.timestamp],
        [Log, Log.dttm],
        [SlaMiss, SlaMiss.execution_date],
        [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
        [TaskFail, TaskFail.execution_date],
        [TaskInstance, TaskInstance.execution_date],
        [TaskReschedule, TaskReschedule.execution_date],
        [XCom, XCom.execution_date],
    ]

    def cleanup_db_fn(**kwargs):
        session = settings.Session()
        print("session: ", str(session))

        oldest_date = days_ago(int(Variable.get("max_metadb_storage_days",
            default_var=DEFAULT_MAX_AGE_IN_DAYS)))
        print("oldest_date: ", oldest_date)

        for x in OBJECTS_TO_CLEAN:
            query = session.query(x[0]).filter(x[1] <= oldest_date)
            print(str(x[0]), ": ", str(query.all()))
            query.delete(synchronize_session=False)

        session.commit()

        return "OK"

    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='@daily',
        tags=['db']
    ) as dag:

        cleanup_db = PythonOperator(
            task_id="cleanup_db",
            python_callable=cleanup_db_fn,
            provide_context=True
        )

```

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, settings

from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from airflow.jobs import BaseJob
from airflow.models import DAG, DagModel, DagRun, ImportError, Log, SlaMiss,
    RenderedTaskInstanceFields, TaskFail, TaskInstance, TaskReschedule, Variable, XCom

from datetime import timedelta
import os

DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 0,
}

DEFAULT_MAX_AGE_IN_DAYS = 31

OBJECTS_TO_CLEAN = [[BaseJob, BaseJob.latest_heartbeat],
    [DagModel, DagModel.last_scheduler_run],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [TaskFail, TaskFail.execution_date],
    [TaskInstance, TaskInstance.execution_date],
    [TaskReschedule, TaskReschedule.execution_date],
    [XCom, XCom.execution_date],
]

def cleanup_db_fn(**kwargs):
    session = settings.Session()
    print("session: ", str(session))

    oldest_date = days_ago(int(Variable.get("max_metadb_storage_days",
    default_var=DEFAULT_MAX_AGE_IN_DAYS)))
    print("oldest_date: ", oldest_date)

    for x in OBJECTS_TO_CLEAN:
        query = session.query(x[0]).filter(x[1] <= oldest_date)
        print(str(x[0]), ": ", str(query.all()))
        query.delete(synchronize_session=False)

    session.commit()

    return "OK"

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='@daily',
    tags=['db']
) as dag:
```

```
cleanup_db = PythonOperator(  
    task_id="cleanup_db",  
    python_callable=cleanup_db_fn,  
    provide_context=True  
)
```

将环境元数据导出到 Amazon S3 上的 CSV 文件

以下代码示例显示如何创建定向非循环图 (DAG)，该图在数据库中查询一系列 DAG 运行信息，并将数据写入 .csv 存储的文件存储在 Amazon S3 上。

您可能需要从环境的 Aurora PostgreSQL 数据库中导出信息，以便在本地检查数据，将它们存档到对象存储中，或者将它们与诸如[Amazon S3 Amazon Redshift S3](#)还有[数据库清除 \(p. 181\)](#)，目的是将 Amazon MWAA 元数据移出环境，但保留它们以备 future 分析。

您可以在数据库中查询中列出的任何对象[Apache Airple](#)。此代码示例使用三个模型，DagRun, TaskFail，以及 TaskInstance，它们提供与 DAG 运行相关的信息。

主题

- [版本 \(p. 185\)](#)
- [先决条件 \(p. 185\)](#)
- [权限 \(p. 185\)](#)
- [要求 \(p. 186\)](#)
- [代码示例 \(p. 186\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在[Python 3.7](#)。

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#)。
- 一个[新的 Amazon S3 存储桶](#)要将元数据信息导出到哪里。

权限

亚马逊 MWAA 需要获得 Amazon S3 操作的许可 s3:PutObject 将查询的元数据信息写入 Amazon S3。将以下策略语句添加到您的环境的执行角色。

```
{  
  "Effect": "Allow",  
  "Action": "s3:PutObject*",  
  "Resource": "arn:aws:s3:::your-new-export-bucket"  
}
```

此策略仅将写入权限限制为 *your-new-export-bucket*。

要求

- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

以下步骤描述了如何创建一个查询 Aurora PostgreSQL 并将结果写入新的 Amazon S3 存储桶的 DAG。

- 在终端中，导航到存储 DAG 代码的目录。例如：

```
cd dags
```

- 复制以下代码示例的内容并将其作为本地保存 metadata_to_csv.py。您可以更改分配的值 MAX_AGE_IN_DAYS 控制您从元数据数据库中查询 DAG 的最旧记录的期限。

```
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. 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，则将在任务日志中输出类似于以下内容的内容export_db任务：

```
[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
```

现在可以访问和下载导出的.csv您的新 Amazon S3 存储桶中的文件/files/export/。

在中使用密钥AWS Secrets Manager对于 Apache 气 流变量

以下示例调用AWS Secrets Manager在亚马逊 Managed Workflow (MWAA) 中获取 Apache Airflow (MWAA)。它假设您已完成中的步骤[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。

主题

- [版本 \(p. 188\)](#)
- [先决条件 \(p. 188\)](#)
- [权限 \(p. 188\)](#)
- [要求 \(p. 188\)](#)
- [代码示例 \(p. 188\)](#)
- [接下来做什么？ \(p. 189\)](#)

版本

- 此页面上的示例代码可用于Apache Airflow在[Python 3.7](#).
- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用本页上的示例代码，您需要以下内容：

- Secrets Manager 后端作为 Apache Airflow 配置选项，如[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#).
- Secrets Manager 中的 Apache Airflow 变量字符串，如所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#).

权限

- Secrets Manager 权限，如中所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#).

要求

- 要在 Apache Airflow v1 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v1 基础安装](#)在您的环境中。
- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

以下步骤介绍如何创建调用 Secrets Manager 获取密钥的 DAG 代码。

1. 在命令提示符下，导航到 DAG 代码存储目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存在secrets-manager-var.py.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.models import Variable
from airflow.utils.dates import days_ago
from datetime import timedelta
import os
DAG_ID = os.path.basename(__file__).replace(".py", "")
DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}
```



```
def get_variable_fn(**kwargs):
    my_variable_name = Variable.get("test-variable", default_var="undefined")
    print("my_variable_name: ", my_variable_name)
    return my_variable_name
with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['variable']
) as dag:
    get_variable = PythonOperator(
        task_id="get_variable",
        python_callable=get_variable_fn,
        provide_context=True
    )
```

接下来做什么？

- 学习如何将此示例中的 DAG 代码上传到 Amazon S3 存储桶中的文件夹 [添加或更新 DAG \(p. 103\)](#).

在中使用密钥AWS Secrets Manager用于 Apache 气流连接

以下示例调用 AWS Secrets Manager 在亚马逊 Apache Airflows (MWAA) 上获取 Apache Airflows (MWAA) 上连接的密钥。它假设您已完成中的步骤 [使用 AWS Secrets Manager 密钥配置 Apache Airflow 连接 \(p. 79\)](#).

主题

- [版本 \(p. 189\)](#)
- [先决条件 \(p. 189\)](#)
- [权限 \(p. 190\)](#)
- [要求 \(p. 188\)](#)
- [代码示例 \(p. 190\)](#)
- [接下来做什么？ \(p. 192\)](#)

版本

- 此页面上的示例代码可用于 Apache Airflows 在 [Python 3.7](#).
- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- Secrets Manager 后端作为 Apache Airflow 配置选项，如 [使用 AWS Secrets Manager 密钥配置 Apache Airflow 连接 \(p. 79\)](#).

- Secrets Manager 中的 Apache Airflow 连接字符串，如所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#).

权限

- Secrets Manager 权限，如中所示[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#).

要求

- 要在 Apache Airflow v1 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v1 基础安装](#)在您的环境中。
- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

以下步骤介绍如何创建调用 Secrets Manager 获取密钥的 DAG 代码。

Apache Airflow v2

1. 在命令提示符下，导航到 DAG 代码存储目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存在secrets-manager.py.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
```

```
'owner': 'airflow',
'start_date': days_ago(1),
'depends_on_past': False
}

### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
    dag_id=os.path.basename(__file__).replace(".py", ""),
    default_args=default_args,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Apache Airflow v1

1. 在命令提示符下，导航到 DAG 代码存储目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存在secrets-manager.py。

```
from airflow import DAG, settings, secrets
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from airflow.contrib.hooks.aws_hook import AwsHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}

### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsHook()
    client = hook.get_client_type('secretsmanager')
```

```
response = client.get_secret_value(SecretId=sm_secretId_name)
myConnSecretString = response["SecretString"]

return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
    dag_id=os.path.basename(__file__).replace(".py", ""),
    default_args=default_args,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

接下来做什么？

- 学习如何将此示例中的 DAG 代码上传到dags您的 Amazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#).

使用 Oracle 创建自定义插件

以下示例指导您完成使用 Oracle for Amazon MWAA 创建自定义插件的步骤，该插件可以与 plugins.zip 文件中的其他自定义插件和二进制文件结合使用。

目录

- [版本 \(p. 192\)](#)
- [先决条件 \(p. 193\)](#)
- [权限 \(p. 193\)](#)
- [要求 \(p. 193\)](#)
- [代码示例 \(p. 193\)](#)
- [创建自定义插件 \(p. 194\)](#)
 - [下载依赖项 \(p. 194\)](#)
 - [自定义插件 \(p. 195\)](#)
 - [Plugins.zip \(p. 195\)](#)
- [气流配置选项 \(p. 196\)](#)
- [接下来做什么？ \(p. 196\)](#)

版本

- 此页面上的示例代码可用于Apache Airflow在[Python 3.7](#).
- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#).
- 工作线程在任何日志级别启用日志，CRITICAL或以上，适用于您的环境。有关 Amazon MWAA 日志类型以及如何管理日志组的更多信息，请参阅[the section called “查看气流日志” \(p. 257\)](#)

权限

- 无需额外权限即可使用此页面上的代码示例。

要求

要使用此页面上的示例代码，请将以下依赖项添加到您的requirements.txt。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。

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
```

代码示例

以下步骤介绍了如何创建 DAG 代码以测试自定义插件。

1. 在命令提示符下，导航到 DAG 代码存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存在oracle.py.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
import os
import cx_Oracle

DAG_ID = os.path.basename(__file__).replace(".py", "")

def testHook(**kwargs):
    cx_Oracle.init_oracle_client()
    version = cx_Oracle.clientversion()
    print("cx_Oracle.clientversion",version)
    return version
```

```
with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False, start_date=days_ago(1))
as dag:
    hook_test = PythonOperator(
        task_id="hook_test",
        python_callable=testHook,
        provide_context=True
    )
```

创建自定义插件

本节介绍如何下载依赖关系、创建自定义插件和 plugins.zip。

下载依赖项

亚马逊 MWAA 会将 plugins.zip 的内容提取到 /usr/local/airflow/plugins 在每个 Amazon MWAA 调度器和工作器容器上。这用于向您的环境添加二进制文件。以下步骤介绍了如何汇编自定义插件所需的文件。

提取 Amazon Linux 容器镜像

1. 在命令提示符下，拉取 Amazon Linux 容器镜像，然后在本地运行容器。例如：

```
docker pull amazonlinux
docker run -it amazonlinux:latest /bin/bash
```

你的命令提示符应该调用 bash 命令行。例如：

```
bash-4.2#
```

2. 安装 Linux 原生异步 I/O 工具 (libaio)。

```
yum -y install libaio
```

3. 保持此窗口处于打开状态以执行后续步骤。我们将在本地复制以下文件：lib64/libaio.so.1, lib64/libaio.so.1.0.0, lib64/libaio.so.1.0.1。

下载客户端文件夹

1. 在本地安装解压缩包。例如：

```
sudo yum install unzip
```

2. 创建 oracle_plugin 目录。例如：

```
mkdir oracle_plugin
cd oracle_plugin
```

3. 使用以下 curl 命令下载 [instantclient-basic-linux.x64-18.5.0.0.0dbru.zip](https://download.oracle.com/otn_software/linux/instantclient/185000/instantclient-basic-linux.x64-18.5.0.0.0dbru.zip) 从 [适用于 Linux x86-64 的 Oracle 即时客户端下载](#)。

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. 解压缩 client.zip 文件。例如：

```
unzip *.zip
```

从 Docker 中提取文件

1. 在新的命令提示符中，显示并写下您的 Docker 容器 ID。例如：

```
docker container ls
```

您的命令提示符应返回所有容器及其 ID。例如：

```
debc16fd6970
```

2. 在你的oracle_plugin目录，解压缩lib64/libaio.so.1,lib64/libaio.so.1.0.0,lib64/libaio.so.1.0.1文件到本地instantclient_18_5文件夹。例如：

```
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/
```

自定义插件

Apache Airflow 将在启动时执行插件文件夹中的 Python 文件的内容。这用于设置和修改环境变量。以下步骤介绍了自定义插件的示例代码。

- 复制以下代码示例的内容并将其作为本地保存在env_var_plugin_oracle.py.

```
from airflow.plugins_manager import AirflowPlugin  
import os  
  
os.environ["LD_LIBRARY_PATH"]='/usr/local/airflow/plugins/instantclient_18_5'  
os.environ["DPI_DEBUG_LEVEL"]="64"  
  
class EnvVarPlugin(AirflowPlugin):  
    name = 'env_var_plugin'
```

Plugins.zip

以下步骤显示如何创建plugins.zip. 此示例的内容可以与您的其他插件和二进制文件合并为一个plugins.zipfile。

压缩插件目录的内容

1. 在命令提示符下，导航到oracle_plugin目录。例如：

```
cd oracle_plugin
```

2. zipinstantclient_18_5plugins.zip 中的目录。例如：

```
zip -r ../plugins.zip ./
```

3. 命令提示符中应该会显示以下内容：

```
oracle_plugin$ ls
client.zip instantclient_18_5
```

4. 删除client.zipfile。例如：

```
rm client.zip
```

压缩 env_var_plugin_oracle.py 文件

1. 添加env_var_plugin_oracle.pyfile 的根目录。plugins.zip 例如：

```
zip plugins.zip env_var_plugin_oracle.py
```

2. 您的 plugins.zip 现在应该包含以下内容：

```
env_var_plugin_oracle.py
instantclient_18_5/
```

气流配置选项

如果你使用的是 Apache Airflow v2，请添加core.lazy_load_plugins : False作为 Apache Airflow 配置选项。要了解更多信息，请参阅[使用配置选项在 2 中加载插件 \(p. 91\)](#)。

接下来做什么？

- 了解如何上传requirements.txt此示例中的文件保存在 Amazon S3 存储桶中[安装 Python 的 \(p. 117\)](#)。
- 学习如何将此示例中的 DAG 代码上传到dagsAmazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#)。
- 详细了解如何上传plugins.zip此示例中的文件保存在 Amazon S3 存储桶中[安装自定义插件 \(p. 106\)](#)。

创建生成运行时环境变量的自定义插件

以下示例引导您完成创建自定义插件的步骤，此插件可在运行时在 Amazon Managed Workflows for for (MWAA) 环境。

主题

- [版本 \(p. 196\)](#)
- [先决条件 \(p. 197\)](#)
- [权限 \(p. 197\)](#)
- [要求 \(p. 197\)](#)
- [自定义插件 \(p. 197\)](#)
- [Plugins.zip \(p. 197\)](#)
- [气流配置选项 \(p. 198\)](#)
- [接下来做什么？ \(p. 198\)](#)

版本

- 此页面上的示例代码可用于Apache Airflow在[Python 3.7](#)。

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#).

权限

- 无需额外权限即可使用此页面上的代码示例。

要求

- 要在 Apache Airflow v1 中使用此代码示例，不需要额外的依赖关系。该代码使用 [Apache Airflow v1 基础安装](#) 在您的环境中。

自定义插件

Apache Airflow 将在启动时执行插件文件夹中的 Python 文件的内容。这用于设置和修改环境变量。以下步骤介绍自定义插件的示例代码。

1. 在命令提示符下，导航到存储插件的目录。例如：

```
cd plugins
```

2. 复制以下代码示例的内容并将其作为本地另存为 `env_var_plugin.py` 在此文件夹中。

```
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

以下步骤显示如何创建 `plugins.zip`。此示例的内容可以与其他插件和二进制文件合并为一个 `plugins.zip` 文件。

1. 在命令提示符下，导航到 `hive_plugin` 上一步中的目录。例如：

```
cd plugins
```

2. 将内容压缩到你的 `plugins` 文件夹。

```
zip -r ../plugins.zip ./
```

气流配置选项

如果你使用的是 Apache Airflow v2，请添加 `core.lazy_load_plugins : False` 作为 Apache Airflow 配置选项。要了解更多信息，请参阅 [使用配置选项在 2 中加载插件 \(p. 91\)](#)。

接下来做什么？

- 了解如何上传 `requirements.txt` 在此示例中将文件保存到 Amazon S3 存储桶 [安装 Python 的 \(p. 117\)](#)。
- 学习如何将此示例中的 DAG 代码上传到 `dags` 您的 Amazon S3 存储桶中的文件夹 [添加或更新 DAG \(p. 103\)](#)。
- 详细了解如何上传 `plugins.zip` 在此示例中将文件保存到 Amazon S3 存储桶 [安装自定义插件 \(p. 106\)](#)。

在亚马逊 MWAA 上更改 DAG 的时区

默认情况下，Apache Airflow 在 UTC+0 中安排你的定向无环图 (DAG)。以下步骤显示如何更改 Amazon MWAA 运行 DAG 时使用的时区 [摆锤](#)。或者，本主题演示如何创建自定义插件来更改环境的 Apache Airflow 日志的时区。

主题

- [版本 \(p. 198\)](#)
- [先决条件 \(p. 198\)](#)
- [权限 \(p. 198\)](#)
- [创建一个插件来更改 Airflow 日志中的时区 \(p. 198\)](#)
- [创建目标 `plugins.zip` \(p. 199\)](#)
- [代码示例 \(p. 199\)](#)
- [接下来做什么？ \(p. 200\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#)。

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#)。

权限

- 无需额外权限即可使用此页面上的代码示例。

创建一个插件来更改 Airflow 日志中的时区

Apache Airflow 将在里面运行 Python 文件 `plugins` 启动时的目录。使用以下插件，您可以覆盖执行器的时区，这会修改 Apache Airflow 写入日志的时区。

1. 创建名为的目录plugins获取您的自定义插件，然后导航到该目录。例如：

```
$ mkdir plugins  
$ cd plugins
```

2. 复制以下代码示例的内容并将其作为本地保存为dag-timezone-plugin.py在里面plugins文件夹。

```
import time  
import os  
  
os.environ['TZ'] = 'America/Los_Angeles'  
time.tzset()
```

3. 在plugins目录，创建一个名为的空 Python 文件__init__.py. 你的plugins目录应类似于以下内容：

```
plugins/  
|-- __init__.py  
|-- dag-timezone-plugin.py
```

创建目标plugins.zip

以下步骤显示如何创建plugins.zip. 此示例的内容可以与其他插件和二进制文件合并为一个plugins.zipfile。

1. 在命令提示符下，导航到plugins上一步中的目录。例如：

```
cd plugins
```

2. 将内容压缩到你的plugins目录。

```
zip -r ../plugins.zip ./
```

3. 上传plugins.zip到 S3 存储桶

```
$ aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

代码示例

要更改 DAG 运行的默认时区 (UTC+0)，我们将使用一个名为[摆锤](#)，一个用于处理时区感知日期时间的 Python 库。

1. 在命令提示符下，导航到 DAG 的存储。例如：

```
$ cd dags
```

2. 复制以下示例的内容并另存为tz-aware-dag.py.

```
from airflow import DAG  
from airflow.operators.bash_operator import BashOperator  
from datetime import datetime, timedelta  
# Import the Pendulum library.  
import pendulum
```

```
# Instantiate Pendulum and set your timezone.
local_tz = pendulum.timezone("America/Los_Angeles")

with DAG(
    dag_id = "tz_test",
    schedule_interval="0 12 * * *",
    catchup=False,
    start_date=datetime(2022, 1, 1, tzinfo=local_tz)
) as dag:
    bash_operator_task = BashOperator(
        task_id="tz_aware_task",
        dag=dag,
        bash_command="date"
    )
```

3. 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，则将在任务日志中输出类似于以下内容的内容tz_aware_task在里面tz_testDAG:

```
[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
```

接下来做什么？

- 详细了解如何上传plugins.zip将此示例中的文件保存到 Amazon S3 存储桶[安装自定义插件 \(p. 106\)](#)。

Refresh CodeArtifact 代币

如果您使用的是 CodeArtifact 要安装 Python 依赖项，亚马逊 MWAA 需要一个有效令牌。要允许亚马逊 MWAA 访问 CodeArtifact 运行时的存储库，您的requirements.txt文件必须包含[--extra-index-url](#)使用令牌。为此，您必须设置机制来更新令牌requirements.txt在到期之前定期进行。

以下主题介绍如何创建使用 DAG[get_authorization_token](#) CodeArtifact API 操作每六小时检索一次新代币。您的 DAG 将新令牌存储在.txt文件置于您环境的 Amazon S3 存储桶中。

主题

- [版本 \(p. 201\)](#)
- [先决条件 \(p. 201\)](#)
- [权限 \(p. 201\)](#)
- [要求 \(p. 188\)](#)

- [代码示例 \(p. 202\)](#)
- [接下来做什么？ \(p. 203\)](#)

版本

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用本页上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#).
- 一个[CodeArtifact 知识库](#)在其中存储环境的依赖关系。

权限

要刷新 CodeArtifact 令牌并将结果写入 Amazon S3 Amazon MWAA 在执行角色中必须具有以下权限。

- 这些区域有：codeartifact:GetAuthorizationToken操作允许亚马逊 MWAA 从中检索新令牌 CodeArtifact。以下策略为每个人授予权限 CodeArtifact 您创建的域名。您可以通过修改语句中的资源值并仅指定您希望环境访问的域来进一步限制对域的访问。

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- 这些区域有：sts:GetServiceBearerToken需要采取行动才能调用 CodeArtifact [GetAuthorizationToken](#)API 操作。此操作返回使用包管理器时必须使用的令牌，例如pip和 CodeArtifact。要将包管理器与 CodeArtifact 存储库，您的环境的执行角色必须允许sts:GetServiceBearerToken如下面的政策声明所示。

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

- 这些区域有：s3:PutObject操作允许Amazon MWAA 写入 Amazon S3。亚马逊 MWAA 使用此权限来替换.txt文件包含 CodeArtifact 存储桶中的 URL 包含一个包含新令牌的新文件。

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:PutObject*",
    "s3:List*"
  ],
  "Resource": [
    "arn:aws:s3:::your-mwaa-bucket",
    "arn:aws:s3:::your-mwaa-bucket/*"
  ]
}
```

```
}
```

要求

要使用此页面上的示例代码，请添加以下依赖关系。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。

您需要将以下文件添加到 Amazon S3 存储桶中。

1. 添加codeartifact.txtfile to todagsAmazon S3 存储桶中的文件夹。例如：

```
--extra-index-url https://aws:{auth_token}@your-codeartifact-  
domain-123456789012.d.codeartifact.us-west-2.amazonaws.com/pypi/your_repo/simple/
```

2. 添加requirements.txtAmazon S3 存储桶中的文件。例如：

```
-r /usr/local/airflow/dags/codeartifact.txt
```

代码示例

以下步骤描述了如何创建更新令牌的 DAG 代码。

1. 在命令提示符下，导航到 DAG 代码存储目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存为code_artifact.py. 您可以更改schedule_intervalcron 表达式用于管理令牌的刷新频率。

```
from airflow.decorators import dag, task  
from datetime import datetime, timedelta  
import os  
import boto3  
from io import StringIO  
  
DAG_ID = os.path.basename(__file__).replace(".py", "")  
  
S3_BUCKET = 'your-mwaa-bucket'  
S3_KEY = 'dags/codeartifact.txt'  
MAX_SECONDS = 43200 # 12 hours  
DOMAIN = "your-codeartifact-domain"  
OWNER = "your-account"  
  
@task()  
def update_token():  
    s3 = boto3.client('s3')  
    codeartifact = boto3.client('codeartifact')  
  
    response = codeartifact.get_authorization_token(  
        domain=DOMAIN,  
        domainOwner=OWNER,  
        durationSeconds=MAX_SECONDS  
    )  
  
    obj = s3.get_object(Bucket=S3_BUCKET, Key=S3_KEY)  
    s_in = obj['Body'].read().decode('utf-8')  
  
    req_lines = s_in.splitlines()
```

```
for i in range(0, len(req_lines)):
    if req_lines[i].startswith("--extra-index-url"):
        s_new=req_lines[i][0:req_lines[i].find('aws:')+4] +
response['authorizationToken'] + req_lines[i][req_lines[i].find('@'):]+\r\n"
        print("old:", req_lines[i])
        print("new:", s_new)
        req_lines[i]=s_new
    else:
        req_lines[i]=req_lines[i]+\r\n"

s_out=""
f = StringIO(s_out)
f.writelines(req_lines)

s3.put_object(Bucket=S3_BUCKET, Key=S3_KEY, Body=f.getvalue())

@dag(
    dag_id=DAG_ID,
    schedule_interval="0 */6 * * *", # Refreshes the token every 6 hours
    start_date=datetime(2022, 1, 1),
    dagrun_timeout=timedelta(minutes=60),
    catchup=False,
)
def update_codeartifact_dag():
    t = update_token()

my_update_codeartifact_dag = update_codeartifact_dag()
```

3. 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果成功，则您将在任务日志中输出类似于以下内容的输出update_token任务：

```
[2022-01-01, 12:00:00 PDT] [{logging_mixin.py:109}] INFO - old: --extra-index-url
https://aws:$AUTH_TOKEN@mwa-test-695325144837.d.codeartifact.us-west-2.amazonaws.com/
pypi/mwa-test/simple/
[2022-01-01, 12:00:00 PDT] [{logging_mixin.py:109}] INFO - new: --extra-
index-url https://aws:your-new-token@mwa-test-695325144837.d.codeartifact.us-
west-2.amazonaws.com/pypi/mwa-test/simple/
[2022-01-01, 12:00:00 PDT] [{warnings.py:110}] WARNING - /usr/local/lib/python3.7/site-
packages/watchtower/__init__.py:349: WatchtowerWarning: Received empty message. Empty
messages cannot be sent to CloudWatch Logs
  warnings.warn("Received empty message. Empty messages cannot be sent to CloudWatch
  Logs", WatchtowerWarning)

[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=code-artifact-test, task_id=update_token, 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
```

接下来做什么？

- 了解如何上传requirements.txt文件将文件置于 Amazon S3 存储桶中[安装 Python 的 \(p. 117\)](#).
- 学习如何将此示例中的 DAG 代码上传到dagsAmazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#).
- 详细了解如何上传plugins.zip文件将文件置于 Amazon S3 存储桶中[安装自定义插件 \(p. 106\)](#).

使用 Apache Hive 和 Hadoop 创建自定义插件

亚马逊 MWAA 提取了以下内容的内容plugins.zip到/usr/local/airflow/plugins. 这可以用来向容器添加二进制文件。此外，Apache Airflow 在plugins文件夹启动—使您能够设置和修改环境变量。以下示例将指导您完成在适用于 Apache Airflow 的亚马逊托管工作流程 (MWAA) 环境中使用 Apache Hive 和 Hadoop 创建自定义插件的步骤，该插件可以与其他自定义插件和二进制文件结合使用。

主题

- [版本 \(p. 204\)](#)
- [先决条件 \(p. 204\)](#)
- [权限 \(p. 204\)](#)
- [要求 \(p. 188\)](#)
- [下载依赖项 \(p. 205\)](#)
- [自定义插件 \(p. 205\)](#)
- [Plugins.zip \(p. 206\)](#)
- [代码示例 \(p. 206\)](#)
- [气流配置选项 \(p. 206\)](#)
- [接下来做什么？ \(p. 206\)](#)

版本

- 此页面上的示例代码可用于Apache Airflow在[Python 3.7](#).
- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#).

权限

- 无需额外权限即可使用此页面上的代码示例。

要求

要使用此页面上的示例代码，请将以下依赖项添加到您的requirements.txt. 要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。

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
```

下载依赖项

亚马逊 MWAA 会将 plugins.zip 的内容提取到 `/usr/local/airflow/plugins` 在每个 Amazon MWAA 调度器和工作器容器上。这用于向您的环境添加二进制文件。以下步骤介绍如何组装自定义插件所需的文件。

1. 在命令提示符下，导航到要创建插件的目录。例如：

```
cd plugins
```

2. 下载 [Hadoop](#) 来自的 [镜子](#)，例如：

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

3. 下载 [Hive](#) 来自的 [镜子](#)，例如：

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

4. 创建目录。例如：

```
mkdir hive_plugin
```

5. 提取 Hadoop。

```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. 提取蜂巢。

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

自定义插件

Apache Airflow 将在启动时执行插件文件夹中的 Python 文件的内容。这用于设置和修改环境变量。以下步骤介绍了自定义插件的示例代码。

1. 在命令提示符下，导航到 `hive_plugin` 目录。例如：

```
cd hive_plugin
```

2. 复制以下代码示例的内容并将其作为 `hive_plugin.py` 在 `hive_plugin` 目录。

```
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. 处理以下文本的内容并在本地另存为 `.airflowignore` 在 `hive_plugin` 目录。

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

以下步骤显示如何创建 `plugins.zip`。此示例的内容可以与其他插件和二进制文件合并为一个 `plugins.zip` 文件。

1. 在命令提示符下，导航到 `hive_plugin` 上一步骤中的目录。例如：

```
cd hive_plugin
```

2. 将内容压缩到你的 `plugins` 文件夹。

```
zip -r ../hive_plugin.zip ./
```

代码示例

以下步骤介绍如何创建 DAG 代码。

1. 在命令提示符下，导航到 DAG 代码存储。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为 `hive.py`。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="hive_test_dag", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    hive_test = BashOperator(
        task_id="hive_test",
        bash_command='hive --help'
    )
```

气流配置选项

如果你使用的是 Apache Airflow v2，请添加 `core.lazy_load_plugins : False` 作为 Apache Airflow 配置选项。要了解更多信息，请参阅 [使用配置选项在 2 中加载插件 \(p. 91\)](#)。

接下来做什么？

- 了解如何上传 `requirements.txt` 此示例中的文件存入您的 Amazon S3 存储桶 [安装 Python 的 \(p. 117\)](#)。

- 学习如何将此示例中的 DAG 代码上传到 dags 您的 Amazon S3 存储桶中的文件夹 [添加或更新 DAG \(p. 103\)](#).
- 详细了解如何上传 plugins.zip 此示例中的文件存入您的 Amazon S3 存储桶 [安装自定义插件 \(p. 106\)](#).

为 Apache Airflow 创建自定义插件 PythonVirtualenvOperator

以下示例显示了如何修补 Apache Airflow PythonVirtualenvOperator 在 Amazon Managed Workflows for Apache Airflow (MWAA)

主题

- [版本 \(p. 207\)](#)
- [先决条件 \(p. 207\)](#)
- [权限 \(p. 207\)](#)
- [要求 \(p. 207\)](#)
- [自定义插件示例代码 \(p. 208\)](#)
- [Plugins.zip \(p. 209\)](#)
- [代码示例 \(p. 209\)](#)
- [气流配置选项 \(p. 211\)](#)
- [接下来做什么？ \(p. 211\)](#)

版本

- 此页面上的示例代码可用于 Apache Airflow 在 [Python 3.7](#).
- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#).

权限

- 无需额外权限即可使用此页面上的代码示例。

要求

要使用此页面上的示例代码，请将以下依赖项添加到您的 requirements.txt。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。

```
virtualenv
```

自定义插件示例代码

Apache Airflow 将在启动时执行插件文件夹中的 Python 文件的内容。这个插件会修补内置的 `PythonVirtualenvOperator` 在启动过程中，使其与亚马逊 MWAA 兼容。以下步骤显示了自定义插件的示例代码。

Apache Airflow v2

1. 在命令提示符下，导航到 `plugins` 上面的目录。例如：

```
cd plugins
```

2. 复制以下代码示例的内容并将其作为 `virtual_python_plugin.py`。

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

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

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

from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

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

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

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

Apache Airflow v1

1. 在命令提示符下，导航到 `plugins` 上面的目录。例如：

```
cd plugins
```

2. 复制以下代码示例的内容并将其作为 `virtual_python_plugin.py`。

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

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

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

Plugins.zip

以下步骤显示如何创建plugins.zip。

1. 在命令提示符下，导航到包含以下内容的目录virtual_python_plugin.py以上。例如：

```
cd plugins
```

2. 将内容压缩到你的plugins文件夹。

```
zip plugins.zip virtual_python_plugin.py
```

代码示例

以下步骤介绍如何创建 DAG 代码。

Apache Airflow v2

1. 在命令提示符下，导航到 DAG 代码存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为virtualenv_test.py。

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

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

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

```
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Apache Airflow v1

1. 在命令提示符下，导航到 DAG 代码存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为virtualenv_test.py.

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

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

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

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

气流配置选项

如果你使用的是 Apache Airflow v2，请添加 `core.lazy_load_plugins : False` 作为 Apache Airflow 配置选项。要了解更多信息，请参阅 [使用配置选项在 2 中加载插件 \(p. 91\)](#)。

接下来做什么？

- 了解如何上传 `requirements.txt` 此示例中的文件保存在 Amazon S3 存储桶 [安装 Python 的 \(p. 117\)](#)。
- 学习如何将此示例中的 DAG 代码上传到 `days` Amazon S3 存储桶中的文件夹 [添加或更新 DAG \(p. 103\)](#)。
- 详细了解如何上传 `plugins.zip` 此示例中的文件保存在 Amazon S3 存储桶 [安装自定义插件 \(p. 106\)](#)。

使用 Lambda 函数调用 DAG

以下代码示例使用 [AWS Lambda](#) 函数用于获取 Apache Airflow CLI 令牌并在亚马逊 MWAA 环境中调用定向非循环图 (DAG)。

主题

- [版本 \(p. 211\)](#)
- [先决条件 \(p. 211\)](#)
- [权限 \(p. 211\)](#)
- [附属物 \(p. 212\)](#)
- [代码示例 \(p. 212\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#)。

先决条件

要使用本代码示例，您必须：

- 使用 [公共网络访问模式 \(p. 53\)](#) 为了你 [Amazon MWAA 环境 \(p. 15\)](#)。
- 有一个 [Lambda 函数](#) 使用最新的 Python 运行时。

Note

如果 Lambda 函数和您的 Amazon MWAA 环境位于同一 VPC 中，则可以在私有网络上使用此代码。对于此配置，Lambda 函数的执行角色需要获得调用 Amazon Elastic Compute Cloud (Amazon EC2) 的权限 `CreateNetworkInterfaceAPI` 操作。您可以使用以下方式提供此权限 [AWSLambdaVPCEAccessExecutionRole](#) AWS 托管策略。

权限

要使用此页面上的代码示例，您的 Amazon MWAA 环境的执行角色需要访问权限才能执行 `airflow:CreateCliToken` 操作。您可以使用以下方式提供此权限 `AmazonMWAAAirflowCliAccess` AWS 托管策略：

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

有关更多信息，请参阅 [Apache Airflow CLI 政策：AmazonmAirflowCliAccess \(p. 38\)](#)。

附属物

- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境上。

代码示例

1. 打开 AWS Lambda 控制台，地址：<https://console.aws.amazon.com/lambda/>。
2. 从中选择您的 Lambda 函数函数列表。
3. 在函数页面上，复制以下代码并将以下代码替换为您的资源名称：
 - YOUR_ENVIRONMENT_NAME— 您的 Amazon MWAA 环境的名称。
 - YOUR_DAG_NAME— 要调用的 DAG 的名称。

```
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. 选择 Deploy (部署) 。

5. 选择测试使用 Lambda 控制台调用您的函数。
6. 要验证您的 Lambda 是否成功调用了您的 DAG，请使用 Amazon MWAA 控制台导航到您环境的 Apache Airflow 用户界面，然后执行以下操作：
 - a. 在 DAG 页面上，在 DAG 列表中找到您的新目标 DAG。
 - b. 下面上次运行上次，请检查最新 DAG 运行的时间戳。此时间戳应与的最新时间戳非常匹配 `invoke_dag` 在你的其他环境中。
 - c. 下面近期任务，请检查上次运行是否成功。

在不同的亚马逊 MWAA 环境中调用 DAG

以下代码示例创建了 Apache Airflow CLI 令牌。然后，该代码使用一个 Amazon MWAA 环境中的有向无环图 (DAG) 在另一个 Amazon MWAA 环境中调用 DAG。

主题

- [版本 \(p. 213\)](#)
- [先决条件 \(p. 213\)](#)
- [权限 \(p. 213\)](#)
- [附属物 \(p. 214\)](#)
- [代码示例 \(p. 214\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#)。

先决条件

要使用此页面上的代码示例，您需要以下内容：

- 两个 [Amazon MWAA 环境 \(p. 15\)](#) 和公共网络 Web 服务器访问，包括您当前的环境。
- 上传到 Amazon Simple Storage S3 存储桶的示例 DAG 上传到 DAG 示例 DAG。

权限

要使用此页面上的代码示例，您的环境的执行角色必须具有创建 Apache Airflow CLI 令牌的权限。您可以挂载 AWS 托管策略 `AmazonMWAAAirflowCliAccess` 授予此权限。

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

有关更多信息，请参阅 [Apache Airflow CLI 政策：AmazonmAirflowCliAccess \(p. 38\)](#)。

附属物

- 要在 Apache Airflow v2 中使用此代码示例，不需要额外的依赖关系。该代码使用[Apache Airflow v2 基础安装](#)在您的环境中。

代码示例

以下代码示例假设您在当前环境中使用 DAG 在另一个环境中调用 DAG。

1. 在终端中，导航到存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其保存在本地 `invoke_dag.py`。将以下值替换为您的信息。

- `your-new-environment-name`— 要调用 DAG 的另一个环境的名称。
- `your-target-dag-id`— 您要调用的其他环境中的 DAG 的 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('mwaas')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwaas/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. 运行以下各项AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. 如果 DAG 成功运行，则将在任务日志中显示类似于以下内容的输出 `invoke_dag_task`。

```
[2022-01-01, 12:00:00 PDT] [{python.py:152}] INFO - Done. Returned value was: None
```

```
[2022-01-01, 12:00:00 PDT] [{taskinstance.py:1280}] INFO - Marking task as SUCCESS.  
dag_id=invoke_dag, task_id=invoke_dag_task, execution_date=20220101T120000,  
start_date=20220101T120000, end_date=20220101T120000  
[2022-01-01, 12:00:00 PDT] [{local_task_job.py:154}] INFO - Task exited with return  
code 0  
[2022-01-01, 12:00:00 PDT] [{local_task_job.py:264}] INFO - 0 downstream tasks  
scheduled from follow-on schedule check
```

要验证您的 DAG 是否已成功调用，请导航到新环境的 Apache Airflow UI，然后执行以下操作：

- 在 DAG 页面上，在 DAG 列表中找到您的新目标 DAG。
- 下面上一次运行，请检查最新 DAG 运行的时间戳。此时间戳应与的最新时间戳非常匹配 invoke_dag 在你的其他环境中。
- 下面近期任务，请检查上次运行是否成功。

通过 Amazon MWAA 与 Amazon RDS for Microsoft SQL Server

您可以使用 Amazon Managed Workflows for Apache Airflow 与 Amazon RDS for Microsoft SQL Server。以下示例代码使用适用于 Apache Airflow 的亚马逊托管工作流程 (MWAA) 环境中的 DAG 连接到适用于 Amazon RDS for Microsoft SQL Server 上执行查询。

主题

- [版本 \(p. 215\)](#)
- [先决条件 \(p. 215\)](#)
- [附属物 \(p. 182\)](#)
- [Apache Airflow v2 连接 \(p. 216\)](#)
- [代码示例 \(p. 216\)](#)
- [接下来做什么？ \(p. 218\)](#)

版本

- 此页面上的示例代码可用于 Apache Airflow 在 [Python 3.7](#)。
- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#)。

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#)。
- 亚马逊 MWAA 和 RDS for SQL Server 在同一个亚马逊 VPC/ 中运行
- 亚马逊 MWAA 和服务器的 VPC 安全组配置了以下连接：
 - 端口的入站规则 1433 在亚马逊 MWAA 的安全组中为 Amazon RDS 开放
 - 或者端口的出站规则 1433 从亚马逊 MWAA 打开到 RDS
- 适用于 SQL Server 的 RDS Airflow Connection 的 Apache Airflow Connection 反映了在之前的流程中创建的 Amazon RDS

附属物

要使用本节中的示例代码，请将以下依赖项添加到您的 `requirements.txt`。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)

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 Airwrow v2 连接

如果你在 Apache Airflow v2 中使用连接，请确保 Airflow 连接对象包含以下键值对：

1. Conn ID：mssql_default
2. Conn 类型：Amazon Web Services
3. 主机：YOUR_DB_HOST
4. Schema：
5. 登录：管理员
6. 密码：
7. 端口：1433
8. 额外：

代码示例

1. 在命令提示符下，导航到 DAG 代码存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其作为本地保存为 `sql-server.py`。

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import pymssql
```

```
import logging
import sys
from airflow import DAG
from datetime import datetime
from airflow.operators.mssql_operator import MsSqlOperator
from airflow.operators.python_operator import PythonOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'mssql_conn_example', default_args=default_args, schedule_interval=None)

drop_db = MsSqlOperator(
    task_id="drop_db",
    sql="DROP DATABASE IF EXISTS testdb;",
    mssql_conn_id="mssql_default",
    autocommit=True,
    dag=dag
)

create_db = MsSqlOperator(
    task_id="create_db",
    sql="create database testdb;",
    mssql_conn_id="mssql_default",
    autocommit=True,
    dag=dag
)

create_table = MsSqlOperator(
    task_id="create_table",
    sql="CREATE TABLE testdb.dbo.pet (name VARCHAR(20), owner VARCHAR(20));",
    mssql_conn_id="mssql_default",
    autocommit=True,
    dag=dag
)

insert_into_table = MsSqlOperator(
    task_id="insert_into_table",
    sql="INSERT INTO testdb.dbo.pet VALUES ('Olaf', 'Disney');",
    mssql_conn_id="mssql_default",
    autocommit=True,
    dag=dag
)

def select_pet(**kwargs):
    try:
        conn = pymssql.connect(
            server='sampledb.<xxxxxx>.<region>.rds.amazonaws.com',
            user='admin',
            password='<yoursupersecretpassword>',
            database='testdb'
        )

        # Create a cursor from the connection
        cursor = conn.cursor()
        cursor.execute("SELECT * from testdb.dbo.pet")
        row = cursor.fetchone()

        if row:
            print(row)
    except:
```

```
        logging.error("Error when creating pymssql database connection: %s",
                        sys.exc_info()[0])

select_query = PythonOperator(
    task_id='select_query',
    python_callable=select_pet,
    dag=dag,
)

drop_db >> create_db >> create_table >> insert_into_table >> select_query
```

接下来做什么？

- 了解如何上传requirements.txt此示例中的文件终止在 Amazon S3 存储桶中[安装 Python 的 \(p. 117\)](#).
- 学习如何将此示例中的 DAG 代码上传到dags您的 Amazon S3 存储桶中的文件夹[添加或更新 DAG \(p. 103\)](#).
- 浏览示例脚本及其他[pymssow 模块示例](#).
- 了解有关在特定的 Microsoft SQL 数据库中执行 SQL 代码的更多信息[mssql_operator](#)在Apache Airflow.

将亚马逊 MWAA 与亚马逊 EMR 结合使用

以下代码示例演示了如何使用 Amazon EMR 和 Amazon Managed Workflow Apache Airflow (MWAA) 启用集成。

主题

- [版本 \(p. 218\)](#)
- [代码示例 \(p. 218\)](#)

版本

- 此页面上的示例代码可以用于Apache Airflow v1在[Python 3.7](#).

代码示例

```
"""
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
```

将适用于 Amazon MWAA

以下示例演示了如何将 Amazon Workflow (MWAA) 与 Amazon Workflow (MWAA) 结合使用

主题

- [版本 \(p. 220\)](#)
- [先决条件 \(p. 221\)](#)
- [为 Amazon EC2 创建公钥 \(p. 221\)](#)
- [创建集群 \(p. 221\)](#)
- [创建mwaa命名空间 \(p. 222\)](#)
- [创建角色mwaa命名空间 \(p. 222\)](#)
- [创建并附加 Amazon EKS 集群的 IAM 角色 \(p. 223\)](#)
- [创建 requirements.txt 文件 \(p. 225\)](#)
- [为亚马逊 EKS 创建身份映射 \(p. 225\)](#)
- [创建 kubeconfig \(p. 225\)](#)
- [创建 DAG \(p. 225\)](#)
- [添加 DAG 和kube_config.yaml到Amazon S3 存储桶 \(p. 227\)](#)
- [启用并触发示例 \(p. 227\)](#)

版本

- 此页面上的示例代码可用于Aplow virflow在[Python 3.7](#).

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用本主题中的示例，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#).
- eksctl。要了解更多信息，请参阅[安装 eksctl](#).
- kubectl。要了解更多信息，请参阅[安装和设置 kubectl](#)。在某些情况下，它是使用 eksctl 安装的。
- 在您创建 Amazon MWAA 环境的区域内的 EC2 key pair。要了解更多信息，请参阅[创建或导入key pair](#).

Note

当您使用以下应用程序时：eksctl命令，您可以添加一个--profile指定默认配置文件以外的配置文件。

为 Amazon EC2 创建公钥

使用以下命令可在您的私有key pair 中创建公有密钥。

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

要了解更多信息，请参阅[检索key pair 的公有密钥](#).

创建集群

使用以下命令创建集群。如果您想为集群创建自定义名称或在其他区域创建集群，请替换名称和区域值。您必须在创建 Amazon MWAA 环境的位于同一区域中创建的集群。替换子网的值，以匹配您用于 Amazon MWAA 的 Amazon VPC 网络中的子网。替换的值ssh-public-key以匹配你使用的密钥。您可以使用位于同一区域的 Amazon EC2 中的现有密钥，也可以在创建 Amazon MWAA 环境的同一区域中创建新密钥。

```
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-3333333333333333, subnet-4444444444444444"
```

创建的集群需要一段时间。完成后，您可以使用以下命令验证集群是否已成功创建并配置了 IAM OIDC 提供商：

```
eksctl utils associate-iam-oidc-provider \  
--region us-west-2 \  
--cluster mwaa-eks \  
--approve
```

创建mwaa命名空间

确认成功创建集群后，使用以下命令为 Pod 创建命名空间。

```
kubectl create namespace mwaa
```

创建角色mwaa命名空间

创建命名空间后，在 EKS 上为 Amazon MWAA 用户创建角色和角色绑定，该用户可以在 MWAA 命名空间中运行 Pod。如果您为命名空间使用了不同的名称，请在中替换 mwaa-n *mwaa* 用你用过的名字。

```
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
```

通过运行以下命令确认新角色可以访问Amazon EKS 集群。如果您没有使用，请务必使用正确的名称*mwaa*：

```
kubectl get pods -n mwaa --as mwaa-service
```

您应该看到返回的消息显示：

No resources found in mwaa namespace.

创建并附加 Amazon EKS 集群的 IAM 角色

您必须创建一个 IAM 角色，然后将其绑定到 Amazon EKS (k8s) 集群，这样它才能通过 IAM 进行身份验证。该角色仅用于登录集群，对控制台或 API 调用没有任何权限。

使用中的步骤为 Amazon MWAA 环境创建一个新角色[Amazon MWAA 执行角色 \(p. 44\)](#)。但是，与其创建和附加该主题中描述的策略，不如附加以下策略：

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

创建角色后，编辑您的 MWAA 环境，以将您创建的角色用作环境的执行角色。要更改角色，请编辑要使用的环境。您可以选择以下应用程序中的执行角色Permissions (权限)。

已知问题：

- 存在一个已知问题，即带有子路径的角色 ARN 无法通过 Amazon EKS 进行身份验证。解决方法是手动创建服务角色，而不是使用 Amazon MWAA 自己创建的服务角色。要了解更多信息，请参阅[当路径包含在 aws-auth configmap 中的 ARN 中时，具有路径的角色不起作用](#)
- 如果 Amazon MWAA 服务列表在 IAM 中不可用，则需要选择替代服务策略，例如 Amazon EC2，然后更新角色的信任策略以匹配以下内容：

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

```
]
}
```

要了解更多信息，请参阅[如何对 IAM 角色使用信任策略](#)。

创建 requirements.txt 文件

要使用本节中的示例代码，请确保您已将以下数据库选项之一添加到您的 requirements.txt。要了解更多信息，请参阅[安装 Python 的 \(p. 117\)](#)。

Apache Airflow v2

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Apache Airflow v1

```
awscli
kubernetes==12.0.1
```

为亚马逊 EKS 创建身份映射

使用您在以下命令中创建的角色角色的 ARN 为 Amazon EKS 创建身份映射。更改区域#####到您创建的环境的区域。替换角色的 ARN，最后替换*mwaa-execution-role*与您环境的执行角色。

```
eksctl create iamidentitymapping \
--region your-region \
--cluster mwaa-eks \
--arn arn:aws:iam::11222333444:role/mwaa-execution-role \
--username mwaa-service
```

创建 kubeconfig

使用以下命令创建 kubeconfig：

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

如果你在跑步时使用了特定的配置文件 update-kubeconfig 你需要删除 env: 部分已添加到 kube_config.yaml 文件中，以便它可以在亚马逊 MWAA 上正常运行。为此，请从文件中删除以下内容，然后将其保存：

```
env:
- name: AWS_PROFILE
  value: profile_name
```

创建 DAG

使用以下代码示例创建 Python 文件，例如 mwaa_pod_example.py 对于 DAG。

Apache Airflow v2

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""

from airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
    KubernetesPodOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
    namespace="mwaa",
    image="ubuntu:18.04",
    cmds=["bash"],
    arguments=["-c", "ls"],
    labels={"foo": "bar"},
    name="mwaa-pod-test",
    task_id="pod-task",
    get_logs=True,
    dag=dag,
    is_delete_operator_pod=False,
    config_file=kube_config_path,
    in_cluster=False,
    cluster_context='aws'
)
```

Apache Airflow v1

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
```

```
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from datetime import datetime
from airflow.contrib.operators.kubernetes_pod_operator import KubernetesPodOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
    namespace="mwaa",
    image="ubuntu:18.04",
    cmds=["bash"],
    arguments=["-c", "ls"],
    labels={"foo": "bar"},
    name="mwaa-pod-test",
    task_id="pod-task",
    get_logs=True,
    dag=dag,
    is_delete_operator_pod=False,
    config_file=kube_config_path,
    in_cluster=False,
    cluster_context='aws'
)
```

添加 DAG 和kube_config.yaml到Amazon S3 存储桶

放入你创建的 DAG 和kube_config.yaml文件到适用于 Amazon Service 环境的 Simple Storkvice Stork
您可以使用 Amazon S3 控制台或AWS Command Line Interface.

启用并触发示例

在 Apache Airflow 中，启用该示例，然后将其触发。

成功运行并完成操作后，使用以下命令验证 pod：

```
kubect1 get pods -n mwaa
```

您应该可以看到类似于如下所示的输出内容：

```
NAME READY STATUS RESTARTS AGE
mwaa-pod-test-aa11bb22cc3344445555666677778888 0/1 Completed 0 2m23s
```

然后，您可以使用以下命令验证 pod 的输出。将名称值替换为上一个命令返回的值：

```
kubect1 logs -n mwaa mwaa-pod-test-aa11bb22cc3344445555666677778888
```

使用连接到 Amazon ECSECSOperator

本主题介绍如何使用ECSOperator从 Amazon MWAA 连接到Amazon Elastic Container Service (Amazon ECS) 在以下步骤中，您将为环境的执行角色，添加所需的权限AWS CloudFormation模板用于创建 Amazon ECS Fargate 集群，最后创建并上传连接到新集群的 DAG。

主题

- [版本 \(p. 228\)](#)
- [先决条件 \(p. 228\)](#)
- [权限 \(p. 228\)](#)
- [创建 Amazon ECS 集群 \(p. 229\)](#)
- [代码示例 \(p. 232\)](#)

版本

- 您可以使用此页面上的代码示例Apache Airflow v2 及以上在[Python 3.7](#).

先决条件

要使用此页面上的示例代码，您需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的[Amazon MWAA 环境 \(p. 15\)](#).

权限

- 您的环境的执行角色需要权限才能在 Amazon ECS 中运行任务。你可以附上[Amazonecs_FullAccess](#) AWS-managed policy，也可以创建以下操作并添加执行角色，也可以创建以下操作并添加执行角色。

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



```
}
```

- 除了添加在 Amazon ECS 中运行任务所需的权限外，您还必须修改 CloudWatch 在您的 Amazon MWAA 执行角色中记录策略声明，以允许访问 Amazon ECS 任务日志组，如下所示。Amazon ECS 日志组由创建AWS CloudFormation在中添加模板[the section called “创建 Amazon ECS 集群” \(p. 229\)](#).

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

有关 Amazon MWAA 执行角色，以及 Amazon MWAA 执行角色，请参阅[执行角色 \(p. 44\)](#).

创建 Amazon ECS 集群

使用以下命令AWS CloudFormation模板，您将构建一个 Amazon ECS Fargate 集群以用于您的亚马逊 MWAA 工作流程。有关更多信息，请参阅 [创建任务定义](#)在Amazon Elastic Conta.

- 使用以下代码创建 JSON 文件并将其保存为ecs-mwaa-cfn.json.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This template deploys an ECS Fargate cluster with an Amazon Linux image as a test for MWAA.",
  "Parameters": {
    "VpcId": {
      "Type": "AWS::EC2::VPC::Id",
      "Description": "Select a VPC that allows instances access to ECR, as used with MWAA."
    },
    "SubnetIds": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "Select at two private subnets in your selected VPC, as used with MWAA."
    },
    "SecurityGroups": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>",
      "Description": "Select at least one security group in your selected VPC, as used with MWAA."
    }
  },
  "Resources": {
    "Cluster": {
      "Type": "AWS::ECS::Cluster",
      "Properties": {
        "ClusterName": {
          "Fn::Sub": "${AWS::StackName}-cluster"
        }
      }
    }
  }
}
```

```

    }
  },
  "LogGroup": {
    "Type": "AWS::Logs::LogGroup",
    "Properties": {
      "LogGroupName": {
        "Ref": "AWS::StackName"
      },
      "RetentionInDays": 30
    }
  },
  "ExecutionRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
      ]
    }
  },
  "TaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
      "Family": {
        "Fn::Sub": "${AWS::StackName}-task"
      },
      "Cpu": 2048,
      "Memory": 4096,
      "NetworkMode": "awsvpc",
      "ExecutionRoleArn": {
        "Ref": "ExecutionRole"
      },
      "ContainerDefinitions": [
        {
          "Name": {
            "Fn::Sub": "${AWS::StackName}-container"
          },
          "Image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest",
          "PortMappings": [
            {
              "Protocol": "tcp",
              "ContainerPort": 8080,
              "HostPort": 8080
            }
          ],
          "LogConfiguration": {
            "LogDriver": "awslogs",
            "Options": {
              "awslogs-region": {
                "Ref": "AWS::Region"
              },
              "awslogs-group": {
                "Ref": "LogGroup"
              }
            }
          }
        }
      ]
    }
  }
}

```

```
        "awslogs-stream-prefix": "ecs"
      }
    }
  },
  "RequiresCompatibilities": [
    "FARGATE"
  ]
},
"Service": {
  "Type": "AWS::ECS::Service",
  "Properties": {
    "ServiceName": {
      "Fn::Sub": "${AWS::StackName}-service"
    },
    "Cluster": {
      "Ref": "Cluster"
    },
    "TaskDefinition": {
      "Ref": "TaskDefinition"
    },
    "DesiredCount": 1,
    "LaunchType": "FARGATE",
    "PlatformVersion": "1.3.0",
    "NetworkConfiguration": {
      "AwsvpcConfiguration": {
        "AssignPublicIp": "ENABLED",
        "Subnets": {
          "Ref": "SubnetIds"
        },
        "SecurityGroups": {
          "Ref": "SecurityGroups"
        }
      }
    }
  }
}
}
```

2. 在命令提示符处，使用以下命令AWS CLI命令创建新的堆栈。必须替换这些值SecurityGroups和SubnetIds使用您的 Amazon MWAA 环境的安全组和子网的值。

```
$ 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
```

或者，您可以使用以下 shell 脚本。该脚本使用[get-environment](#) AWS CLI命令，然后相应地创建堆栈。要运行脚本，请执行以下操作。

- a. 复制脚本并将其另存为ecs-stack-helper.sh和你放在同一个目录里AWS CloudFormation模板。

```
#!/bin/bash

joinByString() {
  local separator="$1"
  shift
  local first="$1"
  shift
```

```

    printf "%s" "$first" "${@/#/$separator}"
}

response=$(aws mwa 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. 使用以下命令运行脚本。Replace (替换) environment-name和stack-name并添加您的信息。

```

$ chmod +x ecs-stack-helper.sh
$ ./ecs-stack-helper.bash environment-name stack-name

```

如果成功，将会看到以下输出：AWS CloudFormation堆栈 ID。

```

{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}

```

在之后AWS CloudFormation堆栈已完成，AWS已经配置了您的 Amazon ECS 资源，您可以创建和上传您的 DAG 了。

代码示例

1. 打开命令提示符，然后导航到存储 DAG 代码的目录。例如：

```
cd dags
```

2. 复制以下代码示例的内容并将其保存在本地mwaa-ecs-operator.py，然后将新的 DAG 上载到 Amazon S3。

```

from http import client
from airflow import DAG
from airflow.providers.amazon.aws.operators.ecs import ECSOperator
from airflow.utils.dates import days_ago
import boto3

CLUSTER_NAME="mwaa-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
information.
CONTAINER_NAME="mwaa-ecs-test-container" #Replace value for CONTAINER_NAME with your
information.
LAUNCH_TYPE="FARGATE"

with DAG(
    dag_id = "ecs_fargate_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1)
) as dag:

```

```
client=boto3.client('ecs')
services=client.list_services(cluster=CLUSTER_NAME,launchType=LAUNCH_TYPE)

service=client.describe_services(cluster=CLUSTER_NAME,services=services['serviceArns'])

ecs_operator_task = ECSOperator(
    task_id = "ecs_operator_task",
    dag=dag,
    cluster=CLUSTER_NAME,
    task_definition=service['services'][0]['taskDefinition'],
    launch_type=LAUNCH_TYPE,
    overrides={
        "containerOverrides":[
            {
                "name":CONTAINER_NAME,
                "command":["ls", "-l", "/"],
            },
        ],
    },
    network_configuration=service['services'][0]['networkConfiguration'],
    awslogs_group="mwaa-ecs-zero",
    awslogs_stream_prefix=f"ecs/{CONTAINER_NAME}",
)
```

Note

在示例 DAG 中，对于awslogs_group，您可能需要使用您的 Amazon ECS 任务日志组的名
称修改日志组。示例假设名为的日志组mwaa-ecs-zero. 对于awslogs_stream_prefix，
使用 Amazon ECS 任务日志流前缀。该示例假定使用日志流前缀，ecs.

- 运行以下命令AWS CLI命令将 DAG 复制到环境的存储桶中，然后使用 Apache Airflow UI 触发 DAG。

```
$ aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

- 如果成功，则将在的任务日志中看到类似于以下内容的输
出ecs_operator_task在ecs_fargate_dagDAG:

```
[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-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
```

在亚马逊 MWAA 上使用 dbt

本主题演示如何在亚马逊 MWAA 中使用 dbt 和 Postgres。在以下步骤中，您将所需的依赖项添加到 requirements.txt，并将示例 dbt 项目上传到环境的 Amazon S3 存储桶。然后，您将使用示例 DAG 来验证 Amazon MWAA 是否已安装依赖项，最后使用 BashOperator 来运行 dbt 项目。

主题

- [版本 \(p. 234\)](#)
- [先决条件 \(p. 234\)](#)
- [附属物 \(p. 235\)](#)
- [将项目 dbt 项目上传到 Amazon S3 \(p. 235\)](#)
- [使用 DAG 验证 dbt 依赖项安装 \(p. 236\)](#)
- [使用 DAG 运行 dbt 项目 \(p. 236\)](#)

版本

- 您可以使用此页面上的代码示例 Apache Airflow v2 及以上在 [Python 3.7](#)。

先决条件

在您完成以下步骤之前，您将需要以下内容：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 [Amazon MWAA 环境 \(p. 15\)](#) 使用 Apache Airflow v2.2.2。此示例是使用 v2.2.2 编写和测试的。您可能需要修改示例，以便与其他 Apache Airflow 版本一起使用。
- 一个债务项目示例。要开始在 Amazon MWAA 中使用 dbt，您可以创建一个分叉并克隆 [dbt 入门项目](#) 来自 dbt-Labs GitHub 存储库。

附属物

要将亚马逊 MWAA 与 dbt 一起使用，请将以下依赖项添加到您的 requirements.txt。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)

当您的环境完成更新后，Amazon MWAA 将安装所需的 dbt 库和其他依赖项，例如psycpg2。

Note

Apache Airflow v2.2.2 提供的默认约束文件版本有冲突jsonschema本指南中使用的 dbt 版本不支持。因此，在将 Amazon MWAA 与 dbt 一起使用时，您可以将 Apache Airflow 限制文件下载并修改到您的 Amazon S3 DAGs 文件夹中，然后在您的requirements.txtfile as--constraint /usr/local/airflow/dags/my-updated-constraint.txt，或者省略--constraint从requirements.txt如下所示。

```
json-rpc==1.13.0
minimal-snowplow-tracker==0.0.2
packaging==20.9
networkx==2.6.3
mashumaro==2.5
sqlparse==0.4.2

logbook==1.5.3
agate==1.6.1
dbt-extractor==0.4.0

pyparsing==2.4.7
msgpack==1.0.2
parsedatetime==2.6
pytimeparse==1.1.8
leather==0.3.4
pyyaml==5.4.1

# Airflow constraints are jsonschema==3.2.0
jsonschema==3.1.1
hologram==0.0.14
dbt-core==0.21.1

psycpg2-binary==2.8.6
dbt-postgres==0.21.1
dbt-redshift==0.21.1
```

在以下部分中，您将把您的 dbt 项目目录上传到 Amazon S3，然后运行一个 DAG 来验证 Amazon MWAA 是否已成功安装所需的 dbt 依赖项。

将项目 dbt 项目上传到 Amazon S3

为了能够在您的 Amazon MWAA 环境中使用 dbt 项目，您可以将整个项目目录上传到您的环境的dags文件夹。当环境更新时，Amazon MWAA 会将 dbt 目录下载到本地usr/local/airflow/dags/文件夹。

将 dbt 项目上传到 Amazon S3

1. 导航到克隆 dbt 入门项目的目录。

2. 运行以下Amazon S3AWS CLI命令以递归方式将项目内容复制到您的环境中dags文件夹，使用--recursive参数。该命令创建了一个名为的子目录dbt您可以将其用于所有的 dbt 项目。如果子目录已经存在，则将项目文件复制到现有目录中，并且不创建新目录。该命令还会在中创建一个子目录dbt这个特定入门项目的目录。

```
$ aws s3 cp dbt-starter-project s3://mwa-bucket/dags/dbt/dbt-starter-project --recursive
```

您可以为项目子目录使用不同的名称来组织父目录中的多个 dbt 项目dbt目录。

使用 DAG 验证 dbt 依赖项安装

以下 DAG 使用BashOperator还有一个 bash 命令用于验证 Amazon MWAA 是否成功安装了中指定的 dbt 依赖项requirements.txt。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="/usr/local/airflow/.local/bin/dbt --version"
    )
```

执行以下操作以查看任务日志并验证 dbt 及其依赖项是否已安装。

1. 导航到亚马逊 MWAA 控制台，然后选择打开气流UI从可用环境列表中。
2. 在 Apache Airflow 用户界面上，找到dbt-installation-test从列表中选择 DAG，然后在Last Run列打开上次成功的任务。
3. 使用图：视图，请选择bash_command任务打开任务实例的详细信息。
4. 选择日志打开任务日志，然后验证日志是否成功列出了我们在中指定的 dbt 版本requirements.txt。

使用 DAG 运行 dbt 项目

以下 DAG 使用BashOperator复制您从本地上传到 Amazon S3 的 dbt 项目usr/local/airflow/dags/到可写入的目录/tmp目录，然后运行 dbt 项目。bash 命令假设一个名为 dbt 的初学者项目dbt-starter-project。根据项目目录的名称修改目录名称。

```
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", "")
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="cp -R /usr/local/airflow/dags/dbt /tmp;\n\ncd /tmp/dbt/dbt-starter-project;\n/usr/local/airflow/.local/bin/dbt run --project-dir /tmp/dbt/dbt-starter-project/ --profiles-dir .;\ncat /tmp/dbt_logs/dbt.log"
    )
```


AWS博客和教程

- [与亚马逊 EKS 和亚马逊 MWAA 合作开发 Apache Airflow v2.x](#)

Amazon Managed Workflows for Apache Airflow

本指南介绍了我们在使用 Amazon Managed Workflows for Apache Airflow (MWAA) 时推荐的最佳实践。

主题

- [亚马逊 MWAA 上的 Apache Airflow 的性能调整 \(p. 238\)](#)
- [在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)

亚马逊 MWAA 上的 Apache Airflow 的性能调整

该页面介绍了我们推荐用于调整 Amazon Managed Workflows for Apache Airflow (MWAA) 环境的性能的最佳实践在[亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。

目录

- [添加气流配置选项 \(p. 238\)](#)
- [气流调度器 \(p. 239\)](#)
 - [参数 \(p. 239\)](#)
 - [Limits \(p. 241\)](#)
- [DAG 文件夹 \(p. 241\)](#)
 - [参数 \(p. 242\)](#)
- [DAG 文件 \(p. 243\)](#)
 - [参数 \(p. 243\)](#)
- [任务 \(p. 245\)](#)
 - [参数 \(p. 245\)](#)
- [延长 Amazon 气流 \(p. 248\)](#)

添加气流配置选项

以下过程将指导您完成将 Airflow 配置选项添加到您的环境中的步骤。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 选择下一步。
5. 在 Airflow 配置选项窗格中选择添加自定义配置。
6. 从下拉列表表中选择一个配置并输入一个值，或者键入自定义配置并输入一个值。
7. 为每个要添加的配置选择 Add Custom Option (添加自定义配置)。
8. 选择保存。

要了解更多信息，请参阅 [在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。

气流调度器

Apache Airflow 调度器是 Apache Airflow 的核心组件。调度程序出现问题可能会阻止 DAG 解析 DAG 和调度任务。有关 Apache Airflow 调度程序调整的更多信息，请参阅 [Apache Airflow 文档网站中的微调调度程序性能](#)。

Note

我们建议在更改本节中参数的默认值时谨慎行事。

参数

本节介绍了 Airflow Scheduler 的可用配置选项及其用例。

Apache Airflow v2

气流版本	气流配置选项	默认值	描述	使用案例
v2	celery_sync_parallelism	1	Celery Executor 用于同步任务状态的进程数。	您可以使用此选项通过限制 Celery Executor 使用的进程来防止队列冲突。默认情况下，将值设置为 1 以防止在将任务日志传送到 CloudWatch 日志时出错。将该值设置为 0 意味着使用最大进程数，但在传送任务日志时可能会导致错误。
v2	scheduler_processor_poll_interval	1	在调度程序“循环”中连续处理 DAG 文件之间等待的秒数。	您可以使用此选项在调度程序完成检索 DAG 解析结果、查找和排队任务以及在执行器中执行排队任务后延长调度程序的睡眠时间，从而释放调度程序上的 CPU 使用率。增加此值会消耗 Apache Airflow v2 和 Apache Airflow <code>1.scheduler.max_threads</code> 的环境中的 <code>scheduler.parsing_process</code> 的调度程序线程数。这可能会降低调度程序解析 DAG 的能力，并延长 DAG 出现在 Web 服务器上的时间。

气流版本	气流配置选项	默认值	描述	使用案例
v2	scheduler.max_dagruns_to_create_parallel	10	DagRuns为每个调度程序“循环”创建的最大 DAG 数。	您可以使用此选项通过减少调度程序“循环”的最大数目来释放资源DagRuns用于调度任务。
v2	调度器解析进程	2	调度程序可以 parallel 运行以调度 DAG 的线程数。	您可以使用此选项通过减少调度程序为解析 DAG 而 parallel 运行的进程数量来释放资源。如果 DAG 解析影响任务调度，我们建议将此数字保持在较低水平。您指定的值必须小于环境中的 vCPU 数量。要了解更多信息，请参阅 限制 (p. 241) 。

Apache Airflow v1

气流版本	气流配置选项	默认值	描述	使用案例
v1	celery.sync_parallelism	0	Celery Executor 用于同步任务状态的进程数。	您可以使用此选项通过限制 Celery Executor 使用的进程来防止队列冲突。默认情况下，将值设置为1以防止在将任务日志传送到 CloudWatch 日志时出错。将该值设置为0意味着使用最大进程数，但在传送任务日志时可能会导致错误。
v1	调度器.processor_poll_interval	1	在调度程序“循环”中连续处理 DAG 文件之间等待的秒数。	您可以使用此选项在调度程序完成检索 DAG 解析结果、查找和排队任务以及在执行器中执行排队任务后延长调度程序的睡眠时间，从而释放调度程序上的 CPU 使用率。增加此值会消耗 Apache Airflow v2 和 Apache Airflow

气流版本	气流配置选项	默认值	描述	使用案例
				<code>v1scheduler.max_threads</code> 的环境 中 <code>scheduler.parsing_processes</code> 的调度程序线程数。这可能会降低调度程序解析 DAG 的能力，并延长 DAG 出现在 Web 服务器上的时间。
v1	调度器.max_threads	4	调度程序可以 parallel 运行以调度 DAG 的线程数。	您可以使用此选项通过减少调度程序为解析 DAG 而 parallel 运行的进程数量来释放资源。如果 DAG 解析影响任务调度，我们建议将此数字保持在较低水平。您指定的值必须小于环境中的 vCPU 数量。要了解更多信息，请参阅 限制 (p. 241) 。

Limits

本节介绍在调整调度程序的默认参数时应考虑的限制。

`scheduler.parsing_processing_processeds`

一个环境类允许每个 vCPU 使用两个线程。必须为环境类的调度器保留至少一个线程。如果您发现计划任务出现延迟，则可能需要增加[环境等级 \(p. 85\)](#)。例如，大型环境的调度器有一个 4 vCPU Fargate 容器实例。这意味着可用于其他进程的最大线程总数为 7。也就是说，两个线程“乘以”四个 vCPUs，减去调度程序的一个。您在中 `scheduler.max_threads` 指定的值 `scheduler.parsing_processes` 不得超过环境类的可用线程数（如下所示）：

- `mw1.s mall` — 不得超过其他进程的 1 线程数。剩余的线程是为调度程序保留的。
- `mw1.med ium` — 不得超过其他进程的 3 线程数。剩余的线程是为调度程序保留的。
- `mw1.l arge` — 不得超过其他进程的 7 线程数。剩余的线程是为调度程序保留的。

DAG 文件夹

Apache Airflow S cheduler 会持续扫描您环境中的 DAG 文件夹。任何包含 `plugins.zip` 的文件或包含“airfrow”导入语句的 Python (.py) 文件。然后，将所有生成的 Python DAG 对象放入以 `DagBag` 供调度程序处理该文件，以确定需要安排哪些任务（如果有）。无论文件是否包含任何可行的 DAG 对象，都会对 `Dag` 文件进行解析。

Note

我们建议在更改本节中参数的默认值时谨慎行事。

参数

本节介绍了 DAG 文件夹的可用配置选项及其用例。

Apache Airflow v2

气流版本	气流配置选项	默认值	描述	使用案例
v2	scheduler.dag_dir_list_interval	每秒钟 0.1 秒	应扫描 DAG 文件夹以获取新文件的秒数。	您可以使用此选项通过增加解析 DAG 文件夹的秒数来释放资源。如果您看到解析时间较长，这可能是由于 DAG 文件夹中有大量文件所致，我们建议您增加此值。
v2	scheduler.min_file_process_interval	30 秒	反映调度程序解析 DAG 并更新 DAG 的秒数。	您可以使用此选项通过增加调度程序在解析 DAG 之前等待的秒数来释放资源。例如，如果指定值，则每 30 秒解析 DAG 文件。我们建议将此数字保持在较高水平，以降低环境中的 CPU 使用率。

Apache Airflow v1

气流版本	气流配置选项	默认值	描述	使用案例
v1	scheduler.dag_dir_list_interval	每秒钟 0.1 秒	应扫描 DAG 文件夹以获取新文件的秒数。	您可以使用此选项通过增加解析 DAG 文件夹的秒数来释放资源。如果您看到解析时间较长，这可能是由于 DAG 文件夹中有大量文件所致，我们建议您增加此值。
v1	scheduler.min_file_process_interval	30 秒	反映调度程序解析 DAG 并更新 DAG 的秒数。	您可以使用此选项通过增加调度程序在解析 DAG 之前等待的秒数来释放资源。例如，如果指定值，则每

气流版本	气流配置选项	默认值	描述	使用案例
				30 秒解析 DAG 文件。30我们建议将此数字保持在较高水平，以降低环境中的 CPU 使用率。

DAG 文件

作为 Apache Airflow S cheduler “循环” 的一部分，对单个 DAG 文件进行解析以提取 DAG Python 对象。Airflow 2.0+ 中调度程序解析进程的最大值，文件是同时处理的，并且scheduler.min_file_process_interval必须经过一定的秒数才能再次解析同一个文件。

Note

我们建议在更改本节中参数的默认值时谨慎行事。

参数

本节介绍可用于 Airflow DAG 文件的配置选项及其用例。

Apache Airflow v2

气流版本	气流配置选项	默认值	描述	使用案例
v2	core.dag_file_processor_timeout	50秒	处理 DAG 文件DagFileProcessor时发生超时前等待的秒数。	您可以使用此选项通过增加超时之前所花费DagFileProcessor的时间来释放资源。如果您在 DAG 处理日志中看到超时导致无法加载任何可行的 DAG，我们建议您增加此值。
v2	core.dagbag_import_timeout	30秒	导入 Python 文件之前的秒数超时。	您可以使用此选项通过增加在导入 Python 文件提取 DAG 对象时调度程序超时之前所花费的时间来释放资源。此选项作为调度程序“循环”的一部分进行处理，并且其包含的值必须低于中指定的值core.dag_file_processor_timeout。
v2	core.min_serialized_dag_update_interval	30秒	更新数据库中序列化的 DAG 的最小秒数。	您可以使用此选项通过增加更新数据库中序列化的 DAG 的秒数来

气流版本	气流配置选项	默认值	描述	使用案例
				释放资源。如果您有大量 DAG 或复杂的 DAG，我们建议您增加此值。随着 DAG 的序列化，增加此值会减少调度程序和数据库的负载。
v2	core.min_serialized_dag_fetch_interval		序列化 DAG 已加载到数据库时从数据库中重新提取的秒数 DagBag。	您可以使用此选项通过增加重新获取序列化 DAG 的秒数来释放资源。该值必须高于中指定的值 <code>core.min_serialized_dag_fetch_interval</code> 以降低数据库的“写入”速率。随着 DAG 的序列化，增加此值可减少 Web 服务器和数据库的负载。

Apache Airflow v1

气流版本	气流配置选项	默认值	描述	使用案例
v1	core.dag_file_processor_timeout	50 秒	处理 DAG 文件 DagFileProcessor 发生超时前等待的秒数。	您可以使用此选项通过增加超时之前所花费 DagFileProcessor 的时间来释放资源。如果您在 DAG 处理日志中看到超时导致无法加载任何可行的 DAG，我们建议您增加此值。
v1	core.min_serialized_dag_update_interval	30 秒	更新数据库中序列化的 DAG 的最小秒数。	您可以使用此选项通过增加更新数据库中序列化的 DAG 的秒数来释放资源。如果您有大量 DAG 或复杂的 DAG，我们建议您增加此值。随着 DAG 的序列化，增加此值会减少调度程序和数据库的负载。
v1	core.min_serialized_dag_fetch_interval		序列化 DAG 已加载到数据库时从数	您可以使用此选项通过增加重

气流版本	气流配置选项	默认值	描述	使用案例
			数据库中重新提取的秒数 DagBag。	新获取序列化 DAG 的秒数来释放资源。该值必须高于中指定的值core.min_serialized_dag能降低数据库的“写入”速率。随着 DAG 的序列化，增加此值可减少 Web 服务器和数据库的负载。

任务

Apache Air flow Sched uler 和 Workers 都参与排队和出队任务。调度程序将准备调度的已解析任务从“无”状态变为“已计划”状态。Executor 也在 Fargate 的调度器容器上运行，对这些任务进行排队并将其状态设置为“已排队”。由于 Worker 容器有容量，它将从队列中获取任务并将状态设置为“正在运行”，然后根据任务的成功或失败将其状态更改为“成功”或“失败”。

Note

我们建议在更改本节中参数的默认值时谨慎行事。

参数

本节介绍可用于 Airflow T asks 的配置选项及其用例。

亚马逊 MWAA 替代的默认 Airflow 配置选项以##标记。

Apache Airflow v2

气流版本	气流配置选项	默认值	描述	使用案例
v2	核心。并行性	10000	可以处于“正在运行”状态的任务实例的最大数量。	您可以使用此选项通过增加可以同时运行的任务实例的数量来释放资源。指定的值应是可用工作器数量乘以 Workers 任务密度的“乘以”。我们建议您仅在看到大量任务处于“正在运行”或“排队”状态时才更改此值。
v2	core.dag_并发	10000	每个 DAG 允许同时运行的任务实例的数量。	您可以使用此选项通过增加允许同时运行的任务实例的数量来释放资源。例如，如果您有一百个 DAG 和十个parallel 任

气流版本	气流配置选项	默认值	描述	使用案例
				务，并且您希望所有 DAG 同时运行，则可以将最大并行度计算为可用 Worker 的数量“乘以 Worker 的任务密度”除以 DAG 的数量（例如 100）。 <code>celery.worker_conc</code>
v2	core.execute_tasks_interpreter	truepython	确定 Apache Airflow 是通过分叉父进程还是通过创建新的 Python 进程来执行任务。	设置为 <code>True</code> ，Apache Airflow 会将你对插件所做的更改识别为一个新的 Python 进程，因此是为执行任务而创建的。
v2	celery.worker_concurrency	不适用	Amazon MWAA 取代了 Airflow 基础安装，使该选项能够将 Workers 作为其自动缩放组件的一部分进行扩展。	##### ###
v2	celery.worker_autoscaling	<code>mw1.small-5,0</code> <code>mw1.medium-10,0</code> <code>mw1.large-20,0</code>	Workers 的任务并发性。	您可以使用此选项通过减少 Workers 的 <code>minimum</code> <code>maximum</code> 任务并发来释放资源。中 <code>minimum</code> 指定的值 <code>maximum</code> 必须相同。无论是否有足够的资源来完成配置，工作人员最多可以接受配置的 <code>maximum</code> 并发任务。如果在没有足够资源的情况下安排任务，则任务会立即失败。我们建议为资源密集型任务更改此值，方法是将其值减少到小于默认值，以允许每个任务增加容量。

Apache Airflow v1

气流版本	气流配置选项	默认值	描述	使用案例
v1.0.12	核心。并行性	10000	可以处于“正在运行”状态的任务实例的最大数量。	您可以使用此选项通过增加可以同时运行的任务实例的数量来释放资源。指定的值应是可用工作器数量乘以 Workers 任务密度的“乘以”。我们建议您仅在看到大量任务处于“正在运行”或“排队”状态时才更改此值。
v1	core.dag_并发	10000	每个 DAG 允许同时运行的任务实例的数量。	您可以使用此选项通过增加允许同时运行的任务实例的数量来释放资源。例如，如果您有一百个 DAG 和十个parallel 任务，并且您希望所有 DAG 同时运行，则可以将最大并行度计算为可用 Worker 的数量“乘以 Worker 的任务密度”除以 DAG 的数量（例如 100）。celery.worker_conc
v1	celery.worker_concurn		Amazon MWAA 取代了 Airflow 基础安装，使该选项能够将 Wor k ers 作为其自动缩放组件的一部分进行扩展。	##### ###
v1	celery.worker_autoS 5,5		Workers 的任务并行性。	您可以使用此选项通过减少 Workers 的minimummaximum任 务并发来释放资源。中minimum指定的值maximum必须相同。无论是否有足够的资源来完成配置，工作人员最多可以接受配置的maximum并发任务。如果在没有足够资源的情况下安排任务，则任务

气流版本	气流配置选项	默认值	描述	使用案例
				会立即失败。我们建议为资源密集型任务更改此值，方法是将该值减少到小于默认值，以允许每个任务增加容量。

延长 Amahe 气流

您可以使用 Apache Airflow v2 及更高版本中的智能传感器，通过将多个小型和轻量级传感器的实例整合到一个进程中来腾出任务槽。要了解更多信息，请参阅[智能传感器](#)。

或者，可以将传感器任务分解到空闲的任务槽中，用于其他活动。例如，与其将传感器配置为 600 秒超时并进行一次重试，不如将同一个传感器配置为 60 次，然后重试 10 次。传感器的功能相同；但是，当它重新调度时，其任务时段会腾出来用于其他活动。

在 requirements.txt 中管理 Python 依赖关系

该页面介绍了我们推荐在 Amazon Managed Workflows for Apache Airflow (MWAA) 环境中安装和管理 Python 依赖关系的最佳做法

目录

- [使用亚马逊 MWAA CLI 实用程序测试 DAG \(p. 248\)](#)
- [使用 PyPi.org 需求文件格式安装 Python 依赖项 \(p. 249\)](#)
 - [选项一：来自 Python Package 索引的 Python 依赖关系 \(p. 249\)](#)
 - [选项二：Python 轮子 \(.whl\) \(p. 250\)](#)
 - [在 Amazon S3 存储桶上使用plugins.zip文件 \(p. 251\)](#)
 - [使用托管在 URL 上的 WHL 文件 \(p. 251\)](#)
 - [从 DAG 创建 WHL 文件 \(p. 252\)](#)
 - [选项三：托管在兼容 PyPi/PEP-503 的私有存储库上的 Python 依赖关系 \(p. 252\)](#)
- [在亚马逊 MWAA 控制台上启用日志 \(p. 253\)](#)
- [在日志控制台上查看 CloudWatch 日志 \(p. 253\)](#)
- [在 Apache Airflow 用户界面中查看错误 \(p. 254\)](#)
 - [登录 Apache Airflow \(p. 58\)](#)
- [示例requirements.txt场景 \(p. 254\)](#)

使用亚马逊 MWAA CLI 实用程序测试 DAG

- 命令行界面 (CLI) 实用程序在本地复制 Amazon Managed Workflows for Apache Airflow (MWAA) 环境。
- CLI 在本地构建 Docker 容器映像，类似于亚马逊 MWAA 生产映像。这允许您在部署到 Amazon MWAA 之前运行本地 Apache Airflow 环境来开发和测试 DAG、自定义插件和依赖关系。
- 要运行 CLI，请参阅[aws-mwaa-local-runner](#)上的 GitHub。

使用 PyPi.org 需求文件格式安装 Python 依赖项

以下部分描述了根据 PyPi.org [需求文件格式](#) 安装 Python 依赖项的不同方法。

选项一：来自 Python Package 索引的 Python 依赖关系

下一节介绍如何在 requirements.txt 文件中指定 Python Package [索引中的 Python](#) 依赖关系。

Apache Airflow v2

1. 在本地测试。在创建 requirements.txt 文件之前，以迭代方式添加其他库以找到软件包及其版本的正确组合。要运行亚马逊 MWAA CLI 实用程序，请参阅[aws-mwaa-local-runner](#)上的 GitHub。
2. 查看 Apache Airflow 套餐额外内容。要查看亚马逊 MWAA 上为 Apache Airflow v2 安装的软件包列表，请参阅 GitHub 网站 requirements.txt 上的[亚马逊 MWAA 本地运行器](#)。
3. 添加约束声明。在文件顶部添加 Apache Airflow v2 环境的限制 requirements.txt 文件。Apache Airflow 限制文件指定了 Apache Airflow 发布时可用的提供者版本。

从 Apache Airflow v2.5.1 开始，您的需求文件必须包含 --constraint 声明。如果您不提供限制条件，Amazon MWAA 将为您指定一个限制条件，以确保您的要求中列出的软件包与您的环境兼容。

Note

Apache Airflow v2.4.3 及更高版本使用 Python 版本 3.10，而较早的 Apache Airflow 版本使用 Python

在以下示例中，替换为环境的版本号以及 {Airflow-version}{Python-version} 与您的环境兼容的 Python 版本。

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

如果限制文件确定该 xyz==1.0 软件包与您环境中的其他软件包不兼容，则 pip3 install 将无法阻止将不兼容的库安装到您的环境中。如果任何软件包的安装失败，则可以在日志的相应日志流中查看每个 Apache Airflow 组件（调度程序、工作程序和 Web 服务器）的错误日 CloudWatch 志。有关日志类型的更多信息，请参阅[the section called “查看气流日志” \(p. 257\)](#)。

4. Apache Airflow 添加[软件包的额外内容](#)和版本 (==)。这有助于防止在您的环境中安装名称相同但版本不同的软件包。

```
apache-airflow[package-extra]==2.5.1
```

5. Python 库。在 requirements.txt 文件中添加软件包名称和版本 (==)。这有助于防止 future 自动应用来自 [PyPi.org](#) 的重大更新。

```
library == version
```

Example Boto3 和 psycopg2-binary

提供此示例是为了演示。boto 和 psycopg2-二进制库包含在 Apache Airflow v2 基础安装中，无需在 requirements.txt 文件中指定。

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

如果指定的软件包没有版本，Amazon MWAA 会从 [PyPi.org](https://pypi.org) 安装该软件包的最新版本。此版本可能与您的其他软件包冲突requirements.txt。

Apache Airflow v1

1. 在本地测试。在创建requirements.txt文件之前，以迭代方式添加其他库以找到软件包及其版本的正确组合。要运行亚马逊 MWAA CLI 实用程序，请参阅[aws-mwaa-local-runner](#)上的 GitHub。
2. 查看 Airflow 套餐的额外内容。查看 Apache Airflow v1.10.12 的可用软件包列表，[网址为 https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt](https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt)。
3. 添加约束文件。将 Apache Airflow v1.10.12 的限制文件添加到requirements.txt文件顶部。如果限制文件确定该xyz==1.0软件包与您环境中的其他软件包不兼容，则pip3 install将无法阻止将不兼容的库安装到您的环境中。

```
--constraint "https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt"
```

4. Apache Airflow 1.10.12 软件包。添加 A [irflow 套餐额外内容](#)和 Apache Airflow v1.10.12 版本(==)。这有助于防止在您的环境中安装名称相同但版本不同的软件包。

```
apache-airflow[package]==1.10.12
```

Example 安全外壳 (SSH)

以下示例requirements.txt文件为 Apache Airflow v1.10.12 安装 SSH。

```
apache-airflow[ssh]==1.10.12
```

5. Python 库。在requirements.txt文件中添加软件包名称和版本(==)。这有助于防止future 自动应用来自 [PyPi.org](https://pypi.org) 的重大更新。

```
library == version
```

Example Boto3

以下示例requirements.txt文件为 Apache Airflow v1.10.12 安装了 Boto3 库。

```
boto3 == 1.17.4
```

如果指定的软件包没有版本，Amazon MWAA 会从 [PyPi.org](https://pypi.org) 安装该软件包的最新版本。此版本可能与您的其他软件包冲突requirements.txt。

选项二：Python 轮子 (.whl)

Python 轮子是一种包格式，旨在提供包含已编译工件的库。将轮子包作为在 Amazon MWAA 中安装依赖项的方法有几个好处：

- 安装速度更快 — 将 WHL 文件作为单个 ZIP 复制到容器中，然后在本地安装，无需下载每个文件。
- 减少冲突-您可以提前确定软件包的版本兼容性。因此，无需递归pip地计算出兼容的版本。
- 更强的弹性 — 使用外部托管的库，下游需求可能会发生变化，从而导致 Amazon MWAA 环境中容器之间的版本不兼容。由于不依赖外部来源提供依赖关系，因此无论每个容器何时实例化，其上的每个容器都具有相同的库。

我们建议使用以下方法将 Python 轮子存档 (.whl) 中的 Python 依赖项安装到您的文件中 requirements.txt。

方法

- [在 Amazon S3 存储桶上使用plugins.zip文件 \(p. 251\)](#)
- [使用托管在 URL 上的 WHL 文件 \(p. 251\)](#)
- [从 DAG 创建 WHL 文件 \(p. 252\)](#)

在 Amazon S3 存储桶上使用plugins.zip文件

Apache Airflow 调度程序、工作程序和 Web 服务器 (适用于 Apache Airflow v2.2.2 及更高版本) 在启动期间在您的环境的AWS托管的 Fargate 容器上查找自定义插件，网址为/usr/local/airflow/plugins/*。此过程在亚马逊 MWAA 针对 Python 依赖项和 Apache Airflow pip3 install -r requirements.txt 服务启动之前开始。plugins.zip文件可用于任何您不希望环境执行期间持续更改的文件，或者您可能不想向编写 DAG 的用户授予访问权限的任何文件。例如，Python 库轮子文件、证书 PEM 文件和配置 YAML 文件。

以下部分介绍如何在 Amazon S3 存储桶上安装plugins.zip文件中的轮子。

1. 下载必要的 WHL 文件您可以[pip download](#)使用亚马逊 MWAA [本地运行器](#)或其他[亚马逊 Linux 2](#) 容器 requirements.txt 上的现有文件来解析和下载必要的 Python 轮子文件。

```
$ pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
$ cd "$AIRFLOW_HOME/plugins"
$ zip "$AIRFLOW_HOME/plugins.zip" *
```

2. 在您的路径中指定路径 requirements.txt。使用指定 requirements.txt 顶部的插件目录，[--find-links](#) 并使用指示 pip 不要从其他来源安装 [--no-index](#)，如下所示

```
--find-links /usr/local/airflow/plugins
--no-index
```

Example 在 requirements.txt 中滚动

以下示例假设您已将轮子上传到位于 Amazon S3 存储桶根目录 plugins.zip 的文件中。例如：

```
--find-links /usr/local/airflow/plugins
--no-index

numpy
```

Amazon MWAA 从 plugins 文件夹中提取 numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl 轮子并将其安装在您的环境中。

使用托管在 URL 上的 WHL 文件

以下部分将介绍如何安装托管在 URL 上的轮子。该 URL 必须可以公开访问，或者可以从您为 Amazon MWAA 环境指定的自定义 Amazon VPC 中访问。

- 提供网址。提供您的轮子的 URL requirements.txt。

Example 公共 URL 上的车轮存档

下面的示例从公共站点下载轮子。

```
--find-links https://files.pythonhosted.org/packages/  
--no-index
```

Amazon MWAA 从您指定的网址获取轮子并将其安装在您的环境中。

Note

安装亚马逊 MWAA v2.2.2 及更高版本要求的私有 Web 服务器无法访问 URL。

从 DAG 创建 WHL 文件

如果您有一台使用 Apache Airflow v2.2.2 或更高版本的私有 Web 服务器，并且由于您的环境无法访问外部存储库而无法安装需求，则可以使用以下 DAG 获取现有的 mAmazon MWAA 要求并将其打包到 Amazon S3 上：

```
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}"  
    )
```

运行 DAG 后，将这个新文件用作 Amazon MWAAplugins.zip，也可以与其他插件打包。然后，在--no-index不添加的情况下更新您的requirements.txt前缀--constraint。--find-links /usr/local/airflow/plugins

此方法允许您离线使用相同的库。

选项三：托管在兼容 PyPi /PEP-503 的私有存储库上的 Python 依赖关系

以下部分介绍如何安装托管在带有身份验证的私有 URL 上的 Apache Airflow extra。

1. 将您的用户名和密码添加为 [Apache Airflow 配置选项 \(p. 90\)](#)。例如：

- foo.user: **YOUR_USER_NAME**
- foo.pass: **YOUR_PASSWORD**

2. 创建您的requirements.txt文件。将以下示例中的占位符替换为您的私有 URL，并将您添加为 [Apache Airflow 配置选项 \(p. 90\)](#) 的用户名和密码。例如：

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
```

3. 将任何其他库添加到您的requirements.txt文件中。例如：

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com  
my-private-package==1.2.3
```


在亚马逊 MWAA 控制台上启用日志

您的 Amazon MWAA 环境的[执行角色 \(p. 44\)](#)需要权限才能将日志发送到 CloudWatch 日志。要更新执行角色的权限，请参阅[Amazon MWAA 执行角色 \(p. 44\)](#)。

您可以在INFO、或CRITICAL级别启用 Apache AirWARNING fERROR low 日志。当您选择日志级别时，Amazon MWAA 会发送该级别和所有更高严重级别的日志。例如，如果您在INFO级别启用日志，Amazon MWAA 会将INFO日志WARNING、ERROR、和CRITICAL日志级别发送到 CloudWatch 日志。我们建议在调度程序INFO级别启用 Apache Airflow 日志，以查看收到的日志requirements.txt。

Airflow scheduler logs

Log level

Specify which types of task events to log

INFO	▲
Log info and higher-severity events	
CRITICAL	
Log critical events only	
ERROR	
Log error and higher-severity events	
WARNING	
Log warning and higher-severity events	
INFO	
Log info and higher-severity events	

在日志控制台上查看 CloudWatch 日志

您可以查看 Apache Airflow 日志，以便调度程序安排您的工作流程和解析您的dags文件夹。以下步骤描述了如何在 Amazon MWAA 控制台上打开调度程序的日志组，以及如何在日志控制台上查看 Apache Airflow CloudWatch 日志。

查看日志requirements.txt

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“监视”窗格中选择 Airflow 调度程序日志组。
4. 在requirements_install_ip日志流中选择日志。
5. 您应该在上看到安装在环境中的软件包列表/usr/local/airflow/.local/bin。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgm5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
```

```
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 查看软件包列表以及这些软件包在安装过程中是否遇到任何错误。如果出现问题，您可能会看到类似以下内容的错误：

```
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))
```

在 Apache Airflow 用户界面中查看错误

您可能还需要检查 Apache Airflow 用户界面，以确定错误是否可能与其他问题有关。你在亚马逊 MWAA 上使用 Apache Airflow 时可能遇到的最常见错误是：

```
Broken DAG: No module named x
```

如果您在 Apache Airflow UI 中看到此错误，则说明您的 `requirements.txt` 文件中可能缺少必需的依赖项。

登录 Apache Airflow

你需要在 AWS Identity and Access Management (IAM) 中拥有 AWS 账户的 [Apache Airflow UI 访问政策](#)：[AmazonMwaaWebServerAccess \(p. 38\)](#) 权限才能查看你的 Apache Airflow 用户界面。

访问您的 Apache Airflow

1. 在亚马逊 MWAA 控制台上打开 [环境页面](#)。
2. 选择环境。
3. 选择打开 Airflow UI。

示例 requirements.txt 场景

您可以在自己的格式中混合搭配不同的格式 `requirements.txt`。以下示例组合使用不同的方法来安装额外内容。

Example PyPi.org 上的额外内容和公共网址

除了在公共 URL 上指定软件包（例如符合 PEP 503 的自定义 repo URL）外，您还需要使用该 `--index-url` 选项。PyPi

```
aws-batch == 0.6
phoenix-letter >= 0.3

--index-url http://dist.repoze.org/zope2/2.10/simple
zopelib
```

亚马逊 Managed Workflows for Apache Airflow (MWAA)

监控是保持 Amazon Managed Workflows for Apache Airflow (MWAA) 和您的AWS解决方案的可靠性、可用性和性能的重要方面。我们建议您从AWS解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。本主题描述了AWS用于监控您的 Amazon MWAA 环境和响应潜在事件的资源。

Note

Apache Airflow 指标和日志记录受[亚马逊标准 CloudWatch 定价](#)的约束。

有关监控 Apache Airflow 的更多信息，请参阅 Apache Airflow 文档网站中的[日志记录和监控](#)。

小节目录

- [亚马逊 MWAA 上的监控概述 \(p. 255\)](#)
- [查看审核日志AWS CloudTrail \(p. 256\)](#)
- [在亚马逊上查看气流日志 CloudWatch \(p. 257\)](#)
- [监控亚马逊 MWAA 上的控制面板和警报 \(p. 260\)](#)
- [中的 Apache Airflow v2 环境指标 CloudWatch \(p. 264\)](#)
- [中的 Apache Airflow v1 环境指标 CloudWatch \(p. 270\)](#)
- [亚马逊 MWAA 的容器、队列和数据库指标 \(p. 274\)](#)

亚马逊 MWAA 上的监控概述

该页面描述用于监控 Amazon Managed Workflows for Apache Airflow (MWAA) 环境。AWS

目录

- [亚马逊 CloudWatch 概述 \(p. 255\)](#)
- [AWS CloudTrail 概览 \(p. 255\)](#)

亚马逊 CloudWatch 概述

CloudWatch 是AWS服务的指标存储库，允许您根据服务发布的[指标](#)和[维度](#)检索统计数据。您可以使用这些指标配置[警报](#)、计算统计数据，然后在控制[面板](#)中显示数据，以帮助您在 Amazon CloudWatch 控制台中评估环境的运行状况。

Apache Airflow (MWAA) 环境中的 [StatSD](#) 指标已设置发送 Amazon Managed Workflows for Apache Airflow CloudWatch w (

要了解更多信息，请参阅[什么是亚马逊 CloudWatch ?](#)。

AWS CloudTrail 概览

CloudTrail 是一种审计服务，提供用户、角色或AWS服务在 Amazon MWAA 中所执行的操作的记录。使用收集的信息 CloudTrail，您可以确定向 Amazon MWAA 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及审计日志中提供的其他详细信息。

要了解更多信息，请参阅[什么是AWS CloudTrail](#)。

查看审核日志AWS CloudTrail

AWS CloudTrail在您创建时，将在该AWS账户上启用。CloudTrail 记录 IAM 实体或AWS服务（例如 Amazon 托管工作流程 for Apache Airflow (MWAA)）开展的活动，这些活动被记录为 CloudTrail 事件。您可以在 CloudTrail 控制台中查看、搜索和下载过去 90 天的事件历史记录。CloudTrail 捕获亚马逊 MWAA 控制台上的所有事件以及对亚马逊 MWAA API 的所有调用。它不捕获只读操作GetEnvironment，例如或动PublishMetrics作。本页面介绍了 CloudTrail 如何使用监控Amazon MWAA 的事件。

目录

- [在中创建跟踪 CloudTrail \(p. 256\)](#)
- [使用事件历史记录查看 CloudTrail 事件 \(p. 256\)](#)
- [的示例路线CreateEnvironment \(p. 256\)](#)
- [接下来做什么？ \(p. 257\)](#)

在中创建跟踪 CloudTrail

您需要创建跟踪以查看AWS账户中的持续事件记录，包括 Amazon MWAA 的事件。通过跟踪 CloudTrail，可将日志文件传送到 Amazon S3 存储桶。如果您没有创建跟踪，则仍可在 CloudTrail 控制台中查看可用事件历史记录。例如，使用收集的信息，您可以确定向 Amazon MWAA 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。CloudTrail要了解更多信息，请参阅[为您的AWS账户创建跟踪](#)。

使用事件历史记录查看 CloudTrail 事件

您可以通过查看事件历史记录，对过去 90 天内的 CloudTrail 操作和安全事故进行故障排除。例如，您可以按区域查看与创建、修改或删除AWS账户中的资源（例如，IAM 用户或其他AWS资源）相关的事件。要了解更多信息，请参阅[使用 CloudTrail 事件历史记录查看事件](#)。

1. 打开 [CloudTrail 控制台](#)。
2. 选择“事件历史记录”。
3. 选择要查看的事件，然后选择“比较事件详细信息”。

的示例路线CreateEnvironment

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 桶。

CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作的信息，例如操作的日期和时间或请求参数。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，也不会按任何特定顺序显示。以下示例是由于缺少权限而被拒绝的CreateEnvironment操作的日志条目。为了保护隐私AirflowConfigurationOptions，已对中的值进行了编辑。

```
{
  "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"
}
```

接下来做什么？

- 了解如何在[CloudTrail 支持的AWS服务和集成中的 CloudTrail 日志中收集的事件数据配置其他服务](#)。
- 在[为配置 Amazon SNS 通知中了解如何在将新的日志文件 CloudTrail 发布到 Amazon S3 存储桶时收到通知 CloudTrail](#)。

在亚马逊上查看气流日志 CloudWatch

亚马逊 MWAA 可以向亚马逊发送 Apache Airflow 日志 CloudWatch。您可以从单个位置查看多个环境的日志，无需其他第三方工具即可轻松识别 Apache Airflow 任务延迟或工作流程错误。需要在 Amazon Apache Airflow 托管工作流程 (MWAA) 控制台上启用 Apache Airflow 日志，才能查看 Apache Airflow DAG 处理、任务、Web 服务器、Worker 日志 CloudWatch。

目录

- [定价 \(p. 258\)](#)
- [开始前的准备工作 \(p. 258\)](#)
- [日志类型 \(p. 258\)](#)
- [启用 Apache 气流日志 \(p. 258\)](#)
- [查看 Apache 气流日志 \(p. 259\)](#)
- [调度器日志示例 \(p. 259\)](#)
- [接下来做什么？ \(p. 259\)](#)

定价

- 将收取标准 CloudWatch 日志费用。有关更多信息，请参阅[CloudWatch 定价](#)。

开始前的准备工作

- 您必须拥有可以查看登录的角色 CloudWatch。有关更多信息，请参阅[访问Amazon MWAA 环境 \(p. 32\)](#)：

日志类型

Amazon MWAA 为您启用的每个 Airflow 日志记录选项创建一个日志组，并将 CloudWatch 日志推送到与环境相关的日志组。日志组按以下格式命名：**YourEnvironmentName-LogType**。例如，如果您的环境名为 **Airflow-v202-Public**，则 Apache Airflow 任务日志将发送到 **Airflow-v202-Public-Task**。

日志类型	描述
YourEnvironmentName-DAGProcessing	DAG 处理器管理器（调度程序中处理 DAG 文件的部分）的日志。
YourEnvironmentName-Scheduler	Airflow 调度程序生成的日志。
YourEnvironmentName-Task	DAG 生成的任务日志。
YourEnvironmentName-WebServer	Airflow 网页界面生成的日志。
YourEnvironmentName-Worker	作为工作流程和 DAG 执行的一部分生成的日志。

启用 Apache 气流日志

您可以在 INFO、或 CRITICAL 级别启用 Apache Airflow WARNING、ERROR 或 low 日志。当您选择日志级别时，Amazon MWAA 会发送该级别和所有更高严重级别的日志。例如，如果您在 INFO 级别启用日志，Amazon MWAA 会将 INFO、WARNING、ERROR、和 CRITICAL 日志级别发送到 CloudWatch 日志。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 选择编辑。
4. 选择下一步。
5. 选择以下一个或多个日志记录选项：
 - a. 在“监视”窗格中选择 Airflow 调度程序日志组。

- b. 在“监控”窗格中选择 Airflow Web 服务器日志组。
 - c. 在“监控”窗格中选择 Airflow 工作日志组。
 - d. 在“监视”窗格中选择 Airflow DAG 处理日志组。
 - e. 在“监控”窗格中选择 Airflow 任务日志组。
 - f. 在日志级别中选择日志级别。
6. 选择下一步。
 7. 选择保存。

查看 Apache 气流日志

以下部分将介绍如何在 CloudWatch 控制台中查看 Apache Airflow 日志。

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“监视”窗格中选择一个日志组。
4. 在日志流中选择一个登录流。

调度器日志示例

您可以查看 Apache Airflow 日志，以便调度程序安排您的工作流程和解析您的 dags 文件夹。以下步骤描述了如何在 Amazon MWAA 控制台上打开调度程序的日志组，以及如何在日志控制台上查看 Apache Airflow CloudWatch 日志。

查看日志 requirements.txt

1. 在亚马逊 MWAA 控制台上打开[环境页面](#)。
2. 选择环境。
3. 在“监视”窗格中选择 Airflow 调度程序日志组。
4. 在 requirements_install_ip 日志流中选择日志。
5. 您应该在上看到安装在环境中的软件包列表 /usr/local/airflow/.local/bin。例如：

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmbjhsgdf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. 查看软件包列表以及这些软件包在安装过程中是否遇到任何错误。如果出现问题，您可能会看到类似以下内容的错误：

```
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))
```

接下来做什么？

- 在[使用 CloudWatch 警报中了解如何配置 AWS CloudTrail 警报](#)。
- 了解如何在[使用 CloudWatch 控制面板中创建控制 CloudWatch 面板](#)。

监控亚马逊 MWAA 上的控制面板和警报

您可以在 Amazon 中创建自定义控制面板 CloudWatch，为特定指标添加警报，以监控 Apache Airflow (MWAA) 环境的亚马逊托管工作流程。当某个告警位于控制面板上时，如果它处于该ALARM状态，则会变成红色，便于您主动监控 Amazon MWAA 环境的运行状况。

Apache Airflow 公开了多个进程的指标，包括 DAG 进程的数量、DAG 包大小、当前正在运行的任务、任务失败和成功次数。当您创建环境时，Airflow 被配置为自动将 Amazon MWAA 环境的指标发送到 CloudWatch。本页介绍如何为 Amazon MWAA 环境中的 Airflow 指标创建健康状态控制面板。CloudWatch

目录

- [指标 \(p. 260\)](#)
- [告警状态概述 \(p. 260\)](#)
- [自定义仪表板和警报示例 \(p. 260\)](#)
 - [关于这些指标 \(p. 260\)](#)
 - [关于控制面板 \(p. 261\)](#)
 - [使用AWS教程 \(p. 261\)](#)
 - [使用 AWS CloudFormation \(p. 262\)](#)
- [删除指标和仪表板 \(p. 264\)](#)
- [接下来做什么？ \(p. 264\)](#)

指标

您可以为 Apache Airflow 版本的任何可用指标创建自定义仪表板和警报。每个指标都对应于 Airflow 关键绩效指标 (KPI)。要查看指标列表，请参阅：

- [中的 Apache Airflow v2 环境指标 CloudWatch \(p. 264\)](#)
- [中的 Apache Airflow v1 环境指标 CloudWatch \(p. 270\)](#)

告警状态概述

指标告警可能具有以下几种状态：

- OK — 指标或表达式在定义的阈值范围内。
- ALM — 指标或表达式超出定义的阈值。
- INSUFFICIENT_DATA (数据不足) — 告警刚启动，指标不可用，或者指标没有足够的数据以确定警报状态。

自定义仪表板和警报示例

您可以构建自定义监控控制面板，显示您的 Amazon MWAA 环境所选指标的图表。

关于这些指标

以下列表描述了本节中的教程和模板定义在自定义仪表板中创建的每个指标。

- QueuedTasks-处于排队状态的任务数。对应于`executor.queued_tasks`气流指标。
- TasksPending-执行器中待处理的任务数。对应于`scheduler.tasks.pending`气流指标。

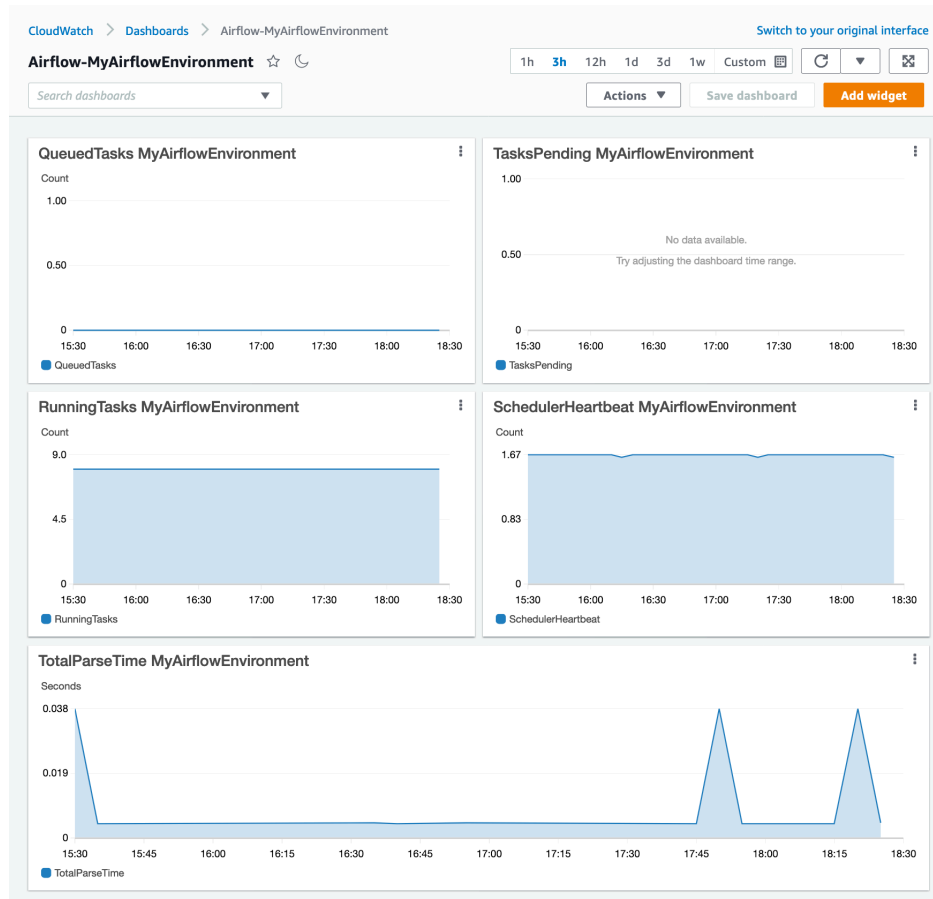
Note

不适用于 Apache Airflow v2.2 及更高版本。

- RunningTasks-执行器中运行的任务数。对应于`executor.running_tasks`气流指标。
- SchedulerHeartbeat-Airflow 在调度程序作业上执行的签入次数。对应于`scheduler_heartbeat`气流指标。
- TotalParseTime-一次扫描和导入所有 DAG 文件所花费的秒数。对应于`dag_processing.total_parse_time`气流指标。

关于控制面板

下图显示了由本节中的教程和模板定义创建的监控仪表板。



使用AWS教程

您可以使用以下AWS教程为当前部署的任何 Amazon MWAA 环境自动创建健康状态控制面板。它还会在所有 Amazon MWAA 环境中为不健康的工作人员和调度程序心跳故障创建 CloudWatch 警报。

- [CloudWatch 亚马逊 MWAA 的控制面板自动化](#)

使用 AWS CloudFormation

您可以使用本节中的AWS CloudFormation模板定义在中创建监控面板，然后在 CloudWatch CloudWatch 控制台上添加警报，以便在指标超过特定阈值时接收通知。要使用此模板定义创建堆栈，请参阅在[AWS CloudFormation控制台上创建堆栈](#)。要向仪表板添加警报，请参阅[使用警报](#)。

```
AWS::CloudFormation::Template
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
}
}
```

删除指标和仪表板

如果您删除 Amazon MWAA 环境，相应的控制面板也会被删除。CloudWatch 指标存储十五 (15) 个月，无法删除。CloudWatch 控制台将指标的搜索限制在上次采集指标后的两 (2) 周内，以确保显示您的 Amazon MWAA 环境的最新实例。要了解更多信息，请参阅[亚马逊 CloudWatch 常见问题解答](#)。

接下来做什么？

- 了解如何创建 DAG 来查询您的环境的 Amazon Aurora PostgreSQL 元数据数据库并将自定义指标发布到 CloudWatch [使用 DAG 写入自定义指标 CloudWatch \(p. 179\)](#)。

中的 Apache Airflow v2 环境指标 CloudWatch

Apache Airflow v2 已经设置为收集适用于 Apache Airflow (MWAA) 的亚马逊托管工作流程 (MWAA) 环境的 [StatSD](#) 指标并将其发送给亚马逊 CloudWatch。Apache Airflow 发送的指标的完整列表可在 Apache Airflow 参考指南的“[指标](#)”页面上找到。本页介绍中提供的 Apache Airflow CloudWatch 指标，以及如何在 CloudWatch 控制台中访问指标。

目录

- [术语 \(p. 264\)](#)
- [维度 \(p. 265\)](#)
- [在 CloudWatch 控制台中访问指标 \(p. 265\)](#)
- [Apache Airflow 指标可用 CloudWatch \(p. 266\)](#)
 - [阿帕奇气流计数器 \(p. 266\)](#)
 - [Apache 气流计 \(p. 267\)](#)
 - [Apache 气流计时器 \(p. 269\)](#)
- [选择要报告哪些指标 \(p. 270\)](#)
- [接下来做什么？ \(p. 270\)](#)

术语

命名空间

命名空间是 AWS 服务 CloudWatch 指标的容器。对于亚马逊 MWAA，命名空间是 AmazonMWAA。

CloudWatch 指标

CloudWatch 指标代表一个按时间顺序排列的数据点集，它们特定于 CloudWatch。
Apache 气流指标

Apache Airflow 的特定[指标](#)。

维度

维度是一个名称/值对，它是指标身份的一部分。

单位

统计数据有一个计量单位。对于 Amazon MWAA，单位包括计数、秒和毫秒。对于亚马逊 MWAA，单位是根据原始 Airflow 指标中的单位设置的。

维度

本节描述了中 Apache Airflow 指标的 CloudWatch 维度分组 CloudWatch。

维度	描述			
天	表示特定的 Apache Airflow DAG 名称。			
DAG 文件名	表示特定的 Apache Airflow DAG 文件名。			
函数	此维度用于改进中的指标分组 CloudWatch。			
任务	表示调度程序正在运行 Apache Airflow Job。总有约 Job 的价值。			
操作符	表示特定的 Apache Airflow 操作员。			
池	表示特定的 Apache Airflow 工作人员池。			
任务	表示特定的 Apache Airflow 任务。			

在 CloudWatch 控制台中访问指标

本节介绍如何在访问特定 DAG CloudWatch 的性能指标。

查看维度的性能指标

1. 在 CloudWatch 控制台上打开 [“指标” 页面](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择 AmazonMWAA 命名空间。
4. 在所有指标选项卡中，选择一个维度。例如，DAG、环境。
5. 为维 CloudWatch 度选择指标。例如，TaskInstanceSuccesses 或 TaskInstanceDuration。选择“绘制所有搜索结果”。

6. 选择“图形化指标”选项卡，查看 Apache Airflow 指标的性能统计信息，例如 DAG、环境、任务。

Apache Airflow 指标可用 CloudWatch

本节描述了发送到的 Apache Airflow 指标和维度 CloudWatch。

阿帕奇气流计数器

本节中的 Apache Airflow 指标包含有关 [Apache Airflow 计数器的数据](#)。

CloudWatch 公制	Apache 气流指标	单位	维度	
SlamisseD Note 适用于 Apache Airflow v2.4.3 及更高版本。	sla_missi	计数	函数、调度器	
slacall回调失败 Note 适用于 Apache Airflow v2.4.3 及更高版本。	sla_callback_通知失败	计数	函数、调度器	
FailedCeleryTaskExecution Note 适用于 Apache Airflow v2.4.3 及更高版本。	celery.execute_command	计数	功能，芹菜	
CriticalSectionBusy	调度程序.critical_section_busy	计数	函数、调度器	
DagBagSize	dagbag_size	计数	函数，DAG 处理	
DagCallbackExceptions	dag 回调异常	计数	DAG，全部	
DSLA 失败EmailAttempts	sla_email_通知_失败	计数	函数、调度器	
TaskInstanceFinished	ti.finish。 {dag_id}。 {task_id}。 {州}	计数	DAG，{dag_id} 任务，{task_id} 州，{州}	
JobEnd	{job_name} _end	计数	作业， {job_name}	
JobHeartbeatFailure	{job_name} _heartbeat_fail	计数	作业， {job_name}	
JobStart	{job_name} _start	计数	作业， {job_name}	

CloudWatch 公制	Apache 气流指标	单位	维度	
ManagerStalls	dag_processing.manager_stall	计数	函数, DAG 处理	
OperatorFailures	operator_failures_{operator_name}	计数	操作员, {operator_name}	
OperatorSuccesses	operator_success_{operator_name}	计数	操作员, {operator_name}	
进程	dag_processing.pid	计数	函数, DAG 处理	
SchedulerHeartbeat	调度器_心跳	计数	函数、调度器	
StartedTaskInstances	ti.start_{dag_id}_{task_id}	计数	DAG, 全部 任务, 全部	
TasksKilledExternally	scheduler.tasks_killed_ex	计数	函数、调度器	
TaskTimeoutError	celery.task_超时_错误	计数	功能, 芹菜	
TaskInstanceCreatedUsingOperator	task_instance_created_{operator_name}	计数	操作员, {operator_name}	
TaskInstancePreviouslySucceeded	以前_成功	计数	DAG, 全部 任务, 全部	
TaskInstanceFailures	ti_failles	计数	DAG, 全部 任务, 全部	
TaskInstanceSuccesses	ti_sucess	计数	DAG, 全部 任务, 全部	
TaskRemovedFromDAG	task_removed_from_dag_{dag_id}	计数	DAG, {dag_id}	
TaskRestoredToDAG	task_restored_to_dag_{dag_id}	计数	DAG, {dag_id}	
ZombiesKilled	僵尸_被击杀	计数	DAG, 全部 任务, 全部	

Apache 气流计

本节中的 Apache Airflow 指标包含有关 [Apache 气流计](#) 的数据。

CloudWatch 公制	Apache 气流指标	单位	维度	
DAGFileRefreshError	dag_file_resh_error	计数	函数, DAG 处理	

CloudWatch 公制	Apache 气流指标	单位	维度	
ImportErrors	dag_processing.import_errors	计数	函数, DAG 处理	
ExceptionFailures	smart_sensor_operator.exception_failures	计数	函数, 智能传感器 操作员	
ExecutedTasks	smart_sensor_operator.executed_tasks	计数	函数, 智能传感器 操作员	
InfraFailures	smart_sensor_operator.infra_failures	计数	函数, 智能传感器 操作员	
LoadedTasks	smart_sensor_operator.loaded_tasks	计数	函数, 智能传感器 操作员	
TotalParseTime	dag_processing.total_parse_time	秒	函数, DAG 处理	
DAGFileProcessingLatency	dag_processing.get_run_secs.ago. {dag_filename}	秒	DAG 文件名, {dag_filename}	
OpenSlots	executor.open_slots	计数	函数, 执行器	
OrphanedTasksAdopted	scheduler.orphaned_tasks. 已通过	计数	函数、调度器	
OrphanedTasksCleared	scheduler.orphaned_tasks. 已清除	计数	函数、调度器	
PokedExceptions	smart_sensor_operator.poked_exc	计数	函数, 智能传感器 操作员	
PokedSuccess	smart_sensor_operator.poked_suc	计数	函数, 智能传感器 操作员	
PokedTasks	smart_sensor_operator.poked_task	计数	函数, 智能传感器 操作员	
PoolFailures	pool.open_slots. {pool_name}	计数	池, {pool_name}	
PoolStarvingTasks	pool.starving_tasks. {pool_name}	计数	池, {pool_name}	
PoolOpenSlots	pool.open_slots. {pool_name}	计数	池, {pool_name}	
PoolQueuedSlots	pool.queued_slots. {pool_name}	计数	池, {pool_name}	
PoolRunningSlots	pool.running_slots. {pool_name}	计数	池, {pool_name}	
ProcessorTimeouts	dag_processing.processor_timeout	计数	函数, DAG 处理	
QueuedTasks	执行者.queued_tasks	计数	函数, 执行器	
RunningTasks	执行者.running_tasks	计数	函数, 执行器	

CloudWatch 公制	Apache 气流指标	单位	维度	
TasksExecutable	调度器.tasks.可执行文件	计数	函数、调度器	
TasksPending Note 不适用于 Apache Airflow v2.2 及更高版本。	调度器.tasks.pendin	计数	函数、调度器	
TasksRunning	调度器.tasks.runnin	计数	函数、调度器	
TasksStarving	调度器.tasks.starving	计数	函数、调度器	
TasksWithoutDagRuns	调度器.tasks.withoutdag	计数	函数、调度器	

Apache 气流计时器

本节中的 Apache Airflow 指标包含有关 [Apache Airflow Timers](#) 的数据。

CloudWatch 公制	Apache 气流指标	单位	维度	
收集 dbdag	collect_db_dags	毫秒	函数，DAG 处理	
CriticalSectionDuration	调度程序.critical_section_durati	毫秒	函数、调度器	
CriticalSectionQueryDuration Note 适用于 Apache Airflow v2.5.1 及更高版本。	调度器.critical_section_query_dury	毫秒	函数、调度器	
DAGDependencyCheck	dagrun.dependency-check。 {dag_id}	毫秒	DAG， {dag_id}	
DAGDurationFailed	dagrun.duration.failed。 {dag_id}	毫秒	DAG， {dag_id}	
DAGDurationSuccessful	dagrun.duration.succes。 {dag_id}	毫秒	DAG， {dag_id}	
DAGFileProcessingLastDuration	dagrun.processing.last_duration {dag_filename}	秒	DAG 文件名， {dag_filename}	

CloudWatch 公制	Apache 气流指标	单位	维度	
DAGScheduleDelay	dagrun.schedule_delay_{dag_id}	毫秒	DAG, {dag_id}	
FirstTaskSchedulingDelay	dagrun.{dag_id}.first_task_scheduling_delay	毫秒	DAG, {dag_id}	
SchedulerLoopDuration Note 适用于 Apache Airflow v2.5.1 及更高版本。	调度器.scheduler_loop_持续时间	毫秒	函数、调度器	
TaskInstanceDuration	天。{dag_id}。{task_id}.持续时间	毫秒	DAG, {dag_id} 任务, {task_id}	

选择要报告哪些指标

您可以使用亚马逊 MWAA [配置选项 \(p. 90\)](#) 选择将 Apache Airflow CloudWatch 指标发送到哪些 Apache Airflow 指标 `metrics.statsd_allow_list`。

例如，应用以下配置选项将仅报告带有前缀的 Apache Airflow 指标，scheduler 以及 executor 来自 [计数器](#)、[仪表](#) 和 [计时器](#) (例如 `scheduler.tasks.executable` 和) 的指标 `executor.running_tasks`：

```
metrics.statsd_allow_list = scheduler,executor
```

接下来做什么？

- 浏览用于发布环境运行状况指标的 Amazon MWAA API 操作，网址为 [PublishMetrics](#)。

中的 Apache Airflow v1 环境指标 CloudWatch

Apache Airflow v1.10.12 已经设置为收集适用于 Apache Airflow (MWAA) 的亚马逊托管工作流程 (MWAA) 环境的 [StatSD](#) 指标并将其发送给亚马逊 CloudWatch。发送的指标的完整列表可在 Apache Airflow 参考指南中的 [Apache Airflow v1.10.12 指标](#) 中找到。本页介绍中提供的 Apache Airflow CloudWatch 指标，以及如何在 CloudWatch 控制台中访问指标。

目录

- [术语 \(p. 271\)](#)
- [维度 \(p. 271\)](#)
- [在 CloudWatch 控制台中访问指标 \(p. 271\)](#)
- [Apache Airflow 指标可用 CloudWatch \(p. 272\)](#)
 - [阿帕奇气流计数器 \(p. 272\)](#)
 - [Apache 气流计 \(p. 273\)](#)
 - [Apache 气流计时器 \(p. 274\)](#)

- [接下来做什么？ \(p. 274\)](#)

术语

命名空间

命名空间是AWS服务 CloudWatch 指标的容器。对于亚马逊 MWAA，命名空间是 AmazonMWAA。
CloudWatch 指标

CloudWatch 指标代表一个按时间顺序排列的数据点集，它们特定于 CloudWatch。
Apache 气流指标

[Apache Airflow v1.10.12 特定于 Apache Airflow 的指标。](#)

维度

维度是一个名称/值对，它是指标身份的一部分。

单位

统计数据有一个计量单位。对于 Amazon MWAA，单位包括计数、秒和毫秒。对于亚马逊 MWAA，单位是根据原始 Airflow 指标中的单位设置的。

维度

本节描述了中 Apache Airflow 指标的 CloudWatch 维度分组 CloudWatch。

维度	描述			
天	表示特定的 Apache Airflow DAG 名称。			
DAG 文件名	表示特定的 Apache Airflow DAG 文件名。			
函数	此维度用于改进中的指标分组 CloudWatch。			
任务	表示调度程序正在运行 Apache Airflow Job。总有约 Job 的价值。			
操作符	表示特定的 Apache Airflow 操作员。			
池	表示特定的 Apache Airflow 工作人员池。			
任务	表示特定的 Apache Airflow 任务。			

在 CloudWatch 控制台中访问指标

本节介绍如何在云访问特定 DAG CloudWatch 的性能指标。

查看维度的性能指标

1. 在 CloudWatch 控制台上打开 [“指标” 页面](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择 AmazonMWAA 命名空间。
4. 在所有指标选项卡中，选择一个维度。例如，DAG、环境。
5. 为维 CloudWatch 度选择指标。例如，TaskInstanceSuccesses 或 TaskInstanceDuration。选择“绘制所有搜索结果”。
6. 选择“图形化指标”选项卡，查看 Apache Airflow 指标的性能统计信息，例如 DAG、环境、任务。

Apache Airflow 指标可用 CloudWatch

本节描述了发送到的 Apache Airflow 指标和维度 CloudWatch。

阿帕奇气流计数器

本节中的 Apache Airflow 指标包含有关 [Apache Airflow 计数器的数据](#)。

CloudWatch 公制	Apache 气流指标	单位	维度	
DagBagSize	dagbag_size	计数	函数，DAG 处理	
JobEnd	{job_name}_end	计数	作业，{job_name}	
JobHeartbeatFailure	{job_name}_heartbeat_fail	计数	作业，{job_name}	
JobStart	{job_name}_start	计数	作业，{job_name}	
OperatorFailures	operator_failures_{operator_name}	计数	操作员，{operator_name}	
OperatorSuccesses	operator_success_{operator_name}	计数	操作员，{operator_name}	
进程	dag_processing_pid	计数	函数，DAG 处理	
SchedulerHeartbeat	调度器_心跳	计数	函数、调度器	
TasksKilledExternally	scheduler.tasks.killed_ex	计数	函数、调度器	
TaskInstanceCreatedUsingOperator	task_instance_created_{operator_name}	计数	操作员，{operator_name}	
TaskInstanceFailures	ti_failures	计数	DAG，全部 任务，全部	
TaskInstanceSuccesses	ti_success	计数	DAG，全部 任务，全部	

CloudWatch 公制	Apache 气流指标	单位	维度	
TaskRemovedFromDAG	task_removed_from_dag。 {dag_id}	计数	DAG , {dag_id}	
TaskRestoredToDAG	task_restored_to_dag。 {dag_id}	计数	DAG , {dag_id}	
ZombiesKilled	僵尸_被击杀	计数	DAG , 全部 任务, 全部	

Apache 气流计

本节中的 Apache Airflow 指标包含有关 [Apache 气流计](#) 的数据。

CloudWatch 公制	Apache 气流指标	单位	维度	
DAGFileRefreshError	dag_file_resch_error	计数	函数, DAG 处理	
ImportErrors	dag_processing.import	计数	函数, DAG 处理	
TotalParseTime	dag_processing.total_parse_time	秒	函数, DAG 处理	
DAGFileProcessingLastRunSecondsAgo	dag_processing.get_run. {dag_filename}	秒.seconds_ago。	DAG 文件名, {dag_filename}	
OpenSlots	ecutor.open_slots	计数	函数, 执行器	
PoolFailures	pool.open_slots。 {pool_name}	计数	池, {pool_name}	
PoolStarvingTasks	pool.starving_tasks。 {pool_name}	计数	池, {pool_name}	
PoolUsedSlots	pool.used_slots。 {pool_name}	计数	池, {pool_name}	
ProcessorTimeouts	dag_processing.processor_超时	计数	函数, DAG 处理	
QueuedTasks	执行者.queued_tasks	计数	函数, 执行器	
RunningTasks	执行者.running_tasks	计数	函数, 执行器	
TasksExecutable	调度器.tasks.可执行文件	计数	函数、调度器	
TasksPending	调度器.tasks.pendin	计数	函数、调度器	
TasksRunning	调度器.tasks.runnin	计数	函数、调度器	
TasksStarving	调度器.tasks.starving	计数	函数、调度器	

CloudWatch 公制	Apache 气流指标	单位	维度	
TasksWithoutDagRunscheduler.tasks.withoutDagRuns	tasks.withoutDagRuns	计数	函数、调度器	

Apache 气流计时器

本节中的 Apache Airflow 指标包含有关 [Apache Airflow Timers](#) 的数据。

CloudWatch 公制	Apache 气流指标	单位	维度	
收集 dbdag	collect_db_dags	毫秒	函数，DAG 处理	
DAGDependencyCheckdagrun.dependency-check. {dag_id}	dagrun.dependency-check. {dag_id}	毫秒	DAG，{dag_id}	
DAGDurationFaileddagrun.duration.failed. {dag_id}	dagrun.duration.failed. {dag_id}	毫秒	DAG，{dag_id}	
DAGDurationSuccessdagrun.duration.success. {dag_id}	dagrun.duration.success. {dag_id}	毫秒	DAG，{dag_id}	
DAGFileProcessingLastDagProcessing.last_dag_processing. {dag_filename}	dagrun.last_dag_processing. {dag_filename}	秒	DAG 文件名，{dag_filename}	
DAGScheduleDelaydagrun.schedule_delay. {dag_id}	dagrun.schedule_delay. {dag_id}	毫秒	DAG，{dag_id}	
TaskInstanceDuration天。 {dag_id}。 {task_id} .持续时间	task_instance_duration. {dag_id}. {task_id}	毫秒	DAG，{dag_id} 任务，{task_id}	

接下来做什么？

- 浏览用于发布环境运行状况指标的 Amazon MWAA API 操作，网址为[PublishMetrics](#)。

亚马逊 MWAA 的容器、队列和数据库指标

除了 Apache Airflow 指标，您还可以使用 CloudWatch Amazon Amazon Amazon Amazon Airflow Workflow (MWAA) 环境的底层组件，该工具可收集原始数据并将数据处理为可读的近乎实时的指标。借助这些环境指标，您可以更好地了解关键性能指标，从而帮助您适当地调整环境规模和调试工作流程问题。这些指标适用于亚马逊 MWAA 上所有支持的 Apache Airflow 版本。

Amazon MWAA 将为每个亚马逊Elastic Container Service (Amazon ECS) 容器和 Amazon Aurora PostgreSQL 实例提供 CPU 和内存利用率，亚马逊简单队列服务 (Amazon SQS) 指标以及 Amazon RDS (用于数据库连接、磁盘队列深度、写入操作、延迟和吞吐量) 的指标，以及 Amazon RDS 代理指标。这些指标还包括基础工作人员、额外工作人员、调度程序和 Web 服务器的数量。

这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解计划失败的原因，并对潜在问题进行故障排除。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 [Amazon CloudWatch 用户指南](#)。

主题

- [术语 \(p. 275\)](#)
- [维度 \(p. 275\)](#)
- [在 CloudWatch 控制台中访问指标 \(p. 275\)](#)
- [指标列表 \(p. 276\)](#)

术语

命名空间

命名空间是AWS服务 CloudWatch 指标的容器。对于亚马逊 MWAA，命名空间是AWS/MWAA。
CloudWatch 指标

CloudWatch 指标代表一个按时间顺序排列的数据点集，它们特定于 CloudWatch。

维度

维度是一个名称/值对，它是指标身份的一部分。

单位

统计数据有一个计量单位。对于亚马逊 MWAA，单位包括数量。

维度

本节描述了中亚马逊 MWAA 指标的 CloudWatch 维度分组 CloudWatch。

维度	描述
集群	亚马逊 MWAA 环境用于运行 Apache Airflow 组件的最少三个 Amazon ECS 容器的指标：调度程序、工作线程和 Web 服务器。
Queue	用于将调度程序与工作程序分离的 Amazon SQS 队列指标。当工作人员阅读消息时，这些消息被视为在飞行中，其他工作人员无法访问这些消息。如果在 12 小时可见性超时之前未删除消息，则这些消息可供其他工作人员阅读。
数据库	衡量亚马逊 MWAA 使用的 Aurora 集群。这包括主数据库实例和支持读取操作的只读副本的指标。亚马逊 MWAA 发布了 READER 和 WRITER 实例的数据库指标。

在 CloudWatch 控制台中访问指标

本部分介绍如何在中访问您的Amazon MWAA 指标 CloudWatch。

查看维度的性能指标

1. 在 CloudWatch 控制台上打开 [“指标” 页面](#)。
2. 使用AWS地区选择器选择您的区域。
3. 选择 AWS/MWAA 命名空间。
4. 在所有指标选项卡中，选择一个维度。例如，集群。

5. 为维 CloudWatch 度选择指标。例如，NumSchedulers或 CPU 利用率。然后，选择绘制所有搜索结果的图表。
6. 选择“图表化指标”选项卡以查看绩效指标。

指标列表

下表列出了 Amazon MWAA 的集群、队列和数据库服务指标。要查看直接从 Amazon ECS、Amazon SQS 或 Amazon RDS 发布的指标的描述，请选择相应的文档链接。

主题

- [集群指标 \(p. 276\)](#)
- [数据库指标 \(p. 276\)](#)
- [Amazon RDS 代理的数据库指标 \(如果可用\) \(p. 277\)](#)
- [队列指标 \(p. 277\)](#)

集群指标

以下指标适用于每个调度程序、基础工作人员、其他工作程序和 Web 服务器。有关每个集群指标的更多信息和描述，请参阅 Amazon ECS 开发人员指南中的[可用指标和维度](#)。

命名空间	指标	Unit
AWS/MWAA	CPUUtilization	百分比
AWS/MWAA	MemoryUtilization	百分比

评估其他工作实例的数量

您可以使用集群维度下提供的组件指标（如以下过程所述）来评估环境在给定时间点使用的额外工作人员。为此，您可以绘制 CPU 利用率或MemoryUtilization指标并将统计类型设置为样本数量。结果值是AdditionalWorker组件的RUNNING任务总数。了解您的环境使用的额外工作实例的数量可以帮助您衡量环境如何auto 扩展，并允许您优化额外工作人员的数量。

1. 选择 AWS/MWAA 命名空间。
2. 在所有指标选项卡中，选择集群维度。
3. 在“集群”维度下 AdditionalWorker，选择 CPU 利用率或MemoryUtilization指标。
4. 在图表化指标选项卡上，将时段设置为 1 分钟，将统计数据设置为样本数。

有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的服务[RUNNING任务数](#)。

数据库指标

以下指标适用于每个数据库实例，直到它被 Amazon RDS 代理取代。有关以下数据库指标的更多信息和描述，请参阅《[Amazon Relational Database Service 用户指南](#)》中的[Amazon RDSCloudWatch 指标](#)。

命名空间	指标	Unit
AWS/MWAA	CPUUtilization	百分比
AWS/MWAA	DatabaseConnections	计数

命名空间	指标	Unit
AWS/MWAA	DiskQueueDepth	计数
AWS/MWAA	FreeableMemory	字节
AWS/MWAA	VolumeWriteIOPS	每五分钟计数
AWS/MWAA	WriteIOPS	每秒计数
AWS/MWAA	WriteLatency	秒
AWS/MWAA	WriteThroughput	每秒字节数

Amazon RDS 代理的数据库指标 (如果可用)

有关以下数据库代理指标的更多信息描述，请参阅《[Amazon Relational Database Service 用户指南](#)》CloudWatch 中的“[使用监控 Amazon RDS 代理指标](#)”。

命名空间	指标	Unit
AWS/MWAA	ClientConnections	计数
AWS/MWAA	ClientConnectionsClosed	计数
AWS/MWAA	ClientConnectionsReceived	计数
AWS/MWAA	AvailabilityPercentage	百分比
AWS/MWAA	DatabaseConnectionsCurrentlyInTransaction	计数
AWS/MWAA	DatabaseConnectionsSetupFailed	计数
AWS/MWAA	DatabaseConnectionsSetupSucceeded	计数
AWS/MWAA	DatabaseConnectionRequests	计数
AWS/MWAA	DatabaseConnections	计数
AWS/MWAA	QueryDatabaseResponseLatency	微秒
AWS/MWAA	QueryRequests	计数
AWS/MWAA	QueryResponseLatency	微秒

队列指标

有关以下队列指标的单位 and 描述的详细信息，请参阅 [Amazon Simple Queue Service 开发人员指南](#) 中的 [Amazon SQS 的可用 CloudWatch 指标](#)。

命名空间	指标	Unit
AWS/MWAA	ApproximateAgeOfOldestMessage	秒
AWS/MWAA	ApproximateNumberOfMessagesNotVisible (正在运行的任务)	计数

命名空间	指标	Unit
AWS/MWAA	ApproximateNumberOfMessagesVisible (任务队列的任务)	计数

Amazon Managed Workflows Apache Airflow Ap

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是AWS和您（客户）的共同责任。[责任共担模型](#)将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 [AWS Compliance Programs](#) 的一部分。要了解适用于 Amazon MWAA 的合规性计划，请参阅[按合规性计划范围内的AWS服务合规性计划](#)范围内的服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Managed Workflows for Apache Airflow (MWAA) 时应用责任共担模式。它说明了如何配置 Amazon MWAA 以实现您的安全性和合规性目标。您还会了解如何使用其他AWS服务以帮助您监控和保护 Amazon MWAA 资源。

本节内容：

- [Amazon Managed Workflows Apache Airflows Apach \(p. 279\)](#)
- [AWS Identity and Access Management \(p. 283\)](#)
- [Amazon Managed Workflows for Apache Airflow \(\(p. 291\)](#)
- [Amazon Managed Workflows Apache Apache Airf \(p. 292\)](#)
- [Amazon MWAA 中的基础设施安全性 \(p. 292\)](#)
- [Amazon MWAA 中的配置和漏洞分析 \(p. 292\)](#)
- [Amazon MWAA 中的安全最佳实践 \(p. 292\)](#)

Amazon Managed Workflows Apache Airflows Apache

AWS[分担责任模型](#)分适用于 Apache Airflow (MWAA) 的 Amazon 托管工作流程中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 [AWS 责任共担模式和 GDPR](#) 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。

- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用控制台、API、或AWS SDK 处理 Amazon MWAA 或其他AWS服务时。AWS CLI 您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

亚马逊 MWAA 上的加密

以下主题介绍了 Amazon MWAA 如何保护您的静态和传输中的数据。使用这些信息了解 Amazon MWAA 如何集成AWS KMS以加密静态数据，以及如何在传输过程中使用传输层安全 (TLS) 协议对数据进行加密。

主题

- [静态加密 \(p. 280\)](#)
- [传输中加密 \(p. 281\)](#)

静态加密

在 Amazon MWAA 上，静态数据是服务保存到永久媒体的数据。

您可以使用[AWS自有密钥](#)进行静态数据加密，也可以选择创建环境时提供[客户管理的密钥](#)进行额外加密。如果您选择使用客户管理的 KMS 密钥，则该密钥必须与您在环境中使用的其他AWS资源和服务位于同一账户中。

要使用客户管理的 KMS 密钥，您必须附上 CloudWatch 访问密钥策略所需的策略声明。当您在环境中使用客户管理的 KMS 密钥时，Amazon MWAA 会代表您附加四项[授权](#)。有关 Amazon MWAA 向客户管理的 KMS 密钥附加的授权的更多信息，请参阅[用于数据加密的客户管理密钥 \(p. 281\)](#)。

如果您不指定客户管理型 KMS 密钥，则默认情况下，Amazon MWAA 会使用AWS拥有的 KMS 密钥来加密和解密您的数据。我们建议使用AWS自有的 KMS 密钥来管理 Amazon MWAA 上的数据加密。

Note

您需要为在 Amazon MWAA 上存储和使用AWS自有或客户管理的 KMS 密钥付费。有关更多信息，请参阅 [AWS KMS 定价](#)。

加密构件

在创建 Amazon MWAA 环境时，您可以通过指定[AWS自有密钥](#)或[客户托管密钥](#)来指定用于静态加密的加密对象。亚马逊 MWAA 将所需的[授权](#)添加到您的指定密钥中。

Amazon S3 — Amazon S3 数据使用服务器端加密 (SSE) 在对象级别加密。Amazon S3 加密和解密过程在存储 DAG 代码和支持文件的 Amazon S3 存储桶上进行。在对象上载到 Amazon S3 时对其进行加密，并在将其下载到 Amazon MWAA 环境时对其进行解密。默认情况下，如果您使用客户管理的 KMS 密钥，Amazon MWAA 会使用它来读取和解密您的 Amazon S3 存储桶上的数据。

CloudWatch 日志 — 如果您使用AWS自有的 KMS 密钥，则发送到日志的 Apache Airflow CloudWatch 日志将使用 LogsAWS 自有 KMS 密钥使用服务器端加密 (SSE) 进行加密。CloudWatch 如果您使用客户管理的 KMS 密钥，则必须向 KMS [密钥添加密钥策略](#)，以允许 CloudWatch Logs 使用您的密钥。

Amazon SQS — 亚马逊 MWAA 为您的环境创建一个Amazon SQS 队列。Amazon MWAA 使用AWS自有的 KMS 密钥或您指定的客户管理的 KMS 密钥使用服务器端加密 (SSE) 对传入和传出队列的数据进行加密。无论您使用的是AWS自有还是客户管理的 KMS 密钥，都必须向您的执行角色添加 Amazon SQS 权限。

Aurora PostgreSQL — 亚马逊 MWAA 为您的环境创建一个 PostgreSQL 集群。Aurora PostgreSQL 使用服务器端加密 (SSE) 使用AWS自有或客户管理的 KMS 密钥对内容进行加密。如果您使用客户管理的 KMS 密

钥，Amazon RDS 会向该密钥添加至少两项授权：一项用于集群，一项用于数据库实例。如果您选择在多个环境中使用客户托管的 KMS 密钥，Amazon RDS 可能会创建额外的授权。有关更多信息，请参阅 [Amazon RDS 中的数据保护](#)。

传输中加密

传输中的数据被称为在网络中传输时可能被截获的数据。

传输层安全性 (TLS) 对您环境的 Apache Airflow 组件和其它与 Amazon MWAA 集成的 AWS 服务（例如 Amazon S3）之间正在传输的 Amazon MWAA 对象进行加密。有关 Amazon S3 加密的更多信息，请参阅 [使用加密保护数据](#)。

使用客户管理的密钥进行加密

您可以选择提供 [客户管理的密钥](#) 用于环境中的数据加密。您必须与您的 Amazon MWAA 环境实例和存储工作流程资源的 Amazon S3 存储桶相同的区域中创建客户管理的 KMS 密钥。如果指定的客户 Managed KMS 密钥没有位于用于配置环境的账户中，则必须使用它的 ARN 指定密钥以进行跨账户访问。有关创建密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 [创建密钥](#)。

支持的内容

AWS KMS 的功能	支持	
密 AWS KMS 密钥 ID 或 ARN 。	是	
密 AWS KMS 密钥别名 。	否	
AWS KMS 多区域密钥 。	否	

使用授权进行加密

该主题介绍 Amazon MWAA 代表您向客户管理型 KMS 密钥授予的用于加密和解密数据的授权。

工作原理

客户管理的 KMS 密钥支持两种基于资源 AWS KMS 的访问控制机制：[密钥策略](#) 和 [授权](#)。

当权限主要为静态权限并在同步服务模式下使用时，将使用密钥策略。当需要更动态和更精细的权限时，例如当服务需要为自身或其他账户定义不同的访问权限时，就会使用授权。

Amazon MWAA 使用四项授权策略并将其附加到您的客户管理的 KMS 密钥。这是因为环境需要精细权限才能加密来自 CloudWatch 日志、Amazon SQS 队列、Aurora PostgreSQL 数据库数据库、Secrets Manager 机密、Amazon S3 存储桶和 DynamoDB 表的静态数据。

当您创建 Amazon MWAA 环境并指定客户管理的 KMS 密钥时，Amazon MWAA 会将授予策略附加到您的客户管理的 KMS 密钥。这些策略允许亚马逊 MWAA 使用您的客户管理的 KMS 密钥代表您加密亚马逊 MWAA 拥有的资源。`airflow.region}.amazonaws.com`

Amazon MWAA 代表您创建并向指定的 KMS 密钥附加额外授权。这包括在您删除环境时取消授权、使用客户管理的 KMS 密钥进行客户端加密 (CSE) 以及需要在 Secrets Manager 中访问受客户托管密钥保护的密钥 AWS Fargate 的执行角色的策略。

拨款政策

Amazon MWAA 代表您向客户管理的 KMS 密钥添加以下 [基于资源的策略](#) 授权。这些策略允许受让人和委托人（Amazon MWAA）执行策略中定义的操作。

Grant 1 : 用于创建数据平面资源

```
{
  "Name": "mwa-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"
  ]
}
```

授权 2 : 用于ControllerLambdaExecutionRole访问

```
{
  "Name": "mwa-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"
  ]
}
```

授权 3 : 用于CfnManagementLambdaExecutionRole访问

```
{
  "Name": " mwa-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"
  ]
}
```

Grant 4 : 用于 Fargate 执行角色访问后端机密

```
{
  "Name": "mwa-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"
    ]
}
```

将密钥策略附加到客户托管密钥

如果您选择在 Amazon MWAA 上使用自己的客户管理的 KMS 密钥，则必须将以下策略附加到该密钥，以允许 Amazon MWAA 使用它来加密您的数据。

如果您在 Amazon MWAA 环境中使用的客户托管 KMS 密钥尚未配置为可使用 CloudWatch，则必须更新[密钥策略](#)以允许加密 CloudWatch 日志。有关更多信息，请参阅[CloudWatch 使用AWS Key Management Service服务加密日志数据](#)。

以下示例代表了 Lo CloudWatch gs 的密钥策略。替换为该区域提供的样本值。

```
{
    "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) 是一种 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（已登录）和授权（具有权限）以使用 Amazon Managed Workflows for Apache Airflow (MWAA) 资源。IAM 是一个可以免费使用的 AWS 服务。

本主题基本概述了亚马逊 MWAA 如何使用AWS Identity and Access Management (IAM)。要了解有关管理亚马逊 MWAA 访问权限的信息，请参阅[管理对 Amazon MWAA 环境的访问 \(p. 32\)](#)。

目录

- [受众 \(p. 284\)](#)
- [使用身份进行身份验证 \(p. 284\)](#)
- [使用策略管理访问 \(p. 285\)](#)
- [允许用户查看他们自己的权限 \(p. 287\)](#)
- [对 Amazon Managed Workflows for Apac \(p. 287\)](#)

- [Amazon MWAA 如何与 IAM 协同工作 \(p. 288\)](#)

受众

AWS Identity and Access Management(IAM) 的使用方式因您可以在 Amazon MWAA 中执行的操作而异。

服务用户 — 如果您使用 Amazon MWAA 服务来完成作业，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon MWAA 功能来完成工作，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon MWAA 中的功能，请参阅[对 Amazon Managed Workflows for Apac \(p. 287\)](#)。

服务管理员 — 如果您在公司负责管理 Amazon MWAA 资源，您可能对 Amazon MWAA 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon MWAA 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon MWAA 搭配使用的更多信息，请参阅[Amazon MWAA 如何与 IAM 协同工作 \(p. 288\)](#)。

IAM 管理员 — 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon MWAA 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon MWAA 基于身份的策略示例，请参阅[Amazon MAA 基于身份的策略示例 \(p. 289\)](#)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center (successor to AWS Single Sign-On)(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的[如何登录到您的 AWS 账户](#)。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行对请求签名的更多信息，请参阅中的[签名版本 4 签名流程AWS 一般参考](#)。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的[多重身份验证](#)和《IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

AWS 账户根用户

当您创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 AWS Account Management 参考指南中的[需要根用户凭证的任务](#)。

IAM 用户和组

[IAM 用户](#)是AWS 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的[对于需要长期凭证的使用场景定期轮换访问密钥](#)。

[IAM 组](#)是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的[何时创建 IAM 用户（而不是角色）](#)。

IAM 角色

[IAM 角色](#)是AWS 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过[切换角色](#)，在 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以担任角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的[使用 IAM 角色](#)。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access (联合用户访问) – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色信息，请参阅《IAM 用户指南》中的[为第三方身份提供商创建角色](#)。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的[权限集](#)。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的[IAM 角色与基于资源的策略有何不同](#)。
- 跨服务访问 – 某些 AWS 服务使用其它 AWS 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service (Amazon S3) 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
 - 主体权限 – 当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅服务授权参考中的[Amazon Amazon Managework on Apache Worklow 的操作、资源和条件键](#)。
 - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 [IAM 角色](#)。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的[创建向 AWS 服务 委派权限的角色](#)。
 - 服务相关角色 – 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的[使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#)。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的[何时创建 IAM 角色（而不是用户）](#)。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的[JSON 策略概览](#)。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。然后，管理员可以向角色添加 IAM 策略，并且用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的 [创建 IAM policy](#)。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的[在托管式策略与内联策略之间进行选择](#)。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service (Amazon S3) 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中[指定主体](#)。主体可以包括账户、用户、角色、联合身份用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的[访问控制列表 \(ACL\) 概览](#)。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 Principal 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 [IAM 实体的权限边界](#)。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的 [SCP 的工作原理](#)。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的 [会话策略](#)。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的[策略评估逻辑](#)。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

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

对 Amazon Managed Workflows for Apac

可以使用以下信息，以帮助您诊断和修复在使用 Amazon MWAA 和 IAM 时可能遇到的常见问题。

我无权在 Amazon MAA 中执行操作

如果 AWS Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

我无权执行 iam : PassRole

如果您收到一个错误，表明您无权执行操作，则iam:PassRole必须更新策略以允许您将角色传递给 Amazon MWAA。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户marymajor尝试使用控制台在 Amazon MWAA 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

我希望允许我的AWS账户以外的人访问我的 Amazon MWAA 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon MWAA 是否支持这些功能，请参阅[Amazon MWAA 如何与 IAM 协同工作 \(p. 288\)](#)。
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的[为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限](#)。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的[为第三方拥有的AWS 账户提供访问权限](#)。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的[为经过外部身份验证的用户（联合身份验证）提供访问权限](#)。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的[IAM 角色与基于资源的策略有何不同](#)。

Amazon MWAA 如何与 IAM 协同工作

亚马逊 MWAA 使用基于 IAM 身份的策略向亚马逊 MWAA 操作和资源授予权限。有关可用于控制对 Amazon MWAA 资源的访问的自定义 IAM 策略的推荐示例，请参阅[the section called “访问Amazon MWAA 环境” \(p. 32\)](#)。

要大致了解 Amazon MWAA 和其他AWS服务如何与 IAM 一起使用，请参阅《IAM 用户指南》中的与 IAM [一起使用的AWS服务](#)。

Amazon MWAA 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。Amazon MWAA 支持特定的操作、资源和条件键。

以下步骤显示如何使用 IAM 控制台创建新的 JSON 策略。此托管策略提供对 Amazon MWAA 资源的只读访问权限。

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航栏中，选择 Policies (策略)。

如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。

3. 在页面的顶部，选择 Create policy (创建策略)。
4. 请选择 JSON 选项卡。

5. 输入以下 JSON 策略文档：

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

6. 选择 Review policy (查看策略)。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的[调整策略结构](#)。

7. 在 Review policy (查看策略) 页面上，为创建的策略输入 Name (名称) 和 Description (说明) (可选)。查看策略摘要以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

要了解在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 [IAM JSON 策略元素参考](#)。

操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件下执行。

JSON 策略的 Action 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

策略语句必须包含 Action 元素或元 NotAction 元素。该 Action 元素列出了策略允许的操作。该 NotAction 元素列出了不允许的操作。

为亚马逊 MWAA 定义的操作反映了您可以使用亚马逊 MWAA 执行的任务。Detective 中的策略操作具有以下前缀:airflow:。

您也可以使用通配符 (*) 指定多个操作。您可以授予对所有以单词结尾的操作的访问权限，而不必单独列出这些操作，例如 environment。

要查看 Amazon MWAA 操作列表，请参阅 IAM 用户指南中的 [Amazon 托管工作流程为 Apache Airflow 定义的操作](#)。

Amazon MAA 基于身份的策略示例

要查看亚马逊 MWAA 政策，请参阅[管理对 Amazon MWAA 环境的访问 \(p. 32\)](#)。

预设情况下，IAM 用户和角色没有创建或修改 Amazon MWAA 资源的权限。它们还无法使用 AWS Management Console、AWS CLI 或 AWS API 执行任务。

IAM 管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员将这些策略附加到需要这些权限的 IAM 用户或组。

Important

我们建议使用 IAM 角色和临时证书来提供对您的 Amazon MWAA 资源的访问权限。避免将权限策略直接附加到您的 IAM 用户。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的 [在 JSON 选项卡上创建策略](#)。

主题

- [策略最佳实践 \(p. 290\)](#)
- [使用 Amazon MWAA 控制台 \(p. 290\)](#)
- [允许用户查看他们自己的权限 \(p. 287\)](#)

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon MWAA 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- AWS 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 [AWS 托管策略](#) 或 [工作职能的 AWS 托管策略](#)。
- 应用最低权限 - 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 [IAM 中的策略和权限](#)。
- 使用 IAM policy 中的条件进一步限制访问权限 - 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 [IAM JSON 策略元素：条件](#)。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 - IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy 语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 [IAM Access Analyzer 策略验证](#)。
- Require multi-factor authentication (MFA) [需要多重身份验证 (MFA)] - 如果您所处的场景要求您的 AWS 账户中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 [配置受 MFA 保护的 API 访问](#)。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 [IAM 中的安全最佳实践](#)。

使用 Amazon MWAA 控制台

要使用 Amazon MWAA 控制台，用户或角色必须有权访问与 API 中的相应操作相匹配的相关操作。

要查看亚马逊 MWAA 政策，请参阅[管理对 Amazon MWAA 环境的访问 \(p. 32\)](#)。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "ViewOwnUserInfo",
    "Effect": "Allow",
    "Action": [
      "iam:GetUserPolicy",
      "iam:ListGroupsWithUser",
      "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 Airflow (

要了解某个 AWS 服务 是否在特定合规性计划范围内，请参阅 [合规性计划范围内的 AWS 服务](#)，然后选择您感兴趣的合规性计划。有关常规信息，请参阅[AWS 合规性计划](#)。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅[在 AWS Artifact 中下载报告](#)。

您使用 AWS 服务 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- [安全性与合规性快速入门指南](#) – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署以安全性和合规性为重点的基准环境的步骤。
- [Amazon Web Services 上的 HIPAA 安全性和合规性架构设计](#) – 该白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

Note

并非所有 AWS 服务都符合 HIPAA 要求。有关更多信息，请参阅[符合 HIPAA 要求的 AWS 服务参考](#)。

- [AWS 合规性资源](#) – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config 开发人员指南中的[使用规则评估资源](#) – 此 AWS Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- [AWS Security Hub](#) – 此 AWS 服务 向您提供 AWS 中安全状态的全面视图。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅[Security Hub 控件参考](#)。
- [AWS Audit Manager](#) – 此 AWS 服务 可帮助您持续审计您的 AWS 使用情况，以简化管理风险以及与相关法规和行业标准的合规性的方式。

Amazon Managed Workflows Apache Airflow

AWS全球基础设施围绕AWS区域和可用区构建。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 [AWS 全球基础设施](#)。

Amazon MWAA 中的基础设施安全性

作为一项托管服务，Amazon MWAA 由 [Amazon Web Services：安全流程概述白皮书](#)中所述的AWS全球网络安全程序提供保护。

您可以使用AWS发布的 API 调用通过网络访问 Amazon MWAA。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 [AWS Security Token Service](#) (AWS STS) 生成临时安全凭证来对请求进行签名。

Amazon MWAA 中的配置和漏洞分析

配置和 IT 控制是AWS和您（我们的客户）之间的共同责任。

Amazon Managed Workflows for Apache Airflow (MWAA)，它会定期修补和升级您环境中的 您应确保对您的 VPC 使用适当的访问策略。

有关更多详细信息，请参阅以下资源：

- [Amazon Managed Workflows for Apache Airflow \(p. 291\)](#)
- [责任共担模式](#)
- [Amazon Web Services：安全过程概述](#)
- [Amazon MWAA 中的基础设施安全性 \(p. 292\)](#)
- [Amazon MWAA 中的安全最佳实践 \(p. 292\)](#)

Amazon MWAA 中的安全最佳实践

Amazon MWAA 提供多个安全功能，供您在开发和实施自己的安全策略时考虑。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

- 使用最不允许的权限策略。仅向用户执行任务所需的资源或操作授予权限。
- AWS CloudTrail用于监控您账户中的用户活动。
- 确保 Amazon S3 存储桶策略和对象 ACL 向关联的 Amazon MWAA 环境中的用户授予将对象放入存储桶的权限。这样可以确保有权向存储桶添加工作流程的用户也有权在 Airflow 中运行工作流程。
- 使用与亚马逊 MWAA 环境关联的 Amazon S3 存储桶。您的 Amazon S3 存储桶可以是任何名称。不要将其他对象存储在存储桶中，也不要将该存储桶与其他服务一起使用。

Apache Airflow 中的安全最佳实践

Apache Airflow 不是多租户的。虽然有一些[访问控制措施](#)可以将某些功能限制为特定用户（[Amazon MWAA 实施了 \(p. 38\)](#)这些措施），但 DAG 创建者确实能够编写可以更改 Apache Airflow 用户权限并与底层元数据库交互的 DAG。

我们建议在 Amazon MWAA 上使用 Apache Airflow 时执行以下步骤，以确保您环境的元数据库和 DAG 的安全。

- 对具有 DAG 写入权限或能够将文件添加到 Amazon S3/dags 文件夹的独立团队使用单独的环境，前提是可以写入环境的用户也可以访问 [Amazon MWAA 执行角色 \(p. 44\)](#)或 [Apache Airflow 连接](#)访问的任何内容。
- 不要提供直接的 Amazon S3 DAG 文件夹访问权限。相反，使用 CI/CD 工具将 DAG 写入 Amazon S3，通过验证步骤确保 DAG 代码符合团队的安全准则。
- 阻止用户访问您环境的 Amazon S3 存储桶。改为使用 DAG 工厂，该工厂基于 YAML、JSON 或其他定义文件生成 DAG，该文件存储在与存储 DAG 的 Amazon MWAA Amazon S3 存储桶不同的位置中。
- 在密钥[管理器中存储密钥 \(p. 79\)](#)。虽然这不会阻止可以编写 DAG 的用户读取机密，但会阻止他们修改您的环境使用的密钥。

检测 Apache Airflow 用户权限的更改

您可以使用 CloudWatch Logs Insights 来检测 DAG 更改 Apache Airflow 用户权限的情况。为此，每当您的一个 DAG 更改 Apache Airflow CloudWatch 用户权限时，您可以使用 EventBridge 计划规则、Lambda 函数和 Logs Insights 向 CloudWatch 指标发送通知。

先决条件

要完成以下步骤，您需要以下满足以下条件：

- 在日志级别启用了所有 Apache Airflow 日志类型的亚马逊 MWAA 环境。INFO 有关更多信息，请参阅[the section called “查看气流日志” \(p. 257\)](#)：

配置有关 Apache Airflow 用户权限更改的通知

1. [创建一个 Lambda 函数](#)，针对五个 Amazon MWAA 环境 CloudWatch 日志组（DAGProcessing、Scheduler 和 Worker）运行以下 Logs Insights 查询字符串。TaskWebServer

```
fields @log, @timestamp, @message | filter @message like "add-role" | stats count() by @log
```

2. [创建按计划运行的 EventBridge 规则](#)，将您在[上一步](#)中创建的 Lambda 函数作为规则的目标。使用 cron 或 rate 表达式配置计划，使其按固定间隔运行。

主题

- ## 关于亚马逊 MWAA 版本

最新版本

Apache Airflow

Note

- 有关在需求文件--constraint中使用的更多信息，请参阅[安装 Python 依赖项 \(p. 118\)](#)。

Apache Airfro	A阿帕奇气Flows	A阿帕奇气流限制	Python 版本
v2.5.1	Apache Airflow v2.5.1 参考指南	Apache Airflow v2.5.1 限制文件	Pyon 10
v2.4.3	Apache Airflow v2.4.3 参考指南	Apache Airflow v2.4.3 限制文件	Pyon 10

AAAAAAAAAAAAAAA AAA

根据 Apache Airflow 社区[发布流程和版本政策](#)，亚马逊 MWAA 承诺在任何给定时间支持至少三个次要版本的 Apache Airflow。我们会在支持结束之日前至少 90 天公告特定 Amazon MWAA 支持结束日期。

常见问题

问：亚马逊 MWAA 支持 Apache Airflow 版本多长时间？

答：亚马逊 MWAA 在 Apache Airflow 次要版本首次推出后至少支持 12 个月。

问：当对 Amazon MWAA 支持结束时，我是否会收到通知？

答：可以。如果您账户中的任何 Amazon MWAA 环境正在运行支持即将结束的版本，Amazon MWAA 会在 AWS Health Dashboard 支持结束之日通过发布通知。

问：在有限支持日期会发生什么？

答：在有限支持日期，您将无法再使用关联版本创建新的 Amazon MWAA 环境。您的现有环境会继续发布，直到支持结束之日。

问：支持结束之日会发生什么？

答：在支持结束之日，您将无法再访问运行相关的 Amazon MWAA 环境。有关在亚马逊 MWAA 上升级到新版本的 Apache Airflow 的说明，请参阅[亚马逊 MWAA 迁移指南](#)。

Amazon Managed Workflows by Apache Airflow (MWAA) 服务终端节点和配额

Amazon Managed Workflows by Apache Airflow (MWAA) 具有以下服务配额和终端节点。服务配额（也称为限制）是您的AWS账户使用的服务资源或操作的最大数量。

目录

- [服务终端节点 \(p. 297\)](#)
- [Service Quotas \(p. 297\)](#)
- [提升配额 \(p. 297\)](#)

服务终端节点

要查看 Amazon MWAA 终端节点列表，请参阅[Amazon Managed Workflows by Apache Airflow \(MWAA\) 终端节点和配额](#)。

Service Quotas

配额名称	描述	默认配额	可调整
环境	每个区域、每个账户的 Amazon MWAA 环境的最大数量。	10	是
每个环境的工件数	每个 Amazon MWAA 环境的最大工作人员数。	25	是

提升配额

要求提升配额，请求提升配额[请求提升配额](#)。

亚马逊 MWAA 常见问题

本页介绍了在使用 Amazon Managed Workflows for Apache Airflow (MWAA) 中可能遇到的常见问题

目录

- [支持的版本 \(p. 298\)](#)
 - [亚马逊 MWAA 对 Apache Airflow v2 支持什么？ \(p. 298\)](#)
 - [为什么不支持旧版本的 Apache Airflow？ \(p. 299\)](#)
 - [我该使用哪个 Python 版本？ \(p. 299\)](#)
 - [亚马逊 MWAA 使用的是pip哪个版本？ \(p. 299\)](#)
- [使用案例 \(p. 299\)](#)
 - [我应在何时使用AWS Step Functions MWAA？ \(p. 299\)](#)
- [环境规范 \(p. 300\)](#)
 - [每个环境有多少任务存储空间可用？ \(p. 300\)](#)
 - [亚马逊 MWAA 环境使用的默认操作系统是什么？ \(p. 300\)](#)
 - [我能否为我的亚马逊 MWAA 环境使用自定义镜像？ \(p. 300\)](#)
 - [亚马逊 MWAA 是否符合 HIPAA 标准？ \(p. 300\)](#)
 - [亚马逊 MWAA 是否支持竞价型实例？ \(p. 300\)](#)
 - [亚马逊 MWAA 是否支持自定义域？ \(p. 300\)](#)
 - [我可以通过 SSH 进入我的环境吗？ \(p. 300\)](#)
 - [为什么 VPC 安全组需要自引用规则？ \(p. 301\)](#)
 - [我能否在 IAM 中隐藏不同群组的环境？ \(p. 301\)](#)
 - [我可以在 Apache Airflow Worker 上存储临时数据吗？ \(p. 301\)](#)
 - [我能否指定超过 25 个 Apache Airflow Worker？ \(p. 301\)](#)
 - [亚马逊 MWAA 是否支持共享亚马逊 VPC 或共享子网？ \(p. 301\)](#)
- [指标 \(p. 301\)](#)
 - [使用哪些指标来确定是否扩展 Workers？ \(p. 301\)](#)
 - [我可以在中创建自定义指标 CloudWatch吗？ \(p. 302\)](#)
- [DAG、操作员、连接和其他问题 \(p. 302\)](#)
 - [我能用PythonVirtualenvOperator吗？ \(p. 302\)](#)
 - [亚马逊 MWAA 识别新的 DAG 文件需要多长时间？ \(p. 302\)](#)
 - [为什么 Apache Airflow 不接收我的 DAG 文件？ \(p. 302\)](#)
 - [我可以requirements.txt从环境中移除plugins.zip或吗？ \(p. 302\)](#)
 - [为什么我在 Apache Airflow v2.0.2 管理员插件菜单中看不到我的插件？ \(p. 302\)](#)
 - [我可以使用DataAWS abase Migration Service e \(DMS\) 操作员？ \(p. 303\)](#)

支持的版本

亚马逊 MWAA 对 Apache Airflow v2 支持什么？

要了解亚马逊 MWAA 支持的内容，请参

阅[Amazon Managed Workflows for Apache Airflow 支持的版本 \(p. 294\)](#)。

为什么不支持旧版本的 Apache Airflow ？

由于旧版本存在安全问题，我们仅支持最新的（截至发布时）Apache Airflow v1.10.12。

我该使用哪个 Python 版本 ？

Amazon Managed Workflows for Apache Airflow (MWAA) 中支持以下版本。

Note

- 从 Apache Airflow v2.2.2 开始，亚马逊 MWAA 支持直接在 Apache Airflow Web 服务器上安装 Python 需求、提供程序包和自定义插件。
- 从 Apache Airflow v2.5.1 开始，如果您在使用安装依赖项时未指定该 -c 标志，Amazon MWAA 将使用 requirements.txt 该标志应用一条 --constraint 语句。亚马逊 MWAA 将应用与以下版本表中列出的 Apache Airflow 版本相对应的限制声明。

有关在需求文件 --constraint 中使用的更多信息，请参阅 [安装 Python 依赖项 \(p. 118\)](#)。

Apache Airflow	Apache 气流指南	Apache 气流限制	Python 版本
v2.5.1	Apache Airflow v2.5.1 参考指南	Apache Airflow v2.5.1 限制文件	Python 3.10
v2.4.3	Apache Airflow v2.4.3 参考指南	Apache Airflow v2.4.3 限制文件	Python 3.10
v2.2.2	Apache Airflow v2.2.2 参考指南	Apache Airflow v2.2.2 限制文件	Python 3.7
v2.0.2	Apache Airflow v2.0.2 参考指南	Apache Airflow v2.0.2 限制文件	Python 3.7
v1.10.12	Apache Airflow	Apache Airflow v1.10.12	Python 3.7

有关迁移自我管理的 Apache Airflow 部署或迁移现有 Amazon MWAA 环境的更多信息，包括备份元数据数据库的说明，请参阅 [Amazon MWAA 迁移指南](#)。

亚马逊 MWAA 使用的是pip哪个版本 ？

对于运行 Apache Airflow v1.10.12 的环境，亚马逊 MWAA 会安装 21.1.2pip 版。

Note

亚马逊 MWAA 不会为 Apache Airflow v1.10.12 环境升级pip。

对于运行 Apache Airflow v2 及更高版本的环境，亚马逊 MWAA 会安装 21.3.1pip 版本。

使用案例

我应在何时使用AWS Step Functions MWAA ？

1. 您可以使用Step Functions来处理单个客户订单，因为Step Functions可以扩展以满足对一个订单或一百万个订单的需求。

2. 如果您正在运行夜间工作流程来处理前一天的订单，则可以使用 Step Functions 或 Amazon MWAA。Amazon MWAA 允许您使用开源选项从您正在使用的AWS资源中抽象工作流程。

环境规范

每个环境有多少任务存储空间可用？

任务存储空间限制在 10 GB 以内，由[亚马逊 ECS Fargate 1.3](#) 指定。RAM 量由您指定的环境类决定。有关环境类的更多信息，请参阅[配置个 Amazon MWAA 环境类 \(p. 85\)](#)。

亚马逊 MWAA 环境使用的默认操作系统是什么？

亚马逊 MWAA 环境是在运行亚马逊 Linux AMI 的实例上创建的。

我能否为我的亚马逊 MWAA 环境使用自定义镜像？

不支持自定义映像。亚马逊 MWAA 使用在亚马逊 Linux AMI 上构建的映像。Amazon MWAA 通过运行您添加到该环境 `pip3 -r install` 的 Amazon S3 存储桶的 `requirements.txt` 文件中指定的要求来安装其他要求。

亚马逊 MWAA 是否符合 HIPAA 标准？

Amazon MWAA 符合[Health Insurance Portability and Accountability Act \(HIPAA\)](#) 的条件。如果您有 HIPAA 商业伙伴附录 (BAA)，则可以使用 Amazon MWAA 在 2022 年 11 月 14 日或之后创建的环境中处理受保护 Health 信息 (PHI) 的工作流程。AWS

亚马逊 MWAA 是否支持竞价型实例？

亚马逊 MWAA 目前不支持 Apache Airflow 的按需 Amazon EC2 竞价实例类型。但是，亚马逊 MWAA 环境可以在亚马逊 EMR 和 Amazon EC2 等平台上触发竞价型实例。

亚马逊 MWAA 是否支持自定义域？

要能够使用自定义域作为 Amazon MWAA 主机名，请执行以下操作之一：

- 对于具有公共 Web 服务器访问权限的亚马逊 MWAA 部署，您可以使用 CloudFront 带有 Lambda @Edge 的亚马逊将流量引导到您的环境，并将自定义域名映射到 CloudFront。有关为公共环境设置自定义域的更多信息和示例，请参阅[Amazon MWAA 示例 GitHub 存储库中的 Amazon MWAA 公共 Web 服务器自定义域示例](#)。
- 对于具有私有 Web 服务器访问权限的 Amazon MWAA 部署，您可以使用 Application Load Balancer (ALB) 将流量定向到 Amazon MWAA 并将自定义域名映射到 ALB。有关更多信息，请参阅[the section called “使用 Load Balancer \(高级\)” \(p. 135\)](#)：

我可以通过 SSH 进入我的环境吗？

虽然 Amazon MWAA 环境不支持 SSH，但可以使用 DAG 运行 `bash` 命令，使用 `BashOperator`。例如：

```
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'] }}"
    )
```

要在 Apache Airflow 用户界面中触发 DAG，请使用：

```
{ "command" : "your bash command" }
```

为什么 VPC 安全组需要自引用规则？

通过创建自引用规则，您可以将源限制在 VPC 中的同一个安全组内，并且它并不对所有网络开放。要了解更多信息，请参阅 [the section called “您的 VPC 安全” \(p. 129\)](#)。

我能否在 IAM 中隐藏不同群组的环境？

您可以通过在中指定环境名称来限制访问权限 AWS Identity and Access Management，但是，AWS 控制台中不提供可见性筛选，如果用户可以看到一个环境，则可以看到所有环境。

我可以在 Apache Airflow Worker 上存储临时数据吗？

您的 Apache Airflow 操作员可以存储工作人员的临时数据。Apache Airflow Workers/tmp 可以访问您环境的 Fargate 容器中的临时文件。

Note

根据 [亚马逊 ECS Fargate 1.3 的说法](#)，任务总存储空间限制在 **10 GB 以内**。无法保证后续任务将在同一 Fargate 容器实例上运行，该实例可能使用不同的/tmp文件夹。

我能否指定超过 25 个 Apache Airflow Worker？

是。尽管您最多可以在 Amazon MWAA 控制台上指定 25 个 Apache Airflow 工作人员，但您可以通过请求增加配额在环境中最多配置 50 个。有关更多信息，请参阅 [Requesting a quota increase](#)（请求增加限额）。

亚马逊 MWAA 是否支持共享亚马逊 VPC 或共享子网？

亚马逊 MWAA 不支持共享的亚马逊 VPC 或共享子网。您在创建环境时选择的 Amazon VPC 应归尝试创建环境的账户所有。但是，您可以将流量从亚马逊 MWAA 账户中的亚马逊 VPC 路由到共享 VPC。要了解更多信息，并查看将流量路由到共享 Amazon VPC 的示例，请参阅 Amazon VPC Transit Gateways 指南中的 [集中出站路由到互联网](#)。

指标

使用哪些指标来确定是否扩展 Workers？

亚马逊 MWAA 监控 QueuedTasks 和输入，CloudWatch 以确定是否 RunningTasks 在您的环境中扩展 Apache Airflow Workers。要了解更多信息，请参阅 [监控和指标 \(p. 255\)](#)。

我可以在中创建自定义指标 CloudWatch 吗？

不在 CloudWatch 主机上。但是，您可以创建一个写入自定义指标的 DAG CloudWatch。有关更多信息，请参阅 [the section called “使用 DAG 写入自定义指标” \(p. 179\)](#)：

DAG、操作员、连接和其他问题

我能用PythonVirtualenvOperator吗？

Amazon MWAA 未明确支持，但您可以创建使用 PythonVirtualenvOperator。PythonVirtualenvOperator 有关示例代码，请参阅 [the section called “要修补的自定义插件 PythonVirtualenvOperator” \(p. 207\)](#)。

亚马逊 MWAA 识别新的 DAG 文件需要多长时间？

DAG 会定期从 Amazon S3 存储桶同步到您的环境。如果您添加新的 DAG 文件，Amazon MWAA 需要大约 300 秒才能开始使用新文件。如果您更新现有 DAG，Amazon MWAA 大约需要 30 秒钟才能识别您的更新。

这些值，新 DAG 为 300 秒，现有 DAG 更新为 30 秒，[min_file_process_interval](#) 分别对应于 Apache Airflow 配置选项 [dag_dir_list_interval](#) 和。

为什么 Apache Airflow 不接收我的 DAG 文件？

以下是可能的解决方案：

1. 检查您的执行角色是否有足够的 Amazon S3 存储桶。要了解更多信息，请参阅 [Amazon MWAA 执行角色 \(p. 44\)](#)。
2. 检查 Amazon S3 存储桶是否配置了封禁公共访问并启用了版本控制。要了解更多信息，请参阅 [为 Amazon MWAA 创建 Amazon S3 存储桶 \(p. 16\)](#)。
3. 验证 DAG 文件本身。例如，确保每个 DAG 都有唯一的 DAG ID。

我可以 requirements.txt 从环境中移除 plugins.zip 吗？

目前，一旦添加 plugins.zip 或 requirements.txt，就无法将其从环境中删除，但我们正在解决这个问题。在此期间，解决方法是分别指向一个空的文本文件或 zip 文件。要了解更多信息，请参阅 [删除 Amazon S3 上的文件 \(p. 122\)](#)。

为什么我在 Apache Airflow v2.0.2 管理员插件菜单中看不到我的插件？

出于安全原因，亚马逊 MWAA 上的 Apache Airflow Web 服务器的网络出口量有限，并且不会直接在 2.0.2 版环境的 Apache Airflow Web 服务器上安装插件或 Python 依赖项。显示的插件允许亚马逊 MWAA 在 AWS Identity and Access Management (IAM) 中对您的 Apache Airflow 用户进行身份验证。

为了能够直接在 Web 服务器上安装插件和 Python 依赖项，我们建议使用 Apache Airflow v2.2 及更高版本创建一个新环境。亚马逊 MWAA 直接在 Apache Airflow v2.2 及更高版本的 Web 服务器上安装 Python 依赖项和自定义插件。

我可以使用DatiDatAWS abase Migration Service e (DMS) 操作员？

亚马逊 MWAA 支持 [DMS 运营商](#)。但是，此运算符不能用于对与亚马逊 MWAA 环境关联的 Amazon Aurora PostgreSQL 元数据数据库执行操作。

Amazon Managed Workflows a Airflow (MWAA)

本主题介绍了在亚马逊托管工作流上使用 Apache Airflow (MWAA) 上的 Apache Airflow 时可能遇到的常见问题和错误，以及解决这些错误的建议步骤。

目录

- [故障排除：Apache Airflow v2 中的 DAG、运营商、连接和其他问题 \(p. 305\)](#)
 - [连接 \(p. 306\)](#)
 - [我无法连接到 Secrets Manager \(p. 306\)](#)
 - [如何配置secretsmanager:ResourceTag/<tag-key>我的执行角色策略中的秘密管理器条件还是资源限制？ \(p. 306\)](#)
 - [我无法连接到 Snowflake \(p. 306\)](#)
 - [我在 Airflow UI 中看不到我的连接 \(p. 307\)](#)
 - [Web 服务器 \(p. 308\)](#)
 - [我看到访问 Web 服务器时出现 5xx 错误 \(p. 308\)](#)
 - [我看到“计划程序似乎没有运行”错误 \(p. 308\)](#)
 - [任务 \(p. 308\)](#)
 - [我看到我的任务卡住或没完成 \(p. 308\)](#)
 - [CLI \(p. 310\)](#)
 - [在 CLI 中触发 DAG 时，我看到“503”错误 \(p. 310\)](#)
 - [为什么选择dags backfillApache 气流 CLI 命令失败？有解决方法吗？ \(p. 311\)](#)
 - [运算符 \(p. 311\)](#)
 - [我收到了PermissionError: \[Errno 13\] Permission denied使用 S3Transform 运算符时出错 \(p. 311\)](#)
- [故障排除：Apache Airflow v1 中的 DAG、运营商、连接和其他问题 \(p. 312\)](#)
 - [正在更新 requirements.txt \(p. 313\)](#)
 - [添加apache-airflow-providers-amazon导致我的环境失败 \(p. 313\)](#)
 - [DAG 破碎 \(p. 313\)](#)
 - [使用 Amazon DynamoDB 运营商时，我收到了“破坏 DAG”错误 \(p. 313\)](#)
 - [我收到了“破碎的 DAG：没有名为 psycpg2' 的模块错误 \(p. 313\)](#)
 - [我在使用 Slack 运算符时收到了“破碎的 DAG”错误 \(p. 313\)](#)
 - [我在安装GOOGLE /GCP/BigQuery 时收到了各种错误 \(p. 314\)](#)
 - [我收到了“破碎的 DAG：没有名为 Cython' 的模块错误 \(p. 314\)](#)
 - [运算符 \(p. 314\)](#)
 - [我使用 BigQuery 运算符收到错误 \(p. 314\)](#)
 - [连接 \(p. 315\)](#)
 - [我无法连接到 Snowflake \(p. 306\)](#)
 - [我无法连接到 Secrets Manager \(p. 316\)](#)
 - [我无法在 '<DB-identifier-name>.cluster-id 上连接到我的 MySQL 服务器。<region>.rds.amazonaws.com' \(p. 316\)](#)
 - [Web 服务器 \(p. 316\)](#)
 - [我正在使用 BigQueryOperator，这导致我的网络服务器崩溃 \(p. 316\)](#)
 - [我看到访问 Web 服务器时出现 5xx 错误 \(p. 316\)](#)

- [我看到“计划程序似乎没有运行”错误 \(p. 317\)](#)
- [任务 \(p. 317\)](#)
 - [我看到我的任务卡住或没完成 \(p. 317\)](#)
- [CLI \(p. 319\)](#)
 - [在 CLI 中触发 DAG 时我看到“503”错误 \(p. 319\)](#)
- [故障排除：创建和更新 Amazon MWAA 环境 \(p. 320\)](#)
 - [更新requirements.txt \(p. 320\)](#)
 - [我指定了我的新版本requirements.txt而且更新我的环境需要 20 多分钟 \(p. 313\)](#)
 - [插件 \(p. 321\)](#)
 - [亚马逊 MWAA 是否支持实施自定义 UI？ \(p. 321\)](#)
 - [我能够在亚马逊 MWAA 本地跑步者通过插件，但是当我尝试在亚马逊 MWAA 上执行同样的操作时，我看不到我的更改或任何错误。为什么会发生这种情况？ \(p. 321\)](#)
 - [创建存储桶 \(p. 321\)](#)
 - [我无法选择 S3 阻止公有访问设置选项 \(p. 321\)](#)
 - [创建环境。\(p. 322\)](#)
 - [我试图创建一个环境，它陷入了“正在创建”状态 \(p. 322\)](#)
 - [我试图创建一个环境，但它显示状态为“创建失败” \(p. 322\)](#)
 - [我尝试选择 VPC 但收到“网络故障”错误 \(p. 322\)](#)
 - [我试图创建环境并收到服务、分区或资源“必须通过”错误 \(p. 323\)](#)
 - [我尝试创建一个环境，它将状态显示为“可用”，但是当我尝试访问 Airflow UI 时，显示“来自服务器的空回复”或“502 坏网关”错误 \(p. 323\)](#)
 - [我试图创建一个环境，我的用户名是一堆随机的角色名称 \(p. 323\)](#)
 - [更新环境 \(p. 323\)](#)
 - [我尝试更改环境类，但更新失败 \(p. 323\)](#)
 - [访问环境 \(p. 323\)](#)
 - [我无法访问 Apache Airflow UI \(p. 323\)](#)
- [故障排除：CloudWatch 日志和 CloudTrail 错误 \(p. 324\)](#)
 - [日志 \(p. 324\)](#)
 - [我看不到我的任务日志，或者我收到了“从 Cloudwatch log_group 读取远程日志”错误 \(p. 324\)](#)
 - [任务失败，没有任何日志 \(p. 324\)](#)
 - [我看见了'ResourceAlreadyExistsException'错误消息 CloudTrail \(p. 325\)](#)
 - [我在中看到“请求无效”错误 CloudTrail \(p. 325\)](#)
 - [我看到“找不到 64 位 Oracle Client 库：” libclntsh.so：无法打开共享目标文件：Apache Airflow 日志中没有“这样的文件或目录” \(p. 326\)](#)
 - [我在我的调度器日志中看到 psycpg2 “服务器意外关闭了连接” \(p. 326\)](#)
 - [我在 DAG 处理日志中看到“Executor 报告任务实例 %s 已完成 \(%s\)，尽管任务显示为 %s” \(p. 326\)](#)
 - [我看见 '无法从 log_group 读取远程日志：airflow-*{*EnvironmentName}-Task log_stream: *{*DAG_ID}/*{*n}.log。”在我的任务日志中 \(p. 327\)](#)

故障排除：Apache Airflow v2 中的 DAG、运营商、连接和其他问题

本页上的主题描述了 Apache Airflow v2 Python 依赖项、自定义插件、DAG、操作员、连接、任务和的解决方案Web 服务器在 Amazon Manage Workflow of Apache Airflow (MWAA) 环境中可能会遇到的问题。

目录

- [连接 \(p. 306\)](#)
 - [我无法连接到 Secrets Manager \(p. 306\)](#)
 - [如何配置secretsmanager:ResourceTag/<tag-key>我的执行角色策略中的秘密管理器条件还是资源限制？ \(p. 306\)](#)
 - [我无法连接到 Snowflake \(p. 306\)](#)
 - [我在 Airflow UI 中看不到我的连接 \(p. 307\)](#)
- [Web 服务器 \(p. 308\)](#)
 - [我看到访问 Web 服务器时出现 5xx 错误 \(p. 308\)](#)
 - [我看到“计划程序似乎没有运行”错误 \(p. 308\)](#)
- [任务 \(p. 308\)](#)
 - [我看到我的任务卡住或没完成 \(p. 308\)](#)
- [CLI \(p. 310\)](#)
 - [在 CLI 中触发 DAG 时，我看到“503”错误 \(p. 310\)](#)
 - [为什么选择dags backfillApache 气流 CLI 命令失败？有解决方法吗？ \(p. 311\)](#)
- [运算符 \(p. 311\)](#)
 - [我收到了PermissionError: \[Errno 13\] Permission denied使用 S3Transform 运算符时出错 \(p. 311\)](#)

连接

以下主题描述了在使用 Apache Airflow 连接或使用其他连接时可能收到的错误AWS数据库。

我无法连接到 Secrets Manager

我们建议您完成以下步骤：

1. 了解如何为 Apache Airflow 连接创建密钥和变量[the section called “配置Secrets Manager” \(p. 79\)](#).
2. 了解如何使用 Apache Airflow 变量的密钥 (test-variable) 在[在中使用密钥AWS Secrets Manager对于 Apache 气流变量 \(p. 187\)](#).
3. 了解如何使用密钥进行 Apache Airflow 连接 (myconn) 在[在中使用密钥AWS Secrets Manager用于 Apache 气流连接 \(p. 189\)](#).

如何配置secretsmanager:ResourceTag/<tag-key>我的执行角色策略中的秘密管理器条件还是资源限制？

Note

适用于 Apache Airflow 2.0 版及更早版本。

目前，由于 Apache Airflow 中存在已知问题，您无法通过在环境的执行角色中使用条件键或其他资源限制来限制对 Secret Secrets Manager 密钥的访问。

我无法连接到 Snowflake

我们建议您完成以下步骤：

1. 使用在本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下条目添加到适用于您的环境的 requirements.txt。

```
apache-airflow-providers-snowflake==1.3.0
```

3. 将以下导入添加到您的 DAG：

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

确保 Apache Airflow 连接对象包含以下键/值对：

1. 康恩 Id：snowflake_conn
2. Conn 类型：Snowflake
3. 主机：<my account>。 <my region if not us-west-2>.snowflake计算ing.com
4. 架构：<my schema>
5. 登录：<my user name>
6. 密码：*****
7. 端口：<port, if any>
8. 额外：

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

例如：

```
>>> 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')),
... )
```

我在 Airflow UI 中看不到我的连接

Apache Airflow 在 Apache Airflow UI 中提供了连接模板。无论连接类型如何，它都使用它来生成连接 URI 字符串。如果 Apache Airflow UI 中没有连接模板，则可以使用备用连接模板生成连接 URI 字符串，例如使用 HTTP 连接模板。

我们建议您完成以下步骤：

1. 在 Apache Airflow UI 中查看亚马逊 MWAA 提供的连接类型，网址为[安装在亚马逊 MWAA 环境中的 Apache Airflow 提供商软件包 \(p. 73\)](#)。
2. 在 CLI 中查看用于创建 Apache Airflow 连接的命令，请访问[Apache Airflow Air \(p. 66\)](#)。
3. 了解如何在 Apache Airflow UI 中互换使用连接模板来处理亚马逊 MWAA 上 Apache Airflow UI 中不可用的连接类型[连接类型概览 \(p. 76\)](#)。

Web 服务器

以下主题描述了 Apache Airflow 可能会收到的错误Web 服务器在 Amazon MWAA 上。

我看到访问 Web 服务器时出现 5xx 错误

我们建议您完成以下步骤：

1. 检查 Apache Airflow 配置选项。验证您指定为 Apache Airflow 配置选项的键值对，例如AWS Secrets Manager，配置正确。要了解更多信息，请参阅 [the section called “我无法连接到 Secrets Manager” \(p. 316\)](#)。
2. 检查requirements.txt。验证你列出的 Airflow “额外” 包和其他库requirements.txt与你的 Apache Airflow 版本兼容。
3. 探索在requirements.txt文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我看到“计划程序似乎没有运行”错误

如果计划程序似乎没有运行，或者几个小时前收到了最后一个“心跳”，那么你的 DAG 可能不会出现在 Apache Airflow 中，也不会计划新的任务。

我们建议您完成以下步骤：

1. 确认您的 VPC 安全组允许入站访问端口5432。连接到您的环境的 Amazon Aurora PostgreSQL 元数据库需要此端口。添加此规则后，给亚马逊 MWAA 几分钟时间，错误应该消失。要了解更多信息，请参阅 [the section called “您的 VPC 安全” \(p. 129\)](#)。

Note

- Aurora PostgreSQL 元数据库是[Amazon MWAA 服务架构 \(p. 2\)](#)并且在你的AWS 账户。
 - 与数据库相关的错误通常是调度程序故障的症状，而不是根本原因。
2. 如果调度程序没有运行，可能是由于许多因素造成的，例如[依赖项安装失败 \(p. 248\)](#)，或者[计划程序重载 \(p. 238\)](#)。通过查看 CloudWatch Logs 中的相应日志组，确认 DAG、插件和要求是否正常运行。要了解更多信息，请参阅 [监控和指标 \(p. 255\)](#)。

任务

以下主题描述了在环境中针对 Apache Airflow 任务可能收到的错误。

我看到我的任务卡住或没完成

如果您的 Apache Airflow 任务被“卡住”或未完成，建议您执行以下步骤：

1. 可能定义了太多 DAG。减少 DAG 的数量并执行环境更新（例如更改日志级别）以强制重置。
 - a. 无论它们是否启用，气流都会解析 DAG。如果你使用的是环境容量的 50% 以上，你可能会开始压倒 Apache Airflow计划程序。这会导致大总解析时间在 CloudWatch 指标中或 CloudWatch Logs 中的长 DAG 处理时间。还有其它方法可以优化 Apache Airflow 配置，这些方法不在本指南的讨论范围内。
 - b. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅[the section called “Amahe Ahe Ahe Ahe” \(p. 238\)](#)。
2. 队列中可能会有大量任务。这通常显示为处于“无”状态的大量且不断增长的任务数量，或者显示为大量处于排队的任务和/或待处理任务在 CloudWatch 中。出现此错误的原因如下：

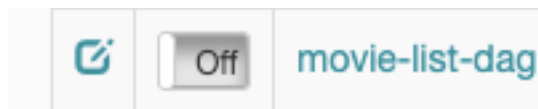
- a. 如果要运行的任务多于环境的运行能力，和/或在自动扩展有时间检测任务并部署其他任务之前排队的大量任务劳动力。
- b. 如果要运行的任务多于环境的运行容量，我们建议减少你的 DAG 同时运行的任务数量，和/或增加最低 Apache Airflow 劳动力。
- c. 如果在自动扩缩有时间检测和部署其他工作人员之前有大量的任务排队，我们建议跟踪任务部署和/或增加最低 Apache Airflow 劳动力。
- d. 您可以使用 [update-environment](#) 中的命令 AWS Command Line Interface (AWS CLI) 更改的最小或最大数量劳动力在你的环境中运行。

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-workers 10
```

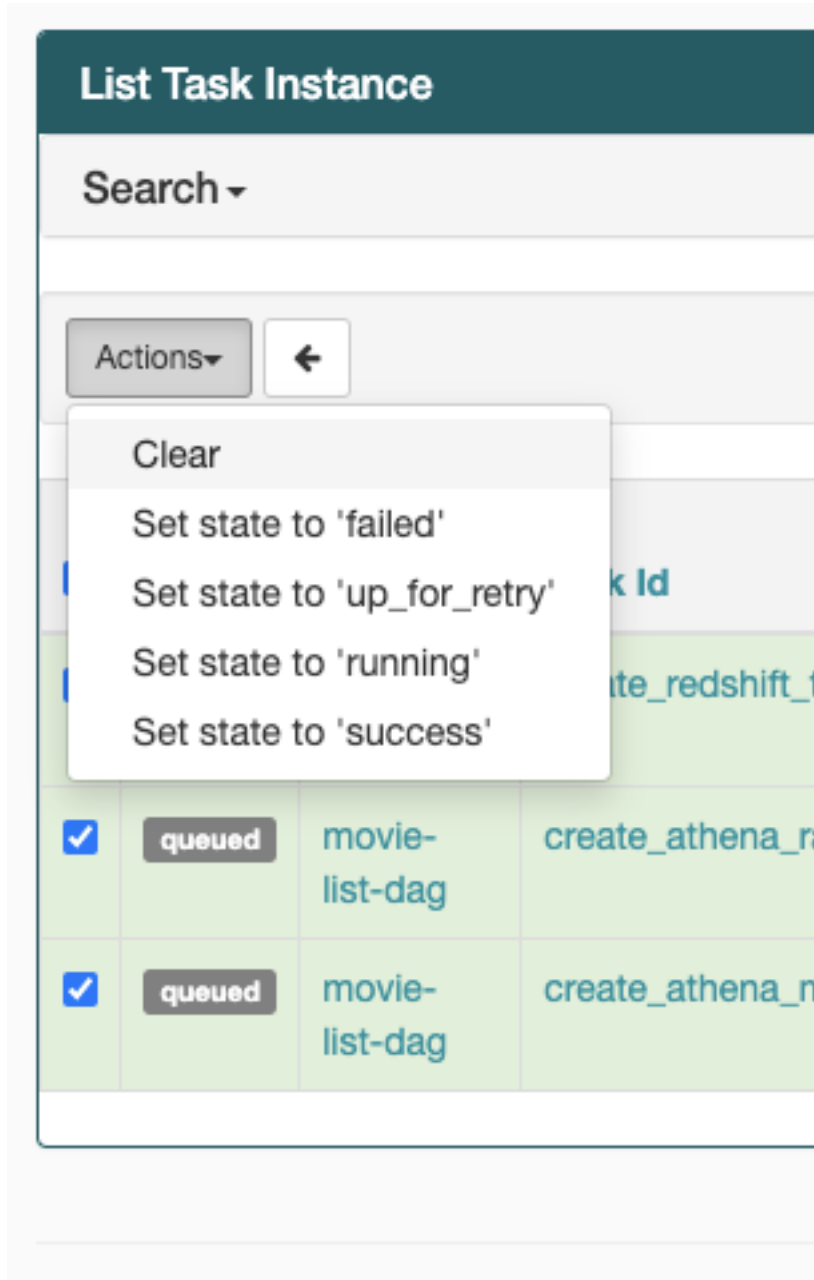
- e. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅 [the section called “Amazon MWAA” \(p. 238\)](#).
3. 可能会在执行过程中删除任务，这些任务显示为任务日志，这些任务将停止，在 Apache Airflow 中没有进一步的指示 出现此错误的原因如下：
 - a. 如果有一个短暂的时刻，1) 当前任务超出当前环境容量，然后是 2) 几分钟没有执行或正在排队的任务，那么 3) 新任务正在排队。
 - b. Amazon MWAA 自动扩缩通过添加额外的工作人员来对第一种情况做出反应。在第二种情况下，它删除了额外的工作人员。一些正在排队的任务可能会导致正在移除的工作人员，并将在容器被删除后结束。
 - c. 我们建议增加环境中的最低工作人员数量。另一种选择是调整 DAG 和任务的时间，以确保不会发生这些情况。
 - d. 您还可以将最少工作线程设置为等于环境中的最大工作线程数，从而有效地禁用自动扩展。使用 [update-environment](#) 中的命令 AWS Command Line Interface (AWS CLI) 到禁用自动扩展将最小工作线程和最大工作线程设置为相同。

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-workers 5
```

- e. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅 [the section called “Amazon MWAA” \(p. 238\)](#).
4. 如果您的任务陷入“正在运行”状态，您也可以清除任务或将其标记为成功或失败。这使您的环境的自动扩展组件可以缩小环境中运行的工作线程的数量。下图显示了搁置任务的示例。



- 选择搁置任务的圆圈，然后选择 Clear (如图所示)。这允许 Amazon MWAA 缩小工作人员；否则，如果仍有排队任务，亚马逊 MWAA 将无法确定哪些 DAG 已启用或禁用，也无法缩小规模。



5. 在此了解有关 Apache Airflow 任务生命周期的更多信息，[概念](#)中的Apache Airflow 参考指南.

CLI

以下主题描述了在AWS Command Line Interface.

在 CLI 中触发 DAG 时，我看到“503”错误

气流 CLI 在 Apache 气流上运行Web 服务器，它的并发性有限。通常最多可以同时运行 4 个 CLI 命令。

为什么选择dags backfillApache 气流 CLI 命令失败？有解决方法吗？

Note

以下条款仅适用于 Apache Airflow v2.0.2 环境。

这些区域有：backfill与其他 Apache Airflow CLI 命令一样，命令会在处理任何 DAG 之前在本地解析所有 DAG，而不管 CLI 操作适用于哪个 DAG。在使用 Apache Airflow v2.0.2 的 Amazon MWAA 环境中，因为在 CLI 命令运行时，网络服务器上尚未安装插件和要求，因此解析操作失败，而且backfill操作未被调用。如果您的环境中没有任何要求或插件，backfill操作将会成功。

为了能够运行backfillCLI 命令，我们建议在 bash 运算符中调用它。在 bash 操作员中，backfill是从工作人员启动的，允许 DAG 成功解析，因为所有必要的要求和插件都可用并已安装。以下示例演示如何使用创建 DAGBashOperator运行时长backfill。

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="backfill_dag", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="airflow dags backfill my_dag_id"
    )
```

运算符

以下主题介绍了在使用 Operator 时可能收到的错误。

我收到了PermissionError: [Errno 13] Permission denied使用 S3Transform 运算符时出错

如果你尝试使用S3Transform运营商而且你正在收到PermissionError: [Errno 13] Permission denied错误消息。以下步骤假定您有一个现有 plugins.zip 文件。如果您在创建新的plugins.zip，请参阅[安装自定义插件 \(p. 106\)](#)。

1. 使用在本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 创建你的“转换”脚本。

```
#!/bin/bash
cp $1 $2
```

3. （可选）macOS 和 Linux 用户可能需要运行以下命令以确保脚本可执行。

```
chmod 777 transform_test.sh
```

4. 将脚本添加到你的 plugins.zip 中。

```
zip plugins.zip transform_test.sh
```

5. 按中的步骤操作。[将 plugins.zip 上传到 Amazon S3 \(p. 115\)](#)。
6. 按中的步骤操作。[在亚马逊 MWAA 控制台上指定 plugins.zip 版本 \(p. 116\)](#)。

7. 创建以下 DAG。

```
from airflow import DAG
from airflow.providers.amazon.aws.operators.s3_file_transform import
    S3FileTransformOperator
from airflow.utils.dates import days_ago
import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False, start_date=days_ago(1))
    as dag:
        file_transform = S3FileTransformOperator(
            task_id='file_transform',
            transform_script='/usr/local/airflow/plugins/transform_test.sh',
            source_s3_key='s3://YOUR_S3_BUCKET/files/input.txt',
            dest_s3_key='s3://YOUR_S3_BUCKET/files/output.txt'
        )
```

8. 按中的步骤操作。 [将 DAG 代码上载到 Amazon S3 \(p. 105\)](#)。

故障排除：Apache Airflow v1 中的 DAG、运营商、连接和其他问题

本页上的主题包含 Apache Airflow v1.10.12 Python 依赖项、自定义插件、DAG、操作员、连接、任务和的解决方案Web 服务器您在 Amazon Manage Workflow for Apache Airflow (MWAA) 环境中可能遇到的问题。

目录

- [正在更新 requirements.txt \(p. 313\)](#)
 - [添加apache-airflow-providers-amazon导致我的环境失败 \(p. 313\)](#)
- [DAG 破碎 \(p. 313\)](#)
 - [使用 Amazon DynamoDB 运营商时，我收到了“破坏 DAG”错误 \(p. 313\)](#)
 - [我收到了“破碎的 DAG：没有名为 psycopg2”的模块错误 \(p. 313\)](#)
 - [我在使用 Slack 运算符时收到了“破碎的 DAG”错误 \(p. 313\)](#)
 - [我在安装GOOGLE /GCP/BigQuery 时收到了各种错误 \(p. 314\)](#)
 - [我收到了“破碎的 DAG：没有名为 Cython”的模块错误 \(p. 314\)](#)
- [运算符 \(p. 314\)](#)
 - [我使用 BigQuery 运算符收到错误 \(p. 314\)](#)
- [连接 \(p. 315\)](#)
 - [我无法连接到 Snowflake \(p. 306\)](#)
 - [我无法连接到 Secrets Manager \(p. 316\)](#)
 - [我无法在 '<DB-identifier-name>.cluster-id' 上连接到我的 MySQL 服务器。<region>.rds.amazonaws.com' \(p. 316\)](#)
- [Web 服务器 \(p. 316\)](#)
 - [我正在使用 BigQueryOperator，这导致我的网络服务器崩溃 \(p. 316\)](#)
 - [我看到访问 Web 服务器时出现 5xx 错误 \(p. 316\)](#)
 - [我看到“计划程序似乎没有运行”错误 \(p. 317\)](#)
- [任务 \(p. 317\)](#)
 - [我看到我的任务卡住或没完成 \(p. 317\)](#)
- [CLI \(p. 319\)](#)

- [在 CLI 中触发 DAG 时我看到“503”错误 \(p. 319\)](#)

正在更新 requirements.txt

以下主题描述了在更新时可能收到的错误requirements.txt。

添加apache-airflow-providers-amazon导致我的环境失败

apache-airflow-providers-xyz仅与 Apache Airflow v2 兼容。apache-airflow-backport-providers-xyz与 Apache 气流 1.10.12 兼容。

DAG 破碎

以下主题描述了运行 DAG 时可能会收到的错误。

使用 Amazon DynamoDB 运营商时，我收到了“破坏 DAG”错误

我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下包添加到您的requirements.txt。

```
boto
```

3. 探索在中指定 Python 依赖关系的方法requirements.txt文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我收到了“破碎的 DAG：没有名为 psycopg2 的模块错误

我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下内容添加到您的requirements.txt使用你的 Apache 气流版本。例如：

```
apache-airflow[postgres]==1.10.12
```

3. 探索在中指定 Python 依赖关系的方法requirements.txt文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我在使用 Slack 运算符时收到了“破碎的 DAG”错误

我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下包添加到您的requirements.txt然后指定你的 Apache Airflow 版本。例如：

```
apache-airflow[slack]==1.10.12
```

3. 探索在中指定 Python 依赖关系的方法requirements.txt文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我在安装GOOGLE /GCP/BigQuery 时收到了各种错误

亚马逊 MWAA 使用亚马逊 Linux，它需要特定版本的 Cython 和加密库。我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下包添加到您的 requirements.txt。

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow-backport-providers-amazon[google]
```

3. 如果您使用的不是 Backport 提供商，您可以使用：

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow[gcp]==1.10.12
```

4. 探索在中指定 Python 依赖关系的方法 requirements.txt 文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我收到了“破碎的 DAG：没有名为 Cython' 的模块错误

亚马逊 MWAA 使用亚马逊 Linux，它需要特定版本的 Cython。我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下包添加到您的 requirements.txt。

```
cython==0.29.21
```

3. Cython 库有各种必需的 pip 依赖关系版本。例如，使用 awswrangler==2.4.0 要求 pyarrow<3.1.0, >=2.0.0，所以 pip3 尝试安装 pyarrow==3.0.0 这会导致损坏的 DAG 错误。我们建议明确指定最早的可接受版本。例如，如果指定最小值 pyarrow==2.0.0 以前 awswrangler==2.4.0 然后错误消失了，然后 requirements.txt 正确安装。最终要求应该如下所示：

```
cython==0.29.21
pyarrow==2.0.0
awswrangler==2.4.0
```

4. 探索在中指定 Python 依赖关系的方法 requirements.txt 文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

运算符

以下主题介绍了在使用 Operator 时可能收到的错误。

我使用 BigQuery 运算符收到错误

亚马逊 MWAA 不支持具有 UI 扩展的运营商。我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。

2. 解决办法是通过在 DAG 中添加要设置的行来覆盖扩展程序<operator name>.operator_extra_links = None导入问题运算符之后。例如：

```
from airflow.contrib.operators.bigquery_operator import BigQueryOperator
BigQueryOperator.operator_extra_links = None
```

3. 您可以通过将上述内容添加到插件中来对所有 DAG 使用此方法。有关示例，请参阅 [the section called “要修补的自定义插件 PythonVirtualenvOperator” \(p. 207\)](#)。

连接

以下主题描述了在使用 Apache Airflow 连接或使用其他连接时可能收到的错误AWS数据库。

我无法连接到 Snowflake

我们建议您完成以下步骤：

1. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 将以下条目添加到您的环境的 requirements.txt。

```
asn1crypto == 0.24.0
snowflake-connector-python == 1.7.2
```

3. 将以下导入添加到您的 DAG：

```
from airflow.contrib.hooks.snowflake_hook import SnowflakeHook
from airflow.contrib.operators.snowflake_operator import SnowflakeOperator
```

确保 Apache Airflow 连接对象包含以下键/值对：

1. 康恩 Id：snowflake_conn
2. Conn 类型：Snowflake
3. 主机：<my account>。 <my region if not us-west-2>.snowflake计算ing.com
4. 架构：<my schema>
5. 登录：<my user name>
6. 密码：*****
7. 端口：<port, if any>
8. 额外：

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

例如：

```
>>> 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')),
... )
```

我无法连接到 Secrets Manager

我们建议您完成以下步骤：

1. 了解如何为 Apache Airflow 连接创建密钥和变量[the section called “配置Secrets Manager” \(p. 79\)](#).
2. 了解如何使用 Apache Airflow 变量的密钥 (test-variable) 在[在中使用密钥AWS Secrets Manager对于 Apache 气流变量 \(p. 187\)](#).
3. 了解如何使用密钥进行 Apache Airflow 连接 (myconn) 在[在中使用密钥AWS Secrets Manager用于 Apache 气流连接 \(p. 189\)](#).

我无法在 '<DB-identifier-name>.cluster-id 上连接到我的 MySQL 服务器。 <region>.rds.amazonaws.com'

Amazon MWAA 的安全组和 RDS 安全组需要一个入口规则来允许进出彼此的流量。我们建议您完成以下步骤：

1. 修改 RDS 安全组以允许来自 Amazon MWAA 的 VPC 安全组的所有流量。
2. 修改 Amazon MWAA 的 VPC 安全组以允许来自 RDS 安全组的所有流量。
3. 再次重新运行任务，并通过检查 CloudWatch Logs 中的 Apache Airflow 日志来验证 SQL 查询是否成功。

Web 服务器

以下主题描述了 Apache Airflow 可能会收到的错误Web 服务器在亚马逊 MWAA 上。

我正在使用 BigQueryOperator，这导致我的网络服务器崩溃

我们建议您完成以下步骤：

1. Apache Airflow 运营商，例如BigQueryOperator和QuboleOperator其中包含operator_extra_links可能会导致 Apache Airflow 网络服务器崩溃。这些操作员试图将代码加载到您的 Web 服务器，出于安全原因，这是不允许的。我们建议通过在导入语句后添加以下代码来修补 DAG 中的运算符：

```
BigQueryOperator.operator_extra_links = None
```

2. 使用本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#) (位于 GitHub 上)。

我看到访问 Web 服务器时出现 5xx 错误

我们建议您完成以下步骤：

1. 检查 Apache Airflow 配置选项。验证您指定为 Apache Airflow 配置选项的键值对，例如AWS Secrets Manager，配置正确。要了解更多信息，请参阅 [the section called “我无法连接到 Secrets Manager” \(p. 316\)](#)。
2. 检查requirements.txt。验证你列出的 Airflow “额外” 包和其他库requirements.txt与你的 Apache Airflow 版本兼容。
3. 探索在中指定 Python 依赖关系的方法requirements.txt文件，请参阅[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

我看到“计划程序似乎没有运行”错误

如果计划程序似乎没有运行，或者几个小时前收到了最后一个“心跳”，那么你的 DAG 可能不会出现在 Apache Airflow 中，也不会计划新的任务。

我们建议您完成以下步骤：

1. 确认您的 VPC 安全组允许对端口的入站访问5432。连接到您的环境的 Amazon Aurora PostgreSQL 元数据库需要此端口。添加此规则后，给亚马逊 MWAA 几分钟时间，错误应该消失。要了解更多信息，请参阅 [the section called “您的 VPC 安全” \(p. 129\)](#)。

Note

- Aurora PostgreSQL 元数据库是[Amazon MWAA 服务架构 \(p. 2\)](#)并且在你的AWS 账户。
 - 与数据库相关的错误通常是调度程序故障的症状，而不是根本原因。
2. 如果调度程序没有运行，可能是由于许多因素造成的，例如[依赖项安装失败 \(p. 248\)](#)，或者[重载调度程序 \(p. 238\)](#)。通过查看 CloudWatch Logs 中的相应日志组，确认 DAG、插件和要求是否正常运行。要了解更多信息，请参阅 [监控和指标 \(p. 255\)](#)。

任务

以下主题描述了环境中的 Apache Airflow 任务可能会收到的错误。

我看到我的任务卡住或没完成

如果您的 Apache Airflow 任务“卡住”或未完成，我们建议您采用以下步骤：

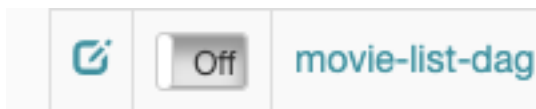
1. 可能定义了大量 DAG。减少 DAG 的数量并执行环境更新（例如更改日志级别）以强制重置。
 - a. 无论它们是否启用，气流都会解析 DAG。如果你使用的是环境容量的 50% 以上，你可能会开始压倒 Apache Airflow计划程序。这会导致大总解析时间在 CloudWatch 指标中或 CloudWatch Logs 中的长 DAG 处理时间。还有其他方法可以优化 Apache Airflow 配置，这些方法不在本指南的讨论范围内。
 - b. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅[the section called “Amahe Ahe Ahe” \(p. 238\)](#)。
2. 队列中可能有大量任务。这通常显示为处于“无”状态的大量且不断增长的任务数量，或者显示为大量处于排队的任务和/或待处理任务在 CloudWatch 中。出现此错误的原因如下：
 - a. 如果要运行的任务多于环境的运行能力，和/或在自动扩展有时间检测任务并部署其他任务之前排队的大量任务劳动力。
 - b. 如果要运行的任务多于环境的运行容量，我们建议减少DAG 同时运行的任务数量，和/或增加最低 Apache Airflow劳动力。
 - c. 如果在自动扩缩有时间检测和部署其他工作人员之前有大量的任务排队，我们建议跟踪任务部署和/或增加最低Apache Airflow劳动力。
 - d. 您可以使用[update-environment](#)中的命令AWS Command Line Interface(AWS CLI) 更改的最小或最大数量劳动力在你的环境中运行。

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-workers 10
```

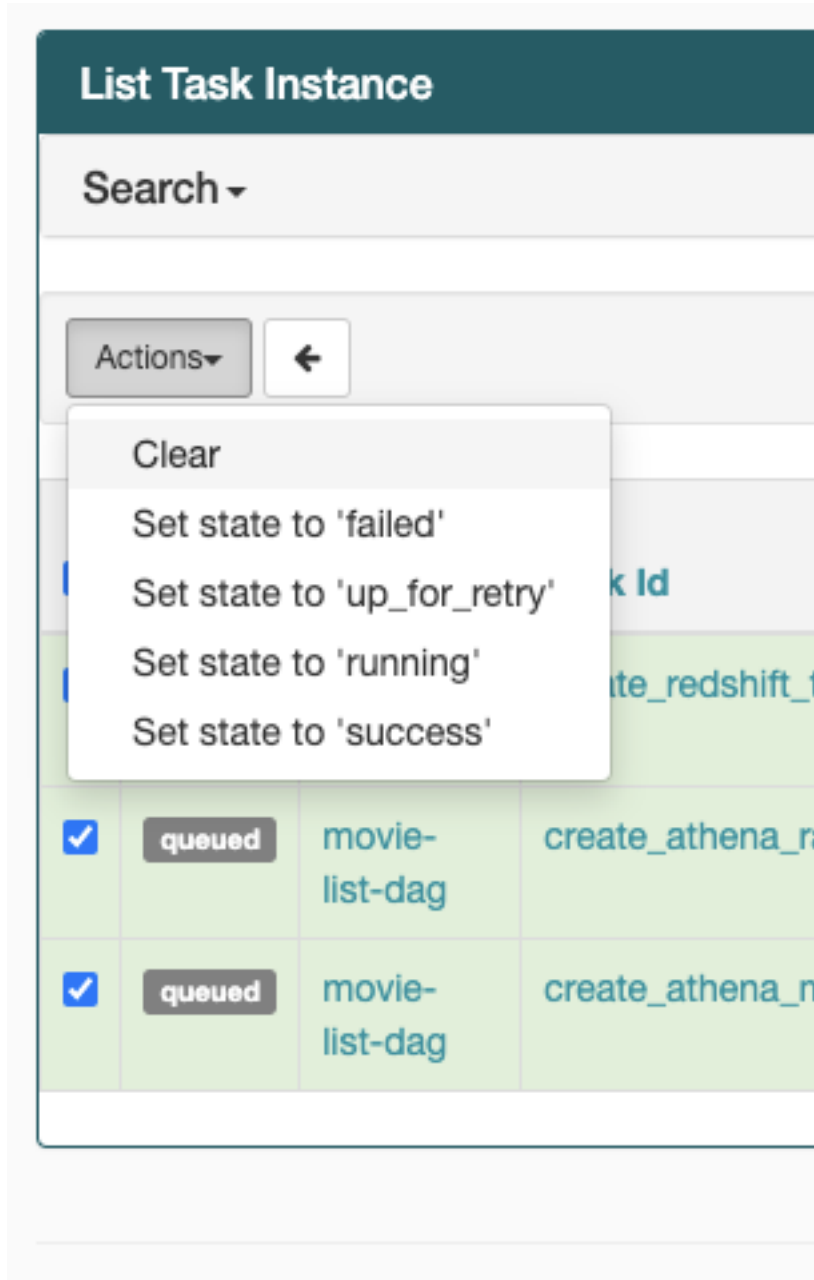
- e. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅[the section called “Amahe Ahe” \(p. 238\)](#).
3. 可能会在执行过程中删除任务，这些任务显示为任务日志，这些任务将停止，在 Apache Airflow 中没有进一步的指示 出现此错误的原因如下：
 - a. 如果有一个短暂的时刻，1) 当前任务超出当前环境容量，然后是 2) 几分钟没有执行或正在排队的任务，那么 3) 新任务正在排队。
 - b. Amazon MWAA 自动扩缩通过添加额外的工作人员来对第一种情况做出反应。在第二种情况下，它删除了额外的工作人员。一些正在排队的任务可能会导致正在移除的工作人员，并将在容器被删除后结束。
 - c. 我们建议增加环境中的最低工作人员数量。另一种选择是调整 DAG 和任务的时间，以确保不会发生这些情况。
 - d. 您还可以将最少工作线程设置为等于环境中的最大工作线程数，从而有效地禁用自动扩展。使用[update-environment](#)中的命令AWS Command Line Interface(AWS CLI) 到禁用自动扩展将最小和最大工作线程设置为相同的。

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-workers 5
```

- e. 要了解有关我们建议调整环境性能的最佳实践的更多信息，请参阅[the section called “Amahe Ahe” \(p. 238\)](#).
4. 如果您的任务陷入“正在运行”状态，您也可以清除任务或将其标记为成功或失败。这使您的环境的自动扩展组件可以缩小环境中运行的工作线程的数量。下图显示了一个搁置任务的示例。



- 选择搁置任务的圆圈，然后选择Clear (如图所示)。这允许 Amazon MWAA 缩小工作人员；否则，如果仍有排队任务，亚马逊 MWAA 将无法确定哪些 DAG 已启用或禁用，也无法缩小规模。



5. 了解有关 Apache Airflow 任务生命周期的更多信息，请访问[概念](#)中的Apache Airflow 参考指南.

CLI

以下主题介绍了在中运行 Airflow CLI 命令时可能收到的错误AWS Command Line Interface.

在 CLI 中触发 DAG 时我看到“503”错误

气流 CLI 在 Apache 气流上运行Web 服务器，它的并发性有限。通常最多可以同时运行 4 个 CLI 命令。

故障排除：创建和更新 Amazon MWAA 环境

本页上的主题包含您在创建和更新 Apache Airflow (MWAA) 的 Amazon 托管工作流 (MWAA) 环境时可能遇到的错误，以及如何解决这些错误。

目录

- [更新requirements.txt \(p. 320\)](#)
 - [我指定了我的新版本requirements.txt而且更新我的环境需要 20 多分钟 \(p. 313\)](#)
- [插件 \(p. 321\)](#)
 - [亚马逊 MWAA 是否支持实施自定义 UI？ \(p. 321\)](#)
 - [我能够在亚马逊 MWAA 本地跑步者通过插件，但是当我尝试在亚马逊 MWAA 上执行同样的操作时，我看不到我的更改或任何错误。为什么会发生这种情况？ \(p. 321\)](#)
- [创建存储桶 \(p. 321\)](#)
 - [我无法选择 S3 阻止公有访问设置选项 \(p. 321\)](#)
- [创建 环境。 \(p. 322\)](#)
 - [我试图创建一个环境，它陷入了“正在创建”状态 \(p. 322\)](#)
 - [我试图创建一个环境，但它显示状态为“创建失败” \(p. 322\)](#)
 - [我尝试选择 VPC 但收到“网络故障”错误 \(p. 322\)](#)
 - [我试图创建环境并收到服务、分区或资源“必须通过”错误 \(p. 323\)](#)
 - [我尝试创建一个环境，它将状态显示为“可用”，但是当我尝试访问 Airflow UI 时，显示“来自服务器的空回复”或“502 坏网关”错误 \(p. 323\)](#)
 - [我试图创建一个环境，我的用户名是一堆随机的角色名称 \(p. 323\)](#)
- [更新环境 \(p. 323\)](#)
 - [我尝试更改环境类，但更新失败 \(p. 323\)](#)
- [访问环境 \(p. 323\)](#)
 - [我无法访问 Apache Airflow UI \(p. 323\)](#)

更新requirements.txt

以下主题描述了在更新时可能收到的错误requirements.txt。

我指定了我的新版本requirements.txt而且更新我的环境需要 20 多分钟

如果您的环境需要二十多分钟才能安装新版本的requirements.txt文件中，环境更新失败，Amazon MWAA 正在回滚到容器映像的上一个稳定版本。

1. 检查软件包版本。我们建议始终指定特定版本 (==) 或最高版本 (>=) 对于你的 Python 依赖关系requirements.txt。
2. 检查 Apache 气流日志。如果您启用了 Apache Airflow 日志，请验证您的日志组是否在[日志组页](#)（位于 CloudWatch 控制台上）。如果您看到空白日志，最常见的原因是在编写日志的 CloudWatch 或 Amazon S3 的执行角色中缺少权限。要了解更多信息，请参阅[执行角色 \(p. 44\)](#)。
3. 选中 Apache Airflow 配置选项。如果您使用的是 Secrets Manager，请验证您指定为 Apache Airflow 配置选项的键值对是否已正确配置。要了解更多信息，请参阅[the section called “配置Secrets Manager” \(p. 79\)](#)。
4. 检查 VPC 网络配置。要了解更多信息，请参阅[the section called “环境卡住” \(p. 322\)](#)。
5. 检查执行角色权限。执行角色是AWS Identity and Access Management(IAM) 角色，具有权限策略，该策略授予 Amazon MWAA 调用其他资源的权限AWS代表您的服务（例如 Amazon

S3、CloudWatch、Amazon SQS、Amazon ECR)。您的[客户管理的密钥](#)要么[AWS拥有的钥匙](#)还需要允许访问。要了解更多信息，请参阅[执行角色 \(p. 44\)](#)。

6. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅[验证环境](#)中的脚本AWSGitHub 上的 Support 工具。

插件

以下主题介绍了在配置或更新 Apache Airflow 插件时可能遇到的问题。

亚马逊 MWAA 是否支持实施自定义 UI？

从 Apache Airflow v2.2.2 开始，亚马逊 MWAA 支持在 Apache Airflow 网络服务器上安装插件，并实现自定义 UI。如果您的亚马逊 MWAA 环境运行的是 Apache Airflow v2.0.2 或更早版本，则无法实现自定义 UI。

有关版本管理和升级现有环境的更多信息，请参阅[版本 \(p. 294\)](#)。

我能够在[亚马逊 MWAA 本地跑步者](#)通过插件，但是当我尝试在[亚马逊 MWAA](#) 上执行同样的操作时，我看不到我的更改或任何错误。为什么会发生这种情况？

亚马逊 MWAA 本地运行器将所有 Apache Airflow 组件捆绑到一个图像中，允许您应用自定义 UI 插件更改。

创建存储桶

以下主题描述了创建 Amazon S3 存储桶时可能会收到的错误。

我无法选择 S3 阻止公有访问设置选项

这些区域有：[执行角色 \(p. 44\)](#)对于您的 Amazon MWAA 环境，需要访问GetBucketPublicAccessBlock对 Amazon S3 存储桶执行操作以验证存储桶阻止公有访问。我们建议您完成以下步骤：

1. 按以下步骤操作：[向执行角色附加 JSON 策略 \(p. 44\)](#)。
2. 附加以下 JSON 策略：

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:GetBucket",
    "s3:List"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_S3_BUCKET_NAME",
    "arn:aws:s3:::YOUR_S3_BUCKET_NAME/*"
  ]
}
```

替换中的示例占位符## `_S3_BUCKET_NAME`使用您的 Amazon S3 存储桶名称，例如`mwaa ### s3 # ###`。

3. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅[验证环境](#)中的脚本AWSGitHub 上的 Support 工具。

创建 环境。

以下主题介绍了创建环境时可能收到的错误。

我试图创建一个环境，它陷入了“正在创建”状态

我们建议您完成以下步骤：

1. 使用此处查看 VPC 网络公共路由。如果您使用的是 Amazon VPC 具有 Internet 访问权限，验证以下几点：
 - 您的 Amazon VPC 已配置为允许在不同 AWS 亚马逊 MWAA 环境使用的资源，如中所定义 [the section called “关于网络” \(p. 126\)](#)。例如，您的 VPC 安全组必须允许自引用规则中的所有流量，或者可以选择指定 HTTPS 端口范围 443 和 TCP 端口范围 5432 的端口范围。
2. 使用此处查看 VPC 网络私人路由。如果您使用的是 Amazon VPC 没有 Internet 访问，验证以下几点：
 - 您的 Amazon VPC 已配置为允许在不同 AWS 适用于您的 Amazon MWAA 环境的资源，如 [the section called “关于网络” \(p. 126\)](#)。例如，你的两个私有子网必须不有一个到 NAT 网关（或 NAT 实例）的路由表，也不 Internet 网关。
3. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅 [验证环境](#) 中的脚本 AWS GitHub 上的 Support 工具。

我试图创建一个环境，但它显示状态为“创建失败”

我们建议您完成以下步骤：

1. 检查 VPC 网络配置。要了解更多信息，请参阅 [the section called “环境卡住” \(p. 322\)](#)。
2. 检查用户权限。在创建环境之前，Amazon MWAA 会根据用户的凭据执行模拟运行。您的 AWS 账户可能没有权限 AWS Identity and Access Management (IAM) 为环境创建一些资源。例如，如果您选择了私有网络 Apache Airflow 访问模式，你 AWS 您的管理员必须授予账户访问 [亚马逊 WaafL 控制台完全访问 \(p. 33\)](#) 适用于您的环境的访问控制策略，允许您的账户创建 VPC 终端节点。
3. 检查执行角色权限。执行角色是 AWS Identity and Access Management (IAM) 角色，具有权限策略，该策略授予 Amazon MWAA 调用其他资源的权限 AWS 代表您的服务（例如 Amazon S3、CloudWatch、Amazon SQS、Amazon ECR）。您的 [客户管理的密钥](#) 要么 [AWS 拥有的钥匙](#) 还需要允许访问。要了解更多信息，请参阅 [执行角色 \(p. 44\)](#)。
4. 检查 Apache 气流日志。如果您启用了 Apache Airflow 日志，请验证您的日志组是否在 [日志组页](#)（位于 CloudWatch 控制台上）。如果您看到空白日志，最常见的原因是在编写日志的 CloudWatch 或 Amazon S3 的执行角色中缺少权限。要了解更多信息，请参阅 [执行角色 \(p. 44\)](#)。
5. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅 [验证环境](#) 中的脚本 AWS GitHub 上的 Support 工具。
6. 如果您使用的是 Amazon VPC 没有互联网访问，请确保您已创建 Amazon S3 网关终端节点，并向 Amazon ECR 授予访问 Amazon S3 所需的最低权限。要了解有关创建 Amazon S3 网关终端节点的更多信息，请参阅以下内容：
 - [创建没有互联网访问权限的 Amazon VPC 网络 \(p. 23\)](#)
 - [创建 Amazon S3 网关终端节点](#) 中的 Amazon Elastic Container Registry 用户指

我尝试选择 VPC 但收到“网络故障”错误

我们建议您完成以下步骤：

- 如果在创建环境时尝试选择 Amazon VPC 时看到“网络故障”错误，请关闭所有正在运行的浏览器内代理，然后重试。

我试图创建环境并收到服务、分区或资源“必须通过”错误

我们建议您完成以下步骤：

- 您可能会收到此错误，因为您为 Amazon S3 存储桶指定的 URI 在 URI 末尾包含“/”。我们建议删除路径中的“/”。值应采用以下格式：

```
s3://your-bucket-name
```

我尝试创建一个环境，它将状态显示为“可用”，但是当我尝试访问 Airflow UI 时，显示“来自服务器的空回复”或“502 坏网关”错误

我们建议您完成以下步骤：

1. 检查 VPC 安全组配置。要了解更多信息，请参阅 [the section called “环境卡住” \(p. 322\)](#)。
2. 确认您在 requirements.txt 与您在亚马逊 MWAA 上运行的 Apache Airflow 版本相对应。要了解更多信息，请参阅 [安装 Python 的 \(p. 117\)](#)。
3. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅[验证环境](#)中的脚本AWSGitHub 上的 Support 工具。

我试图创建一个环境，我的用户名是一堆随机的角色名称

- Apache Airflow 用户名最大长度为 64 个字符。如果您的AWS Identity and Access Management(IAM)角色超过了此长度，使用哈希算法来缩小角色，同时保持唯一性。

更新环境

以下主题介绍了更新环境时可能收到的错误。

我尝试更改环境类，但更新失败

如果您将环境更新为不同的环境类（例如更改mw1.medium到mw1.small），并且更新环境的请求失败，则环境状态将进入UPDATE_FAILED状态和环境会回滚到环境的先前稳定版本，并按照环境的以前的稳定版本计费。

我们建议您完成以下步骤：

1. 使用在本地测试 DAG、自定义插件和 Python 依赖项[aws-mwaa 本地跑步者](#)（位于 GitHub 上）。
2. 要运行故障排除脚本来检查 Amazon VPC 网络设置和配置的 Amazon MWAA 环境，请参阅[验证环境](#)中的脚本AWSGitHub 上的 Support 工具。

访问环境

以下主题介绍了访问环境时可能会收到的错误。

我无法访问 Apache Airflow UI

我们建议您完成以下步骤：

1. 检查用户权限。您可能没有被授予访问允许您查看 Apache Airflow UI 的权限策略的访问权限。要了解更多信息，请参阅 [the section called “访问Amazon MWAA 环境” \(p. 32\)](#)。

2. 检查网络访问。这可能是因为你选择了私有网络访问模式。如果您的 Apache Airflow UI 采用以下格式 387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-west-2.airflow.amazonaws.com，这意味着您使用的是私人路由为了您的 Apache 气流Web 服务器。您可以将 Apache Airflow 访问模式更新为公共网访问模式，或者创建一种机制来访问 Apache Airflow 的 VPC 终端节点Web 服务器。要了解更多信息，请参阅[“the section called “管理对 VPC 终端节点的访问” \(p. 132\)”](#)。

故障排除：CloudWatch 日志和 CloudTrail 错误

此页面上的主题包含亚马逊的解决方案 CloudWatch 日志和AWS CloudTrail您可能会在Amazon Managed Workflows for Apache Airflow (MWAA) 环境中

目录

- [日志 \(p. 324\)](#)
 - [我看不到我的任务日志，或者我收到了“从 Cloudwatch log_group 读取远程日志”错误 \(p. 324\)](#)
 - [任务失败，没有任何日志 \(p. 324\)](#)
 - [我看见了'ResourceAlreadyExistsException'错误消息 CloudTrail \(p. 325\)](#)
 - [我在中看到“请求无效”错误 CloudTrail \(p. 325\)](#)
 - [我看到“找不到 64 位 Oracle Client 库：” libcIntsh.so：无法打开共享目标文件：Apache Airflow 日志中没有“这样的文件或目录” \(p. 326\)](#)
 - [我在我的调度器日志中看到 psycpg2“服务器意外关闭了连接” \(p. 326\)](#)
 - [我在 DAG 处理日志中看到“Executor 报告任务实例 %s 已完成 \(%s\)，尽管任务显示为 %s” \(p. 326\)](#)
 - [我看见 '无法从 log_group 读取远程日志：airflow-*{*EnvironmentName}-Task log_stream: *{*DAG_ID} /*{*n}.log。”在我的任务日志中 \(p. 327\)](#)

日志

以下主题描述了您在查看 Apache Airflow 日志时可能收到的错误。

我看不到我的任务日志，或者我收到了“从 Cloudwatch log_group 读取远程日志”错误

如果您看到空白日志，或者在查看时出现关注错误任务日志在 Airflow UI 中：

```
*** Reading remote log from Cloudwatch log_group: airflow-{environmentName}-Task
log_stream: {DAG_ID}/{TASK_ID}/{time}/{n}.log.Could not read remote logs from log_group:
airflow-{environmentName}-Task log_stream: {DAG_ID}/{TASK_ID}/{time}/{n}.log.
```

- 我们建议您完成以下步骤：
 - a. 验证您是否在环境的 INFO 级别启用了任务日志。要了解更多信息，请参阅[在亚马逊上查看气流日志 CloudWatch \(p. 257\)](#)。
 - b. 验证您的操作员是否有相应的 Python 库可供加载。你可以尝试取消导入，直到找到导致问题的那个。

任务失败，没有任何日志

如果工作流程中的任务失败，并且您找不到失败任务的任何日志，请检查您是否正在设置queue默认参数中的参数，如下所示。


```
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'] }}"
    )
```

要解决此问题，请删除queue从你的代码中，然后再次调用 DAG。

我看见了'ResourceAlreadyExistsException'错误消息 CloudTrail

```
"errorCode": "ResourceAlreadyExistsException",
"errorMessage": "The specified log stream already exists",
"requestParameters": {
    "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
    "logStreamName": "scheduler_cross-account-eks.py.log"
}
```

某些 Python 要求，例如apache-airflow-backport-providers-amazon回滚watchtower亚马逊 MWAA 用来与之通信的库 CloudWatch 改为较旧版本。我们建议您完成以下步骤：

- 将以下库添加到您的requirements.txt

```
watchtower==1.0.6
```

我在中看到“请求无效”错误 CloudTrail

```
Invalid request provided: Provided role does not have sufficient permissions for s3
location airflow-xxx-xxx/dags
```

如果您正在创建 Amazon MWAA 环境和 Amazon S3 存储桶AWS CloudFormation模板，你需要添加一个DependsOn你的AWS CloudFormation模板。这两种资源 (MWAA Environ和MWAA 执行政策) 在... 中有依赖关系AWS CloudFormation. 我们建议您完成以下步骤：

- 添加以下内容**DependsOn**对你的语句AWS CloudFormation模板。

```
...
    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:
...

```

有关示例，请参阅 [Workflows for Apache Airflow \(MWAA\) \(p. 5\)](#)。

我看到“找不到 64 位 Oracle Client 库：” libclntsh.so：无法打开共享目标文件：Apache Airflow 日志中没有“这样的文件或目录”

- 我们建议您完成以下步骤：
 - 如果你使用的是 Apache Airflow v2，请添加 `core.lazy_load_plugins : False` 作为 Apache Airflow 配置选项。要了解更多信息，请参阅 [使用配置选项在 2 中加载插件 \(p. 91\)](#)。

我在我的调度器日志中看到 psycopg2 “服务器意外关闭了连接”

如果您看到与以下内容相似的错误，则您的 Apache Airflows 计划程序可能已经耗尽了资源。

```
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.

```

我们建议您完成以下步骤：

- 考虑升级到 Apache Airflow v2.0.2，它允许你最多指定 5Schedule。

我在 DAG 处理日志中看到“Executor 报告任务实例 %s 已完成 (%s)，尽管任务显示为 %s”

如果您看到类似以下内容的错误，则说明您的长时间运行的任务可能已达到 Amazon MWAA 上的任务时间限制。Amazon MWAA 对任一 Airflow 任务的限制均为 12 小时，以防止任务卡在队列中并阻止自动扩展等活动。

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info: %s)
Was the task killed externally

```

我们建议您完成以下步骤：

- 考虑将任务分解为多个运行时间较短的任务。Airflow 通常采用操作员异步的模型。它调用外部系统上的活动，然后 Apache Airflow Sensors 进行轮询以查看其何时完成。如果传感器出现故障，可以在不影响操作员功能的情况下安全地重试。

我看见 '无法从 log_group 读取远程日志 : airflow- {*EnvironmentName}-Task log_stream: * {*DAG_ID} /* {*n} .log。 ” 在我的任务日志中

如果您看到类似以下内容的错误，则您的环境的执行角色可能不包含为任务日志创建日志流的权限策略。

```
Could not read remote logs from log_group: airflow-*{environmentName}-Task log_stream:*  
{*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

我们建议您完成以下步骤：

- 使用位于的示例策略之一修改环境的执行角色[the section called “执行角色” \(p. 44\)](#).

您可能还在requirements.txt与你的 Apache Airflow 版本不兼容的文件。例如，如果你使用的是 Apache Airflow v2.0.2，你可能已经指定了一个软件包，比如[apache-airflow-providers-databricks](#)封装，仅兼容 Airflow 2.1+。

我们建议您完成以下步骤：

1. 如果你使用Apache Airflows vor Airflows vflows vrequirements.txt文件并添加apache-airflow[databricks]. 这将安装与 Apache Airflow v2.0.2 兼容的 Databricks 软件包的正确版本。
2. 使用以下命令在本地测试你的 DAG、自定义插件和 Python 依赖关系[aws-mwaa-local-runner](#)上 GitHub.

Amazon MWA 文档历史记录

下表介绍了自 2020 年 11 月以来对 Amazon MWAA) 服务文档的重要补充。要接收有关文档更新的通知，您可以订阅 RSS 源。

变更	说明	日期
更新了主题 (p. 328)	更新了客户托管的 IAM 政策，授予用户对 Amazon MWAA 的完全控制台和 API 访问权限。此更新描述了为什么您必须提供权限 <code>iam:PassRole</code> 才能允许用户将角色传递给 Amazon MWAA。亚马逊 MWAA 使用这些权限代表用户执行操作。 the section called “访问 Amazon MWAA 环境” (p. 32)	2023 年 4 月 12 日
新指南 (p. 328)	更新了有关配置 AWS Secrets Manager 为 Amazon MWAA 后端的主题以提供有关使用查询模式的指导。使用查询模式缩小了 Apache Airflow 搜索的秘密范围，减少亚马逊 MWAA 为检索连接和变量而向 Secrets Manager 进行的 API 调用次数。这降低了与使用 Secrets Manager 作为后端相关的成本。 将 Secrets Manager 后端创建为 Apache Airflow 配置选项 (p. 80)	2023 年 4 月 12 日
Apache Airflow (p. 328)	亚马逊 MWAA 现在支持 Apache Airflow v2.5.1。此更新包括有关更新的提供商包的信息，以及有关在亚马逊 MWAA 上使用 Apache Airflow v2.5.1 的详细信息， 版本 (p. 294) the section called “Apache Airflow v2.5.1 连接的提供者软件包” (p. 74)	2023 年 4 月 11 日
新主题和用例 (p. 328)	添加了关于在 Amazon MWAA 环境中使用启动脚本的新主题。本主题介绍为现有环境配置启动脚本、使用该脚本安装 Linux 运行时以及设置环境变量。 the section called “使用启动脚本” (p. 97)	2023 年 4 月 3 日

更新了关于私有 Web 服务器访问的章节 (p. 328)	更新了以下有关专用 Web 服务器访问的主题。此更新阐明，在具有私有 Web 服务器访问权限的环境中，必须使用 Python wheel 存档 (.whl) 来打包和安装依赖关系。 • 专用 Web 服务器访问模式 (p. 54)	2023 年 2 月 24 日
添加了有关过时的 Apache Airflow 版本的信息 (p. 328)	更新了 版本 主题，其中包含有关亚马逊 MWAA 如何设法弃用 Apache Airflow 版本的新信息。删除了关于升级到新版本的 Apache Airflow 的部分，以及描述了 Apache Airflow v1 和 Apache Airflow v2 之间变化的部分。有关迁移到新版本的 Apache Airflow 的更多信息，请参阅 亚马逊 MWAA 迁移指南 。 • the section called “Apache Airflow 过时” (p. 295) • the section called “AAAAAAAAAAAAAAA” (p. 296)	2023 年 2 月 17 日
修复了亚马逊 MWAA 容器指标中的问题 (p. 328)	更新了容器指标主题，并删除了 Cluster 维度下不存在的一组错误指标。添加了另外一节，描述如何通过绘制 AdditionalWorker 组件 CPU Utilization 或 Memory Utilization 指标的图形并将统计类型设置为，来评估环境在给定时间使用的额外工作人员数量 Sample Count。 • the section called “评估其他工作实例的数量” (p. 276)	2023 年 1 月 20 日
Apache Airflow (p. 328)	亚马逊 MWAA 现在支持 Apache Airflow v2.4.3。此更新包括有关更新的提供商包的信息、有关在亚马逊 MWAA 上使用 Apache Airflow v2.4.3 的详细信息，以及有关亚马逊 MWAA 上每个 Apache Airflow 版本支持哪些功能的合并信息。 • 版本 (p. 294) • the section called “Apache Airflow v2.4.3 连接的提供者软件包” (p. 74)	2023 年 1 月 5 日

有关服务相关角色的更新主题 (p. 328)	有关 Amazon MWAA 用来代表您创建和管理AWS资源的服务关联角色的最新信息，包括有关在不再需要服务相关角色时如何将其删除的信息。这包括更新的服务相关角色权限政策，该策略允许 Amazon MWAA 在AWS/MWAA命名空间下发布其他 CloudWatch 指标。	2022 年 11 月 18 日
	• the section called “服务相关角色” (p. 40)	
关于服务指标的新话题 (p. 328)	添加了描述 Amazon MWAA 在AWS/MWAA命名空间下发布的服务指标的新主题。这些指标包括 Amazon ECS 集群指标调度程序、工作程序和 Web 服务器、允许 Amazon MWAA 解耦调度程序和工作程序的队列的 Amazon SQS 指标，以及元数据数据库的 Amazon RDS 指标。	2022 年 11 月 18 日
	• the section called “容器、队列和数据库指标” (p. 274)	
新主题 (p. 328)	添加了关于修改限制文件的新指南，以指定在您的 Amazon MWAA 环境中使用的提供程序包的新版本。	2022 年 11 月 18 日
	• the section called “指定更新的提供程序包” (p. 75)	
更新了常见问题条目 (p. 328)	更新了与亚马逊 MWAA 的 HIPAA 资格相关的信息。	2022 年 11 月 15 日
	• the section called “HIPAA 合规性” (p. 300)	
新主题 (p. 328)	添加了有关在 Amazon MWAA 执行角色信任策略中使用 <code>aws:SourceArn</code> 全局条件上下文密钥的新主题，以防止副手混淆跨服务。	2022 年 10 月 21 日
	• the section called “跨服务混淆代理问题防范” (p. 51)	
新的示例代码 (p. 328)	添加了更新的指令和 DAG 代码示例，用于将自定义操作系统级指标写入 CloudWatch。	2022 年 9 月 13 日
	• the section called “使用 DAG 写入自定义指标” (p. 179)	

新的示例代码 (p. 328)	添加了更新的指令和新的AWS Lambda Python 代码示例，该示例用于检索 Apache Airflow CLI 令牌，然后在指定的 Amazon MWAA 环境中调用 DAG。 • the section called “使用 Lambda 调用 DAG” (p. 211)	2022 年 9 月 12 日
新的架构图 (p. 328)	添加了新的架构图，演示了带有公共和私有 Web 服务器的 Amazon MWAA 环境。 • the section called “Apache Airflow” (p. 52)	2022 年 9 月 12 日
新的示例代码 (p. 328)	添加了更新的指令和新的 DAG 代码示例，该示例用于检索 Apache Airflow CLI 令牌，然后在不同的 Amazon MWAA 环境中调用另一个 DAG。 • the section called “在不同的环境中调用 DAG” (p. 213)	2022 年 8 月 16 日
新的示例代码 (p. 328)	添加了更新的指令和新的 DAG，用于查询环境的 Aurora PostgreSQL 以获取元数据信息，将结果写入 CSV 文件并将文件存储在 Amazon S3 中。 • the section called “将环境元数据导出到 Amazon S3” (p. 185)	2022 年 8 月 12 日
新的示例代码 (p. 328)	添加了更新的指令和新的 DAG，可在运行时刷新AWS CodeArtifact令牌并将结果存储在 Amazon S3 中。 • the section called “RefreshAWS CodeArtifact运行时的令牌” (p. 200)	2022 年 8 月 3 日
新的示例代码 (p. 328)	为在 Amazon MWAAECSOperator 中使用添加了更新的说明和 DAG 代码示例。 • the section called “使用 ECSOperator” (p. 228)	2022 年 7 月 26 日
新的示例代码 (p. 328)	为在 Amazon MWAASSHOperator 中使用添加了更新的说明和 DAG 代码示例。 • the section called “使用 SSHOperator” (p. 174)	2022 年 7 月 15 日

新的示例代码 (p. 328)	添加了在亚马逊 MWAA 上使用 dbt Postgres 的新指令和 DAG 代码示例。 • the section called “在亚马逊 MWAA 上使用 dbt” (p. 234)	2022 年 6 月 17 日
新主题和用例 (p. 328)	添加了新的指令和 DAG 代码示例，用于在具有公共和私有访问权限的 Amazon MWAA 环境中使用 Python 轮子文件安装依赖关系。 • 使用 Python 轮子管理依赖关系 (p. 250)	2022 年 5 月 13 日
新主题和用例 (p. 328)	添加了关于选择亚马逊 MWAA 发送到哪些 Apache Airflow 指标的新指南 CloudWatch。 • 选择报告哪些 Apache Airflow 指标	2022 年 4 月 19 日
新指南 (p. 328)	Amazon MWAA 提供了迁移指南，用于从自我管理部署以及现有 Amazon MWAA 环境中迁移 Apache Airflow 工作流程。 • 亚马逊 MWAA 迁移指南	2022 年 3 月 7 日
新主题和用例 (p. 328)	添加了使用 Apache Airflow 的新安全最佳实践，包括检测 Apache Airflow 用户权限更改的解决方案。 • the section called “Apache Airflow 中的安全最佳实” (p. 293)	2022 年 2 月 18 日
新的示例代码 (p. 328)	添加了使用 Pendulum 创建时区感知型 DAG 的新代码示例，并阐明了如何使用自定义插件更改创建 Apache Airflow 日志的时区。 • the section called “更改 DAG 的时区” (p. 198)	2022 年 2 月 11 日

[Apache Airflow v2.2.2 发布 \(p. 328\)](#)

Amazon Managed Workflows for Apache Airflow (MWAA) 现在支持 Amazon M 从 v2.2 开始，亚马逊 MWAA 将直接在 Apache Airflow Web 服务器上安装 Python 包和自定义插件，从而使您能够更灵活地管理环境。有关更多信息，请参阅下列内容。

2022 年 1 月 27 日

- [Amazon Managed Workflows for Apache Airflow v2.2.2 连接的提供者软件包 \(p. 294\)](#).
- [the section called “Apache Airflow v2.2.2 连接的提供者软件包” \(p. 75\)](#).
- [Apache Airflow v2.2.2 更新日志在 Apache Airflow 文档网站上发布。](#)

[新教程 \(p. 328\)](#)

添加了一个新教程，演示如何创建新的自定义 Apache Airflow 角色，并将该角色分配给从 IAM 映射的 Apache Airflow 用户，以限制用户对指定 DAG 子集的访问权限。

2021 年 12 月 8 日

- [the section called “教程：限制用户访问 DAG 的子集” \(p. 169\)](#)

[修复 \(p. 328\)](#)

修复了设置值以优化 CPU 使用率的最佳实践建议。scheduler.min_file_process_interval 添加了一个 IAM 策略示例，该示例授予对执行角色中的 Secrets Manager 资源的访问权限。添加了有关使用密钥管理器条件密钥的疑难解答主题。

2021 年 11 月 22 日

- [性能调整调度程序解析 DAG 的方式 \(p. 241\)](#)
- [向亚马逊 MWAA 提供访问密钥 Secrets Manager 密钥的权限 \(p. 79\)](#)
- [在 Amazon MWAA 执行角色中为密钥管理器配置条件密钥 \(p. 306\)](#)

[新的示例代码 \(p. 328\)](#)

添加了以下新的代码示例，用于修改使用自定义插件处理 DAG 的时区，以及用于从 bash 操作员中调用 dags backfill Apache Airflow CLI 命令的新故障排除主题。

2021 年 11 月 1 日

- [the section called “更改 DAG 的时区” \(p. 198\)](#)
- [使用 bash 运算符回填 CLI 命令 \(p. 311\)](#)

修复 (p. 328)	修复了 Amazon ECS 操作员代码示例中的问题，并阐明了 Amazon MWAA 执行角色所需的额外权限，以允许环境访问日志中的 Amazon ECS 任务 CloudWatch 日志组。 • 亚马逊 ECS 操作员权限 (p. 228) 。	2021 年 10 月 26 日
新的示例代码 (p. 328)	添加了新的代码示例，用于查询 Aurora PostgreSQL 数据库以获取与 DAG 运行相关的信息，并将结果写入存储在 Amazon S3 上的 CSV 文件中。 • the section called “将环境元数据导出到 Amazon S3” (p. 185) 。	2021 年 10 月 1 日
修复 (p. 328)	更正了有关 Amazon MWAA 如何自动将新的和已更改的对象从您的目标 Amazon S3 存储桶同步到您的调度程序和工作程序的信息。 • DAG 文件夹的工作原理 (p. 104) 。	2021 年 10 月 1 日
现在支持 (p. 328)	亚马逊 MWAA 现在支持 Apache Airflow 2.0+ 的其他提供商套餐。要了解有关支持的软件包的更多信息，请参阅以下内容： • the section called “Apache Airflow v2.0.2” (p. 75) 。	2021 年 9 月 24 日
新命令和程序 (p. 328)	添加了更多指南和 AWS CLI 命令示例，用于在不访问互联网的情况下使用 Amazon VPC 时创建 Amazon S3 网关终端节点： • 创建没有互联网访问权限的亚马逊 VPC 网络 (p. 23) 。	2021 年 9 月 24 日
新主题和用例 (p. 328)	注意以下更改： • 在中添加了使用亚马逊弹性容器服务操作符的新代码示例 the section called “使用 ECSOperator” (p. 228) 。 • 为在中配置 Apache Airflow 插件时出现的问题添加了新的故障排除主题 the section called “插件” (p. 321) 。	2021 年 9 月 19 日

335

[新主题和用例 \(p. 328\)](#)

注意以下更改：

2021 年 8 月 13 日

- 向... 添加了 `celery.sync_parallelism` 用例 [亚马逊 MWAA 上的 Apache Airflow 的性能调整 \(p. 238\)](#)。
- 向配额页面添加了服务终端节点并将名称更改为 [Amazon Managed Workflows by Apache Airflow \(MWAA\) 服务终端节点和配额 \(p. 297\)](#)。
- 根据用户反馈阐明了网络先决条件 [开始使用 Amazon Managed Workflows of Apache Airflow \(MWAA\) \(p. 15\)](#)。
- 已移至 `dags list-runs` 中不支持的 Airflow CLI 命令 [Apache Airflow Air \(p. 66\)](#)。 `dags next-execution`

[新的示例代码 \(p. 328\)](#)

注意以下更改：

2021 年 8 月 13 日

- 添加了 bash 示例来设置、获取或删除 Apache Airflow v2.0.2 变量 [Apache Airflow Air \(p. 66\)](#)。
- 添加了 Apache Airflow v2.0.2 依赖关系和 Airflow [通过 Amazon MWAA 与 Amazon RDS for Microsoft SQL Server \(p. 215\)](#) 连接示例

[修复 \(p. 328\)](#)

注意以下更改：

2021 年 8 月 13 日

- 修复了基于用户反馈的 Python 代码示例 [使用创建 SSH 连接 SSHOperator \(p. 174\)](#)。

[新主题和用例 \(p. 328\)](#)

注意以下更改：

2021 年 8 月 6 日

- 已移 `variables set` 至中支持的 Airflow CLI 命令 [Apache Airflow Air \(p. 66\)](#)。
- 从 Airflow 版本页面将 v2.0.2 中的更改摘要添加到“[安装 Python 的 \(p. 117\)](#)基于用户反馈”。
- 从 Airflow 版本页面将 v2.0.2 中的更改摘要添加到“[Apache Airflow Air \(p. 66\)](#)基于用户反馈”。
- 从 Airflow 版本页面将 v2.0.2 中的更改摘要添加到“[连接类型概览 \(p. 76\)](#)基于用户反馈”。
- 从 Airflow 版本页面将 v2.0.2 中的更改摘要添加到“[安装自定义插件 \(p. 106\)](#)基于用户反馈”。
- 从 Airflow 版本页面将 v2.0.2 中的更改摘要添加到“[添加或更新 DAG \(p. 103\)](#)基于用户反馈”。

[新的示例代码 \(p. 328\)](#)

注意以下更改：

2021 年 8 月 6 日

- 向 Apache Airflow v2.0.2 添加了示例代码[使用 DAG 在 CLI 中导入变量 \(p. 172\)](#)。
- 向 Apache Airflow v2.0.2 添加了示例代码[使用 Lambda 函数调用 DAG \(p. 211\)](#)。

[新主题和用例 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 29 日

- 添加了“我在 Airflow UI 中看不到我的连接”的疑难解答主题，网址为[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了亚马逊 MWAA 支持的亚马逊 VPC 列表[关于 Amazon MWAA 上的联网 \(p. 126\)](#)。

[修复 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 29 日

- 修复了基于用户反馈的 Python 代码示例，以便在上打印 Web 登录令牌[创建 Apache Airflow Web 登录令牌 \(p. 59\)](#)。
- 修复了基于用户反馈的 Snowflake 连接主题，以便在的仓库参数中使用单引号[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。

[已删除或移动的主题 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 23 日

- 重构了现有页面，使其包含中的所有监控和指标文档页面[亚马逊Managed Workflows for Apache Airflow \(MWAA\) \(p. 255\)](#)。
- 移中的 Apache Airflow v2 环境指标 [CloudWatch \(p. 264\)](#)至监控和指标导航菜单。
- 移中的 Apache Airflow v1 环境指标 [CloudWatch \(p. 270\)](#)至监控和指标导航菜单。

[新指南 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 23 日

- 已创建[安装在亚马逊 MWAA 环境中的 Apache Airflow 提供商软件包 \(p. 73\)](#)。
- 已创建[亚马逊 MWAA 上的监控概述 \(p. 255\)](#)。
- 已创建[查看审核日志AWS CloudTrail \(p. 256\)](#)。
- 已创建[在亚马逊上查看气流日志CloudWatch \(p. 257\)](#)。

[修复 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 23 日

- 修复了基于用户反馈的 Python 代码示例，以按正确顺序生成 Airflow 连接字符串，并在中添加了端口参数[使用AWS Secrets Manager密钥配置 Apache Airflow 连接 \(p. 79\)](#)。
- 添加了根据用户反馈在本地安装解压缩包的步骤[使用 Oracle 创建自定义插件 \(p. 192\)](#)。

[新主题和用例 \(p. 328\)](#)

注意以下更改：

2021 年 7 月 16 日

- 为AWS DMS 操作员添加了主题，网址为[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 向中添加了针对远程日志错误的故障排除主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 已移variables set至中不支持的 Airflow CLI 命令[Apache Airflow Air \(p. 66\)](#)。

新主题和用例 (p. 328)	注意以下更改： • 添加了根据用户反馈创建 requirements.txt 文件的顺序步骤 安装 Python 的 (p. 117)。 • 添加了根据用户反馈创建 plugins.zip 文件的顺序步骤 安装自定义插件 (p. 106)。 • 在 Amazon Apache Afrowlow (MWAA) API 参考指南中的 API 参考指南的用户指南 中添加了交叉引用链接。 • 添加了关于为什么插件未显示在 Airflow 2.0 管理员 > 插件菜单中的主题，网址为 亚马逊 MWAA 常见问题 (p. 298)。	2021 年 7 月 9 日
新指南 (p. 328)	注意以下更改： • 已创建 删除 Amazon S3 上的文件 (p. 122)。	2021 年 7 月 9 日
新主题和用例 (p. 328)	添加以下更改： • 在上添加了支持值的列表 使用客户管理的密钥进行加密 (p. 281)。 • 根据中的用户反馈更新并澄清了私有 repo URL 的示例在 requirements.txt 中管理 Python 依赖关系 (p. 248)。	2021 年 7 月 2 日
新的示例代码 (p. 328)	添加以下更改： • 在 Apache Airflow v1.10.12 中添加了使用私钥进行 SSH 连接 AWS Secrets Manager 的示例代码 使用创建 SSH 连接 SSHOperator (p. 174)。	2021 年 7 月 2 日
新主题和用例 (p. 328)	添加以下更改： • 已将 FinishedTaskInstances 指标添加 StartedTaskInstances 到中的 Apache Airflow v2 环境指标 CloudWatch (p. 264)。	2021 年 6 月 25 日
新的示例代码 (p. 328)	添加以下更改： • 在以下位置添加了 Apache Airflow v2.0.2 示例代码 将适用于 Amazon MWAA (p. 220)。	2021 年 6 月 25 日
新指南 (p. 328)	添加以下更改： • 已创建 亚马逊 MWAA 上的 Apache Afriflow 的性能调整 (p. 238)。	2021 年 6 月 25 日

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 6 月 18 日

- 在支持的 Apache Airflow v2.0.2 CLI 命令中添加 `connections add` 和 [Apache Airflow Air \(p. 66\)](#)。 `connections delete`
- 补充说，中可用的最新版本 AWS CloudFormation 是 Apache Airflow v2.0.2 [Workflows for Apache Airflow \(MWAA\) \(p. 5\)](#)。
- 添加了关于在 Apache Airflow Workers 上存储临时数据的问题 [亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 将“执行器报告任务实例 %s 已完成”错误的主题添加到 [Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了“服务器意外关闭了连接”日志的主题 [Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了在堡垒主机的 SSH 隧道上运行 CLI 命令的示例 [创建 Apache 气流 CLI 令牌 \(p. 61\)](#)。
- 为随机生成的用户名添加了主题 [Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 在中添加了有关在 CLI 中运行 DAG 时出现 503 错误的主题 [Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 为 Apache Airflow v2.0.2 中的自定义插件添加了主题，这些插件需要 Airflow 配置选项 `core_lazy_load_plugins` : `False` 才能在每个 Airflow 进程开始时加载插件，才能将版本的默认设置改为 [在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。
- 为 Apache Airflow v2.0.2 插件添加 [使用 Apache Hive 和 Hadoop 创建自定义插件 \(p. 204\)](#) 了 Apache Airflow 配置选项步骤，
- 为 Apache Airflow v2.0.2 插件添加 [创建生成运行时环境变量的自定义插件 \(p. 196\)](#) 了 Apache Airflow 配置选项步骤，
- 为 Apache Airflow v2.0.2 插件添加 [为 Apache](#)

	Airflow 创建自定义插件 PythonVirtualenvOperator (p. 207) 了 Apache Airflow 配置选项步骤，	
	为 Apache Airflow v2.0.2 插件添加使用 Oracle 创建自定义插件 (p. 192)了 Apache Airflow 配置选项步骤，	
新的示例代码 (p. 328)	添加以下更改：	2021 年 6 月 18 日
	在上添加了 Apache Airflow Snowflake 连接的示例代码在使用密钥AWS Secrets Manager用于 Apache Airflow 雪花连接 (p. 177)。	
新主题和用例 (p. 328)	添加以下更改：	2021 年 6 月 2 日
	- 在。中添加了服务器端加密指南为Amazon MWAA 创建 Amazon S3 存储桶 (p. 16)。 - 将 Apache Airflow v2.0.2 的秘密后端添加到使用AWS Secrets Manager密钥配置 Apache Airflow 连接 (p. 79)。 - 为 Apache Airflow Workers 配额增加请求添加了问题亚马逊 MWAA 常见问题 (p. 298)。 - 添加了使用哪些指标来确定是否将 Apache Airflow Workers 扩展到的问题亚马逊 MWAA 常见问题 (p. 298)。 - 在中添加了有关创建自定义指标 CloudWatch 的问题亚马逊 MWAA 常见问题 (p. 298)。 - 添加了为内有私有路由的 VPC 的 Amazon S3 VPC 接口终端节点启用私有 IP 地址的步骤使用私有路由在 Amazon VPC 中创建所需的 VPC 服务终端节点 (p. 136)。 - 添加了使用本地端口转发设置 SSH 隧道的选项教程：使用 Linux 堡垒主机配置专用网络访问权限 (p. 162)。	
新的示例代码 (p. 328)	添加以下更改：	2021 年 6 月 2 日
	添加了 DAG 的示例代码，该代码用于查询 Amazon Aurora PostgreSQL 元数据数据库并将自定义指标发布到亚马逊，网 CloudWatch 址为使用 DAG 写入自定义指标 CloudWatch (p. 179)。	

[新指南 \(p. 328\)](#)

添加以下更改：

2021 年 6 月 2 日

- 创建了关于如何在 Apache Airflow UI 中互换使用连接模板的指南[连接类型概览 \(p. 76\)](#)。

[修复 \(p. 328\)](#)

添加以下更改：

2021 年 6 月 2 日

- 在选项三：创建没有互联网访问权限的 VPC 网络中，将 Apache Airflow VPC 终端节点添加到AWS CloudFormation模板中[创建 VPC 网络 \(p. 18\)](#)。

[Apache Airflow v2.0.2 发布 \(p. 328\)](#)

Apache Airflow v2.2 正式可用。 2021 年 5 月 26 日

- 已创建 [Amazon MWAA 环境指标 CloudWatch \(p. 264\)](#)。
- 已创建中的 Apache Airflow v2 环境指标 [CloudWatch \(p. 264\)](#)。
- 向 Apache Airflow v2.0.2 添加了特定版本的链接在 [亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。
- 在 Apache Airflow v2.0.2 版本特定指南中添加 [安装 Python 的 \(p. 117\)](#) 了。
- 在 Apache Airflow v2.0.2 版本特定指南中添加在 [requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#) 了。
- 向 Apache Airflow v2.0.2 添加了示例插件 [安装自定义插件 \(p. 106\)](#)。
- 向 Apache Airflow v2.0.2 添加了示例代码在 [亚马逊 MWAA 环境中清理 Aurora PostgreSQL \(p. 181\)](#)。
- 向 Apache Airflow v2.0.2 添加了示例代码在 [使用中密钥 AWS Secrets Manager 用于 Apache 气流连接 \(p. 189\)](#)。
- 向 Apache Airflow v2.0.2 添加了示例代码为 [Apache Airflow 创建自定义插件 PythonVirtualenvOperator \(p. 207\)](#)。
- 向 Apache Airflow v2.0.2 添加 [Apache Airflow Air \(p. 66\)](#) 了。
- 将 Apache Airflow v2.0.2 脚本添加到 [创建 Apache 气流 CLI 令牌 \(p. 61\)](#)。
- 澄清了现有的 Apache Airflow v1.10.12 指标内容，并在上添加了 Apache Airflow 指标类别中的 [Apache Airflow v1 环境指标 CloudWatch \(p. 270\)](#)。
- 添加了一条说明，即亚马逊 MWAA 默认使用最新的 Apache Airflow 版本 [创建亚马逊 MWAA 环境 \(p. 28\)](#)。

新主题和用例 (p. 328)	添加以下更改：	2021 年 5 月 14 日
	添加了对卡住或未运行的 Airflow 任务进行故障排除的指南Amazon Managed Workflows a Airflow (MWAA) (p. 304)。	
修复 (p. 328)	添加以下更改：	2021 年 5 月 12 日
	我们更新了示例插件代码，以使用最新的 Java 版本使用 Apache Hive 和 Hadoop 创建自定义插件 (p. 204)。以前，确实如此<code>os.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"</code>。	
已删除或移动的主题 (p. 328)	添加以下更改：	2021 年 5 月 10 日
	按类别Amazon Managed Workflows a Airflow (MWAA) (p. 304)将主题移入新页面。	
新主题和用例 (p. 328)	添加以下更改：	2021 年 5 月 10 日
	在 Amazon S3 存储桶概述中添加在亚马逊上使用 DAG MWAA (p. 103)了。	
已删除或移动的主题 (p. 328)	添加以下更改：	2021 年 5 月 7 日
	移访问 Apache Airflow 界面 (p. 58)至顶级导航，并添加了创建 Apache Airflow Web 登录令牌 (p. 59)创建 Apache 气流 CLI 令牌 (p. 61)、和的页面Apache Airflow Air (p. 66)。	
新主题和用例 (p. 328)	添加以下更改：	2021 年 5 月 7 日
	- 在 Apache Airflow 参考指南中添加了特定版本的链接，其中包含所有支持和不支持的 Airflow CLI 命令Apache Airflow Air (p. 66)。 - 为中的所有配置选项添加了指向 Apache Airflow 参考指南的特定版本链接在亚马逊 MWAA 上使用 Apache Airflow 配置选项 (p. 90)。 - 将亚马逊 MWAA CLI 实用程序添加到在 requirements.txt 中管理 Python 依赖关系 (p. 248)。	

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 30 日

- 添加了有关如何在中构造 plugins.zip 的平面和嵌套示例[安装自定义插件 \(p. 106\)](#)。
- 将亚马逊 MWAA CLI 实用程序添加到[添加或更新 DAG \(p. 103\)](#)[安装自定义插件 \(p. 106\)](#)、和[安装 Python 的 \(p. 117\)](#)页面。
- 根据和[安装 Python 的 \(p. 117\)](#)页面中的用户反馈，将内容重构为概览，上传到 Amazon S3，并在 Amazon MWAA 部分上安装。[安装自定义插件 \(p. 106\)](#)
- 添加了一个示例用例，用于创建所需的 VPC 终端节点并将其连接到无需访问互联网的现有 Amazon VPC[关于 Amazon MWAA 上的联网 \(p. 126\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 30 日

- 为中的 Apache Airflow 变量添加了使用 Secrets Manager 中的密钥的示例代码[在中使用密钥AWS Secrets Manager对于 Apache 气流变量 \(p. 187\)](#)。

[新指南 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 30 日

- 已创建[使用私有路由在 Amazon VPC 中创建所需的 VPC 服务终端节点 \(p. 136\)](#)。

[修复 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 30 日

- 哎哟！我们已更新 core.default_ui_timezonewebserver.default_ui_timezone为[在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 23 日

- 添加了 SSH 隧道的 Windows (PuTTY) 步骤教程：[使用 Linux 堡垒主机配置专用网络访问权限 \(p. 162\)](#)。
- 添加了仅与 Apache Airflow 2.0 兼容的主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。apache-airflow-providers-amazon

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 23 日

- 在中添加了使用密钥 Secrets Manager 中的 Apache Airflow 连接的示例代码 [在中使用密钥 AWS Secrets Manager 用于 Apache 气流连接 \(p. 189\)](#)。

[新指南 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 23 日

- 已创建关于 [Amazon MWAA 上的联网 \(p. 126\)](#)。
- 已创建 [您在亚马逊 MWAA 上的 VPC 中的安全性 \(p. 129\)](#)。
- 已创建在 [Amazon MWAA 上管理 VPC 终端节点的访问 \(p. 132\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 16 日

- 添加了一个新 AWS CloudFormation 模板，用于在不访问互联网的情况下创建亚马逊 VPC 网络 [创建 VPC 网络 \(p. 18\)](#)。
- 添加了创建入口的新教程教程：[使用配置专用网络访问权限 AWS Client VPN \(p. 155\)](#)。AWS Client VPN
- 根据用户反馈和中的简化文档，将网络访问页面的名称更改为 Apache Airflow 访问模式 [Apache Airflow \(p. 52\)](#)。
- 简化了文档，仅包含基于用户反馈的 Amazon VPC 入门信息和模板 [创建 VPC 网络 \(p. 18\)](#)。
- 向中添加了 BigQuery 操作员解决方法 [Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了 Apache Airflow v1.10.12 限制文件最佳实践 [安装 Python 的 \(p. 117\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 16 日

- 添加了使用 Oracle 创建自定义插件的示例代码 [使用 Oracle 创建自定义插件 \(p. 192\)](#)。
- 添加了示例代码，用于创建在中生成运行时环境变量的自定义插件 [创建生成运行时环境变量的自定义插件 \(p. 196\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 9 日

- 向 VPC 安全组添加了有关自引用规则要求的主题[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 添加了自定义插件目录和大小限制[安装自定义插件 \(p. 106\)](#)。
- 向添加了需求目录和大小限制[安装 Python 的 \(p. 117\)](#)。
- 澄清了 `foo.user` 和 `foo.pass` 中的 Apache Airflow 配置选项在 [requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。
- 向中添加了配置选项概述在[亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 4 月 9 日

- 添加了使用 `PythonVirtualenvOperator` 中创建自定义插件的示例代码为[Apache Airflow 创建自定义插件 PythonVirtualenvOperator \(p. 207\)](#)。
- 添加了示例代码，用于创建包含 Apache Hive 和 Hadoop 的自定义插件[使用 Apache Hive 和 Hadoop 创建自定义插件 \(p. 204\)](#)。

[修复 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 31 日

- 哎哟！我们更新了 `requirements.txt` 的格式，并在中添加了一个与 Apache Airflow v1.10.12 兼容的示例[安装 Python 的 \(p. 117\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 26 日

- 添加了删除 `requirements.txt` 或 `plugins.zip` 的解决方法[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 在环境中添加了 SSH 的 bash 解决方法[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 向中添加了 `CloudTrail ResourceAlreadyExistsException` 错误主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 29 日

- 添加了使用的AWS服务列表[Amazon MWAA 执行角色 \(p. 44\)](#)。
- 添加了使用的AWS服务列表[Amazon MWAA 的服务相关角色 \(p. 40\)](#)。
- 添加了亚马逊 MWAA 的 Python 3.7 版本的问题[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 为 PythonVirtualenvOperator 其添加了问题[亚马逊 MWAA 常见问题 \(p. 298\)](#)。
- 为与 VPC 和环境配置相关的所有主题添加了故障排除脚本作为后续步骤[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 澄清了文档，即 Ling 堡垒必须与环境相同的区域[教程：使用 Linux 堡垒主机配置专用网络访问权限 \(p. 162\)](#)。

[新指南 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 29 日

- 为 aAWS Secrets Manager t 创建了 Apache Airflow 连接指南[使用AWS Secrets Manager 密钥配置 Apache Airflow 连接 \(p. 79\)](#)。
- 使用AWS CloudFormation 模板创建了快速入门教程，用于创建 Amazon VPC 基础设施、Amazon S3 存储桶和 Amazon MWAA 环境，网址为[Workflows for Apache Airflow \(MWAA\) \(p. 5\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 12 日

- 添加了创建 Amazon S3 存储桶疑难解答主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了创建 JSON 策略并将其附加到的步骤[Amazon MWAA 执行角色 \(p. 44\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 12 日

- 添加了在触发 DAG 时添加配置的示例代码[访问 Apache Airflow 界面 \(p. 58\)](#)。

[新指南 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 12 日

- 创建了最佳实践指南，网址为[在 requirements.txt 中管理 Python 依赖关系 \(p. 248\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 5 日

- 向 Google/GCP/ 添加了 BigQuery 疑难解答主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 向 Cython 添加了疑难解答主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 向 MySQL 添加了故障排除主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 向中添加了 5xx Web 服务器错误疑难解答主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。

[现在支持 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 4 日

- 以前 backend_kwargs，不支持，您需要一种变通方法来覆盖 Secrets Manager 函数调用。AWS Secrets Manager 现在，backend_kwargs 已支持。请参阅中的 AWS Secrets Manager 疑难解答主题[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。

[修复 \(p. 328\)](#)

添加以下更改：

2021 年 3 月 4 日

- 哎哟！我们已经更新了每个环境类的大小，以反映其中的实际 GB 配置个 [Amazon MWAA 环境类 \(p. 85\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 26 日

- 添加了使用 VPC 终端节点策略的私有网络访问权限[Apache Airflow \(p. 52\)](#)。
- 向“创建环境疑难解答”主题添加了其他检查[Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了查看日志 requirements.txt 的步骤[安装 Python 的 \(p. 117\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 25 日

- 将 Apache Hive 用例添加到[安装 Python 的 \(p. 117\)](#)。
- 澄清了需要在 requirements.txt 文件中包含 Apache Airflow 软件包所需的依赖项的文档[安装 Python 的 \(p. 117\)](#)。
- 将更新 requirements.txt 疑难解答主题添加到[Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。

[新教程 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 22 日

- 向... 添加了专用网络教程[教程：使用 Linux 堡垒主机配置专用网络访问权限 \(p. 162\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 22 日

- 将私有和公共网络配置添加到[Apache Airflow \(p. 52\)](#)。
- 向中添加了开发组用例和用户场景[Amazon MWAA 执行角色 \(p. 44\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 22 日

- 向 Web 登录令牌和 CLI 令牌添加了示例 Python 脚本[访问 Apache Airflow 界面 \(p. 58\)](#)。
- 添加了在其他环境中触发 DAG 的示例代码[Managed Workflows for Airflow \(MWAA\) \(p. 172\)](#)。
- 添加了使用 Lambda 函数触发 DAG 的示例代码[使用 Lambda 函数调用 DAG \(p. 211\)](#)。

[新命令和程序 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 22 日

- 在的所有脚本中添加了分步程序[访问 Apache Airflow 界面 \(p. 58\)](#)。

[新的示例代码 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 17 日

- 更新了 Web 登录令牌的 curl 示例，网址为[访问 Apache Airflow 界面 \(p. 58\)](#)。
- 添加了连接到 Amazon RDS Microsoft SQL Server 的示例代码[通过 Amazon MWAA 连接 Amazon RDS for Microsoft SQL Server \(p. 215\)](#)。

[新命令和程序 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 17 日

- 向[在亚马逊上使用 DAG MWAA \(p. 103\)](#)页面添加了 AWS CLI 命令。
- Apache Airflow 在 CLI 命令中不支持序列化 DAG。由于 CLI 在 Web 服务器上运行，出于安全原因，该服务器没有插件或要求，因此任何带有 plugins.zip 或 requirements.txt 的 MWAA 环境都不支持这些命令。已将 Apache Airflow dags 的 backfill 命令移至不支持的命令[访问 Apache Airflow 界面 \(p. 58\)](#)。

[GitHub 发射 \(p. 328\)](#)

用户指南文档现已开源 GitHub。在任何页面上选择“编辑此页面 GitHub”。

2021 年 2 月 17 日

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 12 日

- 在 Step Functions 与亚马逊 MWAA 用例中添加了问题[Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 304\)](#)。
- 添加了 CLI 访问策略[访问 Amazon MWAA 环境 \(p. 32\)](#)。
- 澄清了可以在上指定任何支持的 Apache Airflow 配置选项的文档[在亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。
- 澄清了文档，即如果一个可用区中的 Fargate 容器出现故障，MWAA 会切换到另一个可用区中的另一个容器[创建 VPC 网络 \(p. 18\)](#)。

[新主题和用例 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 5 日

- 增加了[配置个 Amazon MWAA 环境类 \(p. 85\)](#)。

[已删除或移动的主题 \(p. 328\)](#)

添加以下更改：

2021 年 2 月 4 日

- 取消了 Amazon S3 存储桶名称以开 airflow- 头的要求[开始使用 Amazon Managed Workflows for Apache Airflow \(MWAA\) \(p. 15\)](#)。
- 已[访问 Amazon MWAA 环境 \(p. 32\)](#)移至[Amazon MWAA 执行角色 \(p. 44\)](#)至[管理对 Amazon MWAA 环境的访问 \(p. 32\)](#)。

[亚马逊 MWAA CloudFormation \(p. 328\)](#)

更新参数以在 A [amazon MWAA](#) 创建环境 CloudFormation。 2021 年 2 月 4 日

- 移除 SubnetList。
- 移除 TagList。
- 添加 NetworkConfiguration。
- 添加 TagMap。
- 添加创建环境请求示例。

[新主题和用例 \(p. 328\)](#)

添加以下更改： 2021 年 1 月 29 日

- 向中添加了电子邮件配置示例在 [亚马逊 MWAA 上使用 Apache Airflow 配置选项 \(p. 90\)](#)。
- 向中添加了 PostgresHook 疑难解答主题 [Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 向中添加了 AWS Secrets Manager 疑难解答主题 [Amazon Managed Workflows a Airflow \(MWAA\) \(p. 304\)](#)。
- 向... 添加了高性能用例 [配置亚马逊 MWAA 自动扩展 \(p. 87\)](#)。

[亚马逊 MWAA 发布 \(p. 328\)](#)

Amazon Managed Workflows for Apache Airflow (MAA) 2020 年 11 月 24 日

- 用户指南文档
- AWS CloudFormation 文档

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。