



Guide de l'utilisateur

Amazon Managed Workflows for Apache Airflow



Amazon Managed Workflows for Apache Airflow: Guide de l'utilisateur

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce qu'Amazon MWAA ?	1
Caractéristiques	1
Architecture	2
Integration	3
Versions prises en charge	4
Quelle est la prochaine étape ?	4
Démarrage rapide	5
Dans ce tutoriel	5
Prérequis	7
Étape 1 : enregistrer le CloudFormation modèle localement	7
Deuxième étape : créer la pile à l'aide du AWS CLI	16
Troisième étape : télécharger un DAG sur Amazon S3 et l'exécuter dans l'interface utilisateur d'Apache Airflow	17
Quatrième étape : accéder aux journaux dans CloudWatch Logs	18
Quelle est la prochaine étape ?	18
Mise en route	19
Prérequis	19
A propos de ce manuel	19
Avant de commencer	20
Régions disponibles	21
Création d'un compartiment	22
Avant de commencer	23
Créez le bucket	23
Quelle est la prochaine étape ?	25
Création du réseau VPC	25
Prérequis	26
Avant de commencer	26
Options pour créer le réseau Amazon VPC	26
Quelle est la prochaine étape ?	38
Création d'un environnement	38
Avant de commencer	39
Versions d'Apache Airflow	39
Création d'un environnement	41
Quelle est la prochaine étape ?	25

Gestion des accès	46
Accès à un environnement Amazon MWAA	46
Comment ça marche	47
Accès complet à la console	49
Accès complet à l'API	55
Accès à la console en lecture seule	60
Accès à l'interface utilisateur d'Apache Airflow	60
Accès à l'API Rest d'Apache Airflow	62
Accès à la CLI Apache Airflow	63
Création d'une politique JSON	63
Exemple de cas d'utilisation	64
Quelle est la prochaine étape ?	66
Rôle lié à un service	66
Autorisations de rôle liées à un service pour Amazon MWAA	67
Création d'un rôle lié à un service pour Amazon MWAA	70
Modification d'un rôle lié à un service pour Amazon MWAA	70
Supprimer un rôle lié à un service pour Amazon MWAA	71
Régions prises en charge pour les rôles liés aux services Amazon MWAA	71
Mises à jour des politiques	71
Rôle d'exécution	72
Vue d'ensemble des rôles d'exécution	73
Créer un rôle	75
Accéder à une politique de rôle d'exécution et la mettre à jour	75
Accorder l'accès au compartiment Amazon S3 avec un blocage d'accès public au niveau du compte	77
Utiliser les connexions Apache Airflow	78
Exemples de politiques	78
Quelle est la prochaine étape ?	84
Prévention du cas de figure de l'adjoint désorienté entre services	84
Modes d'accès Apache Airflow	86
Modes d'accès Apache Airflow	87
Vue d'ensemble des modes d'accès	88
Configuration pour les modes d'accès privé et public	88
Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)	90
Accès à Apache Airflow	91

Conditions préalables	91
Accès	91
AWS CLI	92
Ouvrez l'interface utilisateur d'Apache Airflow	92
Connectez-vous à Apache Airflow	92
Création d'un jeton d'accès au serveur Web	92
Prérequis	93
À l'aide du AWS CLI	94
Utiliser un script bash	94
Utilisation d'un script Python	95
Quelle est la prochaine étape ?	96
Configuration d'un domaine personnalisé	96
Configuration du domaine personnalisé	97
Configuration de l'infrastructure réseau	97
Jeton CLI Apache Airflow	102
Prérequis	103
En utilisant le AWS CLI	104
Utilisation d'un script curl	104
Utiliser un script bash	106
Utilisation d'un script Python	108
Quelle est la prochaine étape ?	111
Utilisation de l'API REST Apache Airflow	111
Octroi de l'accès à l'API REST Apache Airflow : <code>airflow:InvokeRestApi</code>	114
Appel de l'API REST Apache Airflow	115
Création d'un jeton de session de serveur Web et appel de l'API REST Apache Airflow	116
Référence des commandes de la CLI Apache Airflow	122
Prérequis	123
Qu'est-ce qui a changé ?	123
Commandes CLI prises en charge	124
Exemple de code	131
Gestion des connexions	134
Aperçu	134
Paquets Apache Airflow	135
Fichier de contraintes	135
Packages de fournisseurs spécifiques à la version	136
Types de connexion	144

Exemple de chaîne d'URI de connexion	145
Exemple de modèle de connexion	145
Exemple d'utilisation d'un modèle de connexion HTTP pour une connexion Jdbc	146
Configuration de Secrets Manager	146
Étape 1 : donnez à Amazon MWAA l'autorisation d'accéder aux clés secrètes de Secrets Manager	147
Deuxième étape : créer le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow	148
Troisième étape : générer une chaîne d'URI de AWS connexion Apache Airflow	150
Étape 4 : ajouter les variables dans Secrets Manager	153
Étape 5 : ajouter la connexion dans Secrets Manager	154
Exemple de code	155
Ressources	155
Quelle est la prochaine étape ?	156
Gestion des environnements	157
Configuration de la classe d'environnement	157
Capacités environnementales	158
Planificateurs Apache Airflow	160
Configuration de la mise à l'échelle automatique des travailleurs	161
Comment fonctionne Worker Scaling	161
Utilisation de la console Amazon MWAA	162
Exemple de cas d'utilisation à hautes performances	162
Tâches de dépannage bloquées en cours d'exécution	164
Quelle est la prochaine étape ?	165
Configuration de la mise à l'échelle automatique du serveur Web	165
Comment fonctionne la mise à l'échelle du serveur Web	165
Utilisation de la console Amazon MWAA	166
Utilisation des options de configuration	166
Prérequis	167
Comment ça marche	168
Utilisation des options de configuration pour charger des plugins	168
Vue d'ensemble des options de configuration	168
Référence de configuration	169
Exemples et exemple de code	175
Quelle est la prochaine étape ?	176
Mise à jour d'un environnement	176

Avant de commencer	177
Stratégie de remplacement des travailleurs	177
Mettre à jour les ressources environnementales	178
Mise à jour d'un environnement	178
Modification de la version	182
Améliorez ou rétrogradez les ressources de votre flux de travail	183
Spécifiez la nouvelle version	184
Utilisation d'un script de démarrage	185
Configuration d'un script de démarrage	186
Installation des environnements d'exécution Linux	189
Définir les variables d'environnement	191
Travailler avec DAGs	195
Présentation du bucket Amazon S3	195
Ajouter ou mettre à jour DAGs	196
Prérequis	196
Comment ça marche	197
Qu'est-ce qui a changé ?	198
Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI	198
Téléchargement du code DAG sur Amazon S3	198
Spécifier le chemin d'accès à un DAGs dossier	200
Accès aux modifications sur votre interface utilisateur Apache Airflow	200
Quelle est la prochaine étape ?	200
Installation de plugins personnalisés	201
Prérequis	201
Comment ça marche	202
Quand utiliser les plugins	202
Vue d'ensemble des plugins personnalisés	203
Exemples de plugins personnalisés	204
Création d'un fichier plugins.zip	214
Téléchargement plugins.zip vers Amazon S3	215
Installation de plugins personnalisés sur votre environnement	216
Exemples de cas d'utilisation pour plugins.zip	217
Quelle est la prochaine étape ?	217
Installation des dépendances Python	218
Conditions préalables	218
Comment ça marche	219

Vue d'ensemble des dépendances Python	220
Création d'un fichier requirements.txt	220
Téléchargement requirements.txt vers Amazon S3	224
Installation de dépendances Python dans votre environnement	225
Accès aux journaux de votre requirements.txt	226
Quelle est la prochaine étape ?	227
Suppression de fichiers sur Amazon S3	227
Prérequis	228
Vue d'ensemble du versionnement	228
Fonctionnement	229
Supprimer un DAG sur Amazon S3	229
Suppression du fichier « actuel » plugins.zip ou requirements.txt	230
Supprimer le fichier « non actuel » plugins.zip ou requirements.txt	230
Supprimer des fichiers avec des cycles de vie	230
Exemple de politique de cycle de vie	231
Quelle est la prochaine étape ?	231
Réseaux	233
À propos du réseau	233
Conditions	234
Ce qui est pris en charge	234
Présentation de l'infrastructure VPC	234
Exemples de cas d'utilisation pour un Amazon VPC et le mode d'accès Apache Airflow	238
Sécurité dans votre VPC	240
Conditions	241
Aperçu de la sécurité	241
Listes de contrôle d'accès au réseau (ACLs)	242
Groupes de sécurité VPC	242
Politiques de point de terminaison VPC (routage privé uniquement)	245
Gestion de l'accès aux points de terminaison VPC	246
Tarification	247
Vue d'ensemble des points de terminaison VPC	247
Autorisation d'utiliser d'autres AWS services	248
Accès aux points de terminaison VPC	248
Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)	250
Points de terminaison de service VPC sur Amazon privé VPCs	252

Tarification	252
Réseau privé et routage privé	252
Points de terminaison VPC (obligatoires)	253
Connexion des points de terminaison VPC requis	254
(Facultatif) Activez les adresses IP privées pour le point de terminaison de votre interface Amazon S3 VPC	257
Gestion de vos propres points de terminaison Amazon VPC	258
Création d'un environnement dans un Amazon VPC partagé	259
Tutoriels	270
Tutoriel : AWS Client VPN	270
Réseau privé	271
Cas d'utilisation	271
Avant de commencer	271
Objectifs	272
(Facultatif) Première étape : identifier votre VPC, vos règles CIDR et la sécurité de votre VPC	272
Deuxième étape : créer les certificats du serveur et du client	273
Troisième étape : enregistrer le CloudFormation modèle localement	275
Quatrième étape : créer la CloudFormation pile VPN du Client	276
Étape 5 : associer des sous-réseaux à votre Client VPN	277
Sixième étape : ajouter une règle d'entrée d'autorisation à votre Client VPN	277
Étape 7 : Téléchargez le fichier de configuration du point de terminaison du Client VPN	278
Huitième étape : Connectez-vous au AWS Client VPN	280
Quelle est la prochaine étape ?	280
Tutoriel : Linux Bastion Host	281
Réseau privé	281
Cas d'utilisation	282
Avant de commencer	282
Objectifs	282
Première étape : créer l'instance de bastion	283
Deuxième étape : créer le tunnel SSH	284
Troisième étape : configurer le groupe de sécurité Bastion en tant que règle entrante	285
Étape 4 : Copier l'URL d'Apache Airflow	286
Étape 5 : configurer les paramètres du proxy	286
Sixième étape : ouvrir l'interface utilisateur d'Apache Airflow	289
Quelle est la prochaine étape ?	289

Tutoriel : Restreindre les utilisateurs à un sous-ensemble de DAGs	290
Conditions préalables	290
Étape 1 : fournissez un accès au serveur Web Amazon MWAA à votre principal IAM avec le rôle Apache Airflow Public par défaut.	291
Deuxième étape : créer un nouveau rôle personnalisé pour Apache Airflow	292
Troisième étape : attribuer le rôle que vous avez créé à votre utilisateur Amazon MWAA	293
Étapes suivantes	294
Ressources connexes	294
Tutoriel : Automatisez la gestion des points de terminaison de votre propre environnement	294
Prérequis	295
Création de l'Amazon VPC	296
Créer la fonction Lambda	296
Créez la EventBridge règle	297
Création de l'environnement	298
Exemples de code	300
Variables d'importation DAG	301
Version	301
Prérequis	301
Autorisations	301
Dépendances	301
Exemple de code	302
Quelle est la prochaine étape ?	303
Utilisation de l'SSHOperator	303
Version	304
Prérequis	304
Autorisations	304
Prérequis	305
Copiez votre clé secrète sur Amazon S3	305
Création d'une nouvelle connexion Apache Airflow	305
Exemple de code	306
Connexion Apache Airflow Snowflake dans Secrets Manager	308
Version	308
Prérequis	308
Autorisations	308
Prérequis	309
Exemple de code	309

Quelle est la prochaine étape ?	310
Utiliser un DAG pour écrire des métriques personnalisées	310
Version	311
Prérequis	311
Autorisations	311
Dépendances	311
Exemple de code	311
Nettoyage de base de données Aurora PostgreSQL	314
Version	315
Conditions préalables	315
Dépendances	315
Exemple de code	315
Exportation des métadonnées de l'environnement vers Amazon S3	319
Version	319
Conditions préalables	319
Permissions	320
Exigences	320
Exemple de code	320
Utilisation d'une variable Apache Airflow dans Secrets Manager	323
Version	323
Prérequis	323
Autorisations	323
Prérequis	324
Exemple de code	324
Quelle est la prochaine étape ?	325
Utilisation d'une connexion Apache Airflow dans Secrets Manager	325
Version	325
Prérequis	326
Autorisations	326
Prérequis	324
Exemple de code	326
Quelle est la prochaine étape ?	328
Plug-in personnalisé avec Oracle	328
Version	329
Prérequis	329
Autorisations	329

Prérequis	329
Exemple de code	329
Créez le plugin personnalisé	330
Options de configuration d'Airflow	334
Quelle est la prochaine étape ?	334
Modifier le fuseau horaire d'un DAG	334
Version	334
Prérequis	335
Permissions	335
Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow	335
Créer une plugins.zip	336
Exemple de code	336
Quelle est la prochaine étape ?	337
Actualisation d'un AWS CodeArtifact jeton lors de l'exécution	338
Version	338
Prérequis	338
Autorisations	338
Exemple de code	339
Quelle est la prochaine étape ?	340
Plugin personnalisé avec Apache Hive et Hadoop	341
Version	341
Prérequis	341
Autorisations	342
Prérequis	324
Dépendances de téléchargement	342
Plug-in personnalisé	343
Plugins.zip	344
Exemple de code	344
Options de configuration d'Airflow	345
Quelle est la prochaine étape ?	345
Plugin personnalisé à patcher PythonVirtualenvOperator	345
Version	346
Prérequis	346
Autorisations	346
Prérequis	346
Exemple de code de plugin personnalisé	346

Plugins.zip	347
Exemple de code	348
Options de configuration d'Airflow	349
Quelle est la prochaine étape ?	349
Invoker DAGs avec Lambda	349
Version	350
Conditions préalables	350
Permissions	350
Dépendances	351
Exemple de code	351
Invocation DAGs dans différents environnements	352
Version	353
Conditions préalables	353
Permissions	353
Dépendances	354
Exemple de code	354
Serveur Amazon RDS	356
Version	356
Prérequis	356
Dépendances	315
Connexion Apache Airflow v2	357
Exemple de code	358
Quelle est la prochaine étape ?	360
Amazon EKS (extrait)	360
Version	361
Conditions préalables	361
Création d'une clé publique pour Amazon EC2	361
Créer le cluster	362
Création d'un espace mwaa de noms	363
Création d'un rôle pour l'espace de mwaa noms	363
Création et attachement d'un rôle IAM pour le cluster Amazon EKS	364
Créez le fichier requirements.txt	367
Création d'un mappage d'identité pour Amazon EKS	368
Créer le kubeconfig	368
Création d'un DAG	368
Ajoutez le DAG et kube_config.yaml au compartiment Amazon S3	370

Activer et déclencher l'exemple	370
Utilisation de l'ECSOperator	370
Version	371
Conditions préalables	371
Permissions	371
Création d'un cluster Amazon ECS	373
Exemple de code	377
Utiliser dbt avec Amazon MWAA	381
Version	381
Conditions préalables	381
Dépendances	381
Importer un projet dbt sur Amazon S3	383
Utiliser un DAG pour vérifier l'installation de la dépendance dbt	383
Utiliser un DAG pour exécuter un projet dbt	384
AWS blogs et tutoriels	385
Bonnes pratiques	386
Optimisation des performances pour Apache Airflow	386
Ajout d'une option de configuration d'Apache Airflow	386
Planificateur Apache Airflow	387
Dossiers DAG	391
fichiers DAG	393
Tâches	396
Gestion des dépendances en Python	400
Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI	401
Installation de dépendances Python à l'aide du format de fichier d'exigences PyPi .org	401
Activation des journaux sur la console Amazon MWAA	409
Accès aux journaux sur la console CloudWatch Logs	409
Erreurs d'accès dans l'interface utilisateur d'Apache Airflow	410
Exemples de requirements.txt scénarios	410
Surveillance et mesures	412
Présentation de	412
CloudWatch Présentation d'Amazon	413
Présentation de AWS CloudTrail	413
Accès aux journaux d'audit	413
Création d'un parcours dans CloudTrail	414
Accès aux événements avec l'historique des CloudTrail événements	414

Exemple de parcours pour CreateEnvironment	414
Quelle est la prochaine étape ?	416
Accès aux journaux Airflow	416
Tarification	417
Avant de commencer	417
Types de journaux	417
Activation des journaux Apache Airflow	418
Accès aux journaux d'Apache Airflow	419
Exemples de journaux du planificateur	419
Quelle est la prochaine étape ?	420
Tableaux de bord et alarmes de surveillance	420
Métriques	421
Vue d'ensemble des états d'alarme	421
Exemples de tableaux de bord et d'alarmes personnalisés	421
Supprimer des métriques et des tableaux de bord	426
Quelle est la prochaine étape ?	426
Métriques de l'environnement Apache Airflow	426
Conditions	427
Dimensions	427
Accès aux métriques dans la CloudWatch console	429
Les métriques d'Apache Airflow sont disponibles dans CloudWatch	429
Choix des indicateurs à signaler	450
Quelle est la prochaine étape ?	450
Mesures relatives aux conteneurs, aux files d'attente et aux bases de données	451
Conditions	452
Dimensions	452
Accès aux métriques	453
Liste des métriques	453
Sécurité	458
Protection des données	459
Chiffrement	460
Utilisation de clés gérées par le client	462
Gestion des identités et des accès AWS	466
Public ciblé	466
Authentification avec des identités	467
Gestion des accès à l'aide de politiques	468

Permettre aux utilisateurs d'accéder à leurs propres autorisations	470
Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow	471
Comment Amazon MWAA fonctionne avec IAM	472
Validation de la conformité	478
Résilience	478
Sécurité de l'infrastructure	479
Configuration et analyse des vulnérabilités	479
Bonnes pratiques	480
Bonnes pratiques de sécurité dans Apache Airflow	480
Versions	483
À propos des versions Amazon MWAA	483
Dernière version	483
Versions d'Apache Airflow	483
Composants d'Apache Airflow	485
Schedulers	485
Workers	486
Mise à niveau de la version d'Apache Airflow	486
Rétrogradation de la version d'Apache Airflow	486
Versions obsolètes d'Apache Airflow	487
Support des versions d'Apache Airflow et FAQ	487
Questions fréquentes (FAQ)	487
Points de terminaison et quotas	490
Points de terminaison de service	490
Quotas de service	490
Augmenter les quotas	491
FAQs	492
Versions prises en charge	493
Assistance avec Apache Airflow	493
Version de Python	493
Cas d'utilisation	495
Puis-je utiliser Amazon MWAA avec Amazon SageMaker Unified Studio ?	495
Quand puis-je utiliser AWS Step Functions vs. Amazon MWAA ?	495
Spécifications relatives à l'environnement	496
Quelle est la capacité de stockage des tâches disponible pour chaque environnement ?	496
Système d'exploitation par défaut	496

Images personnalisées	496
Conformité à la loi américaine HIPAA	496
Amazon MWAA prend-il en charge les instances ponctuelles ?	496
Domaine personnalisé	497
Accès SSH	497
Règle d'autoréférencement	498
Métriques personnalisées	498
Stockez les données	498
Quota de travailleurs	498
Amazon partagé VPCs	499
Amazon partagé VPCs	499
Métriques	499
Indicateurs relatifs aux travailleurs	499
Métriques personnalisées	499
DAGs, Opérateurs, connexions et autres questions	500
PythonVirtualenvOperator	500
Combien de temps faut-il à Amazon MWAA pour reconnaître un nouveau fichier DAG ?	500
Pourquoi mon fichier DAG n'est-il pas récupéré par Apache Airflow ?	500
Supprimer plugins.zip ou requirements.txt	501
Supprimer plugins.zip ou requirements.txt	501
Puis-je utiliser les opérateurs du Service AWS de Migration de Base de Données (DMS) ? ..	501
Lorsque j'accède à l'API REST Airflow à l'aide des AWS informations d'identification, puis-je augmenter la limite de régulation à plus de 10 transactions par seconde (TPS) ?	501
Où s'exécute le serveur d'API d'exécution des tâches Airflow dans Amazon MWAA ?	502
Résolution de problème	503
Apache Airflow versions 2 et 3	505
Connexions	506
Serveur Web	508
Tâches	510
INTERFACE DE LIGNE DE COMMANDE (CLI)	512
Opérateurs	513
Amazon MWAA Créer/Mettre à jour	514
Mise à jour des requirements.txt	515
Plug-ins	516
Créer un bucket	517
Créer un environnement	517

Mettre à jour l'environnement	521
Environnement d'accès	521
CloudWatch Logs et CloudTrail	522
Journaux	522
Historique du guide de l'utilisateur Amazon MWAA	528
.....	dciv

Qu'est-ce qu'Amazon Managed Workflows pour Apache Airflow ?

Utilisez Amazon Managed Workflows pour Apache Airflow, un service géré pour [Apache Airflow](#), pour configurer et exécuter des pipelines de données dans le cloud à grande échelle. Apache Airflow est un outil open source utilisé pour créer, planifier et surveiller les flux de travail.

Avec Amazon MWAA, vous pouvez utiliser Apache Airflow et Python pour créer des flux de travail sans gérer l'infrastructure en termes d'évolutivité, de disponibilité et de sécurité. Amazon MWAA s'adapte automatiquement à vos besoins en matière de flux de travail. Il s'intègre aux services AWS de sécurité pour fournir un accès rapide et sécurisé à vos données.

Contenu

- [Caractéristiques](#)
- [Architecture](#)
- [Integration](#)
- [Versions prises en charge](#)
- [Quelle est la prochaine étape ?](#)

Caractéristiques

Consultez les fonctionnalités suivantes pour découvrir comment Amazon MWAA peut simplifier la gestion de vos flux de travail Apache Airflow.

- Configuration automatique du flux d'air : configurez rapidement Apache Airflow en choisissant une version d'[Apache Airflow lorsque](#) vous créez un environnement Amazon MWAA. Amazon MWAA configure Apache Airflow pour vous en utilisant la même interface utilisateur Apache Airflow et le même code open source que ceux disponibles sur Internet.
- Mise à l'échelle automatique : dimensionnez automatiquement les travailleurs d'Apache Airflow (les ressources de calcul qui exécutent vos tâches) en définissant des limites minimales et maximales. Amazon MWAA surveille les travailleurs de votre environnement et utilise son [composant de mise à l'échelle automatique](#) pour ajouter des travailleurs afin de répondre à la demande, dans la limite du nombre maximum que vous avez défini.
- Authentification intégrée : activez l'authentification et l'autorisation basées sur les rôles pour votre serveur Web Apache Airflow en définissant les politiques Gestion des identités et des accès AWS

de [contrôle d'accès dans](#) (IAM). Les travailleurs d'Apache Airflow adoptent ces politiques pour un accès sécurisé aux AWS services.

- **Sécurité intégrée** : les serveurs et les planificateurs d'Apache Airflow s'exécutent dans Amazon VPC d'[Amazon MWAA](#). Les données sont également automatiquement cryptées à l'aide de ce logiciel AWS Key Management Service, de sorte que votre environnement est sécurisé par défaut.
- **Modes d'accès public ou privé** : [accédez à votre serveur Web Apache Airflow en utilisant un mode d'accès privé ou public](#). Le mode d'accès au réseau public utilise un point de terminaison VPC pour votre serveur Web Apache Airflow accessible via Internet. Le mode d'accès au réseau privé utilise un point de terminaison VPC pour votre serveur Web Apache Airflow accessible depuis votre VPC. Dans les deux cas, l'accès de vos utilisateurs d'Apache Airflow est contrôlé par la politique de contrôle d'accès que vous définissez dans Gestion des identités et des accès AWS (IAM) et par le SSO. AWS
- **Mises à niveau et correctifs simplifiés** : Amazon MWAA fournit régulièrement de nouvelles versions d'Apache Airflow. L'équipe Amazon MWAA mettra à jour et corrigera les images pour ces versions.
- **Surveillance du flux de travail** : accédez aux journaux Apache Airflow et aux [métriques d'Apache Airflow](#) sur CloudWatch Amazon pour identifier les retards ou les erreurs de flux de travail d'Apache Airflow sans avoir besoin d'outils tiers supplémentaires. Amazon MWAA envoie automatiquement les métriques de l'environnement et, si elles sont activées, Apache Airflow se connecte à CloudWatch
- **AWS intégration** — Amazon MWAA prend en charge les intégrations open source avec Amazon Athena, CloudWatch Amazon AWS Batch, Amazon DynamoDB, Amazon EMR, AWS DataSync Amazon EKS, Amazon Data Firehose AWS Fargate, Amazon AWS Lambda Redshift, Amazon SQS AWS Glue, Amazon SNS, Amazon AI et Amazon S3, ainsi que des centaines d'Amazon S3 intégrés et créés par la communauté SageMaker opérateurs et capteurs.
- **Flottes de travailleurs** : [Amazon MWAA propose une assistance pour l'utilisation de conteneurs afin de faire évoluer le parc de travailleurs à la demande et de réduire les interruptions de service du planificateur à l'aide d'Amazon ECS on. AWS Fargate](#) Les opérateurs qui appellent des tâches sur des conteneurs Amazon ECS et les opérateurs Kubernetes qui créent et exécutent des pods sur un cluster Kubernetes sont pris en charge.

Architecture

Tous les composants contenus dans la boîte extérieure (dans l'image suivante) sont présentés sous la forme d'un seul environnement Amazon MWAA dans votre compte. Le planificateur et les outils de travail Apache Airflow sont des AWS Fargate conteneurs qui se connectent aux sous-réseaux

privés de votre environnement Amazon VPC. Chaque environnement possède sa propre base de métadonnées Apache Airflow gérée par AWS laquelle le planificateur et les travailleurs ont accès aux conteneurs Fargate via un point de terminaison VPC sécurisé par le biais d'un point de terminaison VPC privé.

Amazon CloudWatch, Amazon S3, Amazon SQS et Amazon AWS KMS sont distincts d'Amazon MWAA et doivent être accessibles depuis les planificateurs et les opérateurs d'Apache Airflow dans les conteneurs Fargate. Plusieurs planificateurs Apache Airflow ne sont disponibles qu'avec Apache Airflow v2 et versions ultérieures. Pour en savoir plus sur le cycle de vie des tâches d'Apache Airflow, consultez [Concepts](#) dans le guide de référence d'Apache Airflow.

Le serveur Web Apache Airflow est accessible soit via Internet en sélectionnant le mode d'accès Apache Airflow au réseau public, soit au sein de votre VPC en sélectionnant le mode d'accès Apache Airflow au réseau privé. Dans les deux cas, l'accès de vos utilisateurs d'Apache Airflow est contrôlé par la politique de contrôle d'accès que vous définissez dans Gestion des identités et des accès AWS (IAM).

Note

À partir d'Apache Airflow v3, le serveur Web Amazon MWAA héberge également le serveur d'API d'exécution d'Apache Airflow.

Integration

La communauté open source active et croissante d'Apache Airflow fournit des opérateurs (plugins qui simplifient les connexions aux services) permettant à Apache Airflow de s'intégrer aux services. AWS Cela inclut des services tels qu'Amazon S3, Amazon Redshift, Amazon EMR AWS Batch et Amazon SageMaker AI, ainsi que des services sur d'autres plateformes cloud.

L'utilisation d'Apache Airflow avec Amazon MWAA prend entièrement en charge l'intégration avec AWS des services et des outils tiers populaires tels qu'Apache Hadoop, Presto, Hive et Spark pour effectuer des tâches de traitement des données. Amazon MWAA s'engage à maintenir la compatibilité avec l'API Apache Airflow, et Amazon MWAA a l'intention de fournir des intégrations fiables aux services, de les mettre AWS à la disposition de la communauté et de participer au développement de fonctionnalités communautaires.

Pour un exemple de code, reportez-vous à [Exemples de code pour Amazon Managed Workflows pour Apache Airflow](#).

Versions prises en charge

Amazon MWAA prend en charge plusieurs versions d'Apache Airflow. Pour plus d'informations sur les versions d'Apache Airflow que nous prenons en charge et sur les composants Apache Airflow inclus dans chaque version, reportez-vous à [Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow](#)

Quelle est la prochaine étape ?

- Commencez avec un CloudFormation modèle unique qui crée un compartiment Amazon S3 pour votre flux d'air DAGs et les fichiers de support, un Amazon VPC avec routage public et un environnement Amazon MWAA dans [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#)
- Commencez progressivement en créant un compartiment Amazon S3 pour votre flux d'air DAGs et les fichiers de support, en choisissant l'une des trois options de mise en réseau Amazon VPC et en créant un environnement Amazon MWAA dans [Commencez avec Amazon Managed Workflows for Apache Airflow](#)

Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow

Ce didacticiel de démarrage rapide utilise un AWS CloudFormation modèle qui crée ensemble l'infrastructure Amazon VPC, un compartiment Amazon S3 avec un dags dossier et un environnement Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Dans ce tutoriel](#)
- [Prérequis](#)
- [Étape 1 : enregistrer le CloudFormation modèle localement](#)
- [Deuxième étape : créer la pile à l'aide du AWS CLI](#)
- [Troisième étape : télécharger un DAG sur Amazon S3 et l'exécuter dans l'interface utilisateur d'Apache Airflow](#)
- [Quatrième étape : accéder aux journaux dans CloudWatch Logs](#)
- [Quelle est la prochaine étape ?](#)

Dans ce tutoriel

Utilisez ce didacticiel pour télécharger un DAG sur Amazon S3, exécuter le DAG dans Apache Airflow et accéder aux logs à CloudWatch l'aide de trois AWS Command Line Interface (AWS CLI) commandes. Enfin, vous apprendrez à créer une politique IAM pour une équipe de développement d'Apache Airflow.

Note

Le CloudFormation modèle de cette page crée un environnement Amazon Managed Workflows for Apache Airflow pour la dernière version d'Apache Airflow disponible dans. CloudFormation La dernière version disponible est Apache Airflow v3.0.6.

Le CloudFormation modèle crée les éléments suivants :

- Infrastructure VPC. Le modèle utilise [Routage public sur Internet](#). Il utilise le [Mode d'accès au réseau public](#) pour le serveur Web Apache Airflow dans. `WebserverAccessMode` : `PUBLIC_ONLY`
- Compartiment Amazon S3. Le modèle crée un compartiment Amazon S3 avec un dags dossier. Il est configuré pour bloquer tout accès public, avec le contrôle de version des compartiments activé, comme défini dans [Création d'un compartiment Amazon S3 pour Amazon MWAA](#).
- Environnement Amazon MWAA. Le modèle crée un environnement Amazon MWAA associé au dags dossier du compartiment Amazon S3, un rôle d'exécution autorisant les AWS services utilisés par Amazon MWAA et le chiffrement par défaut à l'aide d'une [clé AWS appartenant](#) à Amazon, comme défini dans. [Création d'un environnement Amazon MWAA](#)
- CloudWatch Journaux. Le modèle active la connexion Apache Airflow au niveau INFO et CloudWatch au niveau supérieur pour le groupe de journaux du planificateur Airflow, le groupe de journaux du serveur Web Airflow, le groupe de journaux de travail Airflow, le groupe de journaux de traitement Airflow DAG et le groupe de journaux de tâches Airflow, tels que définis dans. [Accès aux journaux Airflow sur Amazon CloudWatch](#)

Dans ce didacticiel, vous allez effectuer les tâches suivantes :

- Téléchargez et exécutez un DAG. Téléchargez le didacticiel DAG d'Apache Airflow pour la dernière version d'Apache Airflow compatible avec Amazon MWAA sur Amazon S3, puis exécutez-le dans l'interface utilisateur d'Apache Airflow, comme défini dans. [Ajouter ou mettre à jour DAGs](#)
- Journaux d'accès. Accédez au groupe de journaux du serveur Web Airflow dans CloudWatch Logs, comme défini dans. [Accès aux journaux Airflow sur Amazon CloudWatch](#)
- Créez une politique de contrôle d'accès. Créez une politique de contrôle d'accès dans IAM pour votre équipe de développement Apache Airflow, comme défini dans. [Accès à un environnement Amazon MWAA](#)

Note

Dans le VPC qui héberge l'environnement Amazon MWAA, définissez ce paramètre sur `assignIpv6AddressOnCreation true` pour tous les sous-réseaux attachés. Ce paramètre garantit l'attribution automatique des adresses du protocole Internet version 6 (IPv6) aux ressources de ces sous-réseaux.

Prérequis

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Étape 1 : enregistrer le CloudFormation modèle localement

- Copiez le contenu du modèle suivant et enregistrez-le localement sous `mwaa-public-network.yml`. Vous pouvez également [télécharger le modèle](#).

```
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

MwaaExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17&TCX5-2025-waiver;
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - airflow-env.amazonaws.com
              - airflow.amazonaws.com
          Action:
            - "sts:AssumeRole"
    Path: "/service-role/"

MwaaExecutionPolicy:
  DependsOn: EnvironmentBucket
  Type: AWS::IAM::ManagedPolicy
  Properties:
    Roles:
      - !Ref MwaaExecutionRole
    PolicyDocument:
      Version: 2012-10-17&TCX5-2025-waiver;
      Statement:
        - Effect: Allow
          Action: airflow:PublishMetrics
          Resource:
            - !Sub "arn:aws:airflow:${AWS::Region}:${AWS::AccountId}:environment/
${EnvironmentName}"
        - Effect: Deny
          Action: s3:ListAllMyBuckets
          Resource:
            - !Sub "${EnvironmentBucket.Arn}"
            - !Sub "${EnvironmentBucket.Arn}/*"

        - Effect: Allow
          Action:
            - "s3:GetObject*"
            - "s3:GetBucket*"
            - "s3:List*"
          Resource:

```



```

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

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

```

Outputs:**VPC:**

Description: A reference to the created VPC

Value: !Ref VPC

PublicSubnets:

Description: A list of the public subnets

Value: !Join [",", [!Ref PublicSubnet1, !Ref PublicSubnet2]]

PrivateSubnets:

Description: A list of the private subnets

Value: !Join [",", [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]

PublicSubnet1:

Description: A reference to the public subnet in the 1st Availability Zone

Value: !Ref PublicSubnet1

PublicSubnet2:

Description: A reference to the public subnet in the 2nd Availability Zone

Value: !Ref PublicSubnet2

PrivateSubnet1:

Description: A reference to the private subnet in the 1st Availability Zone

Value: !Ref PrivateSubnet1

PrivateSubnet2:

Description: A reference to the private subnet in the 2nd Availability Zone

Value: !Ref PrivateSubnet2

SecurityGroupIngress:

Description: Security group with self-referencing inbound rule

Value: !Ref SecurityGroupIngress

MwaaApacheAirflowUI:

Description: MWAA Environment

Value: !Sub "https://\${MwaaEnvironment.WebserverUrl}"

Deuxième étape : créer la pile à l'aide du AWS CLI

1. Dans votre invite de commande, accédez au répertoire dans lequel `mwaa-public-network.yml` est stocké. Par exemple :

```
cd mwaaproject
```

2. Utilisez la [aws cloudformation create-stack](#) commande pour créer la pile à l'aide du AWS CLI.

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

 Note

La création de l'infrastructure Amazon VPC, du compartiment Amazon S3 et de l'environnement Amazon MWAA prend plus de 30 minutes.

Troisième étape : télécharger un DAG sur Amazon S3 et l'exécuter dans l'interface utilisateur d'Apache Airflow

1. Copiez le contenu du `tutorial.py` fichier correspondant à la [dernière version prise en charge d'Apache Airflow](#) et enregistrez-le localement sous `tutorial.py`
2. Dans votre invite de commande, accédez au répertoire dans lequel `tutorial.py` est stocké. Par exemple :

```
cd mwaaproject
```

3. Utilisez la commande suivante pour répertorier tous vos compartiments Amazon S3.

```
aws s3 ls
```

4. Utilisez la commande suivante pour répertorier les fichiers et les dossiers du compartiment Amazon S3 de votre environnement.

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

5. Utilisez le script suivant pour télécharger le `tutorial.py` fichier dags dans votre dossier. Remplacez la valeur de l'échantillon par *amzn-s3-demo-bucket*.

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

6. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
7. Choisissez un environnement.
8. Choisissez Open Airflow UI.
9. Dans l'interface utilisateur d'Apache Airflow, dans la liste des options disponibles DAGs, choisissez le didacticiel DAG.
10. Sur la page des détails du DAG, cliquez sur le bouton Pause/Annuler le DAG à côté du nom de votre DAG pour le remettre en pause.
11. Choisissez Trigger DAG.

Quatrième étape : accéder aux journaux dans CloudWatch Logs

Vous pouvez accéder aux journaux Apache Airflow dans la CloudWatch console pour tous les journaux Apache Airflow activés par la pile. CloudFormation La section suivante explique comment accéder aux journaux du groupe de journaux du serveur Web Airflow.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez le groupe de journaux du serveur Web Airflow dans le volet de surveillance.
4. Choisissez le `webserver_console_ip` log in Log streams.

Quelle est la prochaine étape ?

- En savoir plus sur la façon de télécharger DAGs et de spécifier les dépendances Python dans un fichier d'entrée `requirements.txt` et les plugins personnalisés `plugins.zip` dans un fichier d'entrée [Travailler avec DAGs sur Amazon MWAA](#).
- Découvrez les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement [Optimisation des performances pour Apache Airflow sur Amazon MWAA](#).
- Créez un tableau de bord de surveillance pour votre environnement dans [Tableaux de bord de surveillance et alarmes sur Amazon MWAA](#).
- Exécutez certains des exemples de code DAG dans [Exemples de code pour Amazon Managed Workflows pour Apache Airflow](#).

Commencez avec Amazon Managed Workflows for Apache Airflow

Amazon Managed Workflows pour Apache Airflow utilise le VPC Amazon, les fichiers DAG et les fichiers de support de votre compartiment de stockage Amazon S3 pour créer un environnement. Ce chapitre décrit les prérequis et les AWS ressources nécessaires pour démarrer avec Amazon MWAA.

Rubriques

- [Prérequis](#)
- [A propos de ce manuel](#)
- [Avant de commencer](#)
- [Régions disponibles](#)
- [Création d'un compartiment Amazon S3 pour Amazon MWAA](#)
- [Création du réseau VPC](#)
- [Création d'un environnement Amazon MWAA](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Pour créer un environnement Amazon MWAA, assurez-vous que vous êtes autorisé à accéder aux AWS ressources que vous devez créer.

- Compte AWS— Et Compte AWS avec l'autorisation d'utiliser Amazon MWAA et les AWS services et ressources utilisés par votre environnement.

A propos de ce manuel

Ce guide couvre l' AWS infrastructure et les ressources que vous allez créer.

- Amazon VPC — Les composants réseau Amazon VPC requis par un environnement Amazon MWAA. Vous pouvez configurer un VPC existant qui répond à ces exigences (avancées), comme indiqué dans [À propos de la mise en réseau sur Amazon MWAA](#), ou créer le VPC et les composants réseau, tels que définis dans. [the section called “Création du réseau VPC”](#)

- Compartiment Amazon S3 : compartiment Amazon S3 pour stocker vos fichiers DAGs et ceux associés, tels que `plugins.zip` et `requirements.txt`. Votre compartiment Amazon S3 doit être configuré pour bloquer tout accès public, avec le contrôle de version des compartiments activé, comme défini dans [Création d'un compartiment Amazon S3 pour Amazon MWAA](#).
- Environnement Amazon MWAA : environnement Amazon MWAA configuré avec l'emplacement de votre compartiment Amazon S3, le chemin d'accès à votre code DAG et à tous les plugins personnalisés ou dépendances Python, ainsi que votre Amazon VPC et son groupe de sécurité, tels que définis dans. [Création d'un environnement Amazon MWAA](#)

Avant de commencer

Pour créer un environnement Amazon MWAA, vous pouvez prendre des mesures supplémentaires pour créer et configurer d'autres AWS ressources avant de créer votre environnement.

Pour créer un environnement, vous avez besoin des éléments suivants :

- AWS KMS clé : AWS KMS clé pour le chiffrement des données de votre environnement. Vous pouvez choisir l'option par défaut sur la console Amazon MWAA pour créer une [clé AWS détenue](#) lorsque vous créez un environnement, ou spécifier une [clé gérée par le client](#) existante avec des autorisations pour les autres AWS services utilisés par votre environnement configurées (avancées). Pour en savoir plus, reportez-vous à [Utilisation de clés gérées par le client pour le chiffrement](#).
- Rôle d'exécution : rôle d'exécution qui permet à Amazon MWAA d'accéder aux AWS ressources de votre environnement. Vous pouvez choisir l'option par défaut sur la console Amazon MWAA pour créer un rôle d'exécution lorsque vous créez un environnement. Pour en savoir plus, reportez-vous à [Rôle d'exécution Amazon MWAA](#).
- Groupe de sécurité VPC : groupe de sécurité VPC qui permet à Amazon MWAA d'accéder à d'autres ressources AWS de votre réseau VPC. Vous pouvez choisir l'option par défaut sur la console Amazon MWAA pour créer un groupe de sécurité lorsque vous créez un environnement, ou fournir à un groupe de sécurité les règles entrantes et sortantes appropriées (avancées). Pour en savoir plus, reportez-vous à [Sécurité de votre VPC sur Amazon MWAA](#).

Régions disponibles

Amazon MWAA est disponible dans les versions suivantes Régions AWS. Pour en savoir plus sur chaque région, notamment celles qui sont activées ou désactivées par défaut, reportez-vous à [Régions AWS](#).

Code	Nom
us-east-1	USA Est (Virginie du Nord)
us-east-2	USA Est (Ohio)
us-west-1	USA Ouest (Californie du Nord)
us-west-2	USA Ouest (Oregon)
af-south-1	Afrique (Le Cap)
ap-east-1	Asie-Pacifique (Hong Kong)
ap-south-2	Asie-Pacifique (Hyderabad)
ap-southeast-3	Asie-Pacifique (Jakarta)
ap-southeast-5	Asie-Pacifique (Malaisie)
ap-southeast-4	Asie-Pacifique (Melbourne)
ap-south-1	Asie-Pacifique (Mumbai)
ap-northeast-3	Asie-Pacifique (Osaka)
ap-northeast-2	Asie-Pacifique (Séoul)
ap-southeast-1	Asie-Pacifique (Singapour)
ap-southeast-2	Asie-Pacifique (Sydney)
ap-northeast-1	Asie-Pacifique (Tokyo)
ca-central-1	Canada (Centre)

Code	Nom
ca-west-1	Canada-Ouest (Calgary)
eu-central-1	Europe (Francfort)
eu-west-1	Europe (Irlande)
eu-west-2	Europe (Londres)
eu-south-1	Europe (Milan)
eu-west-3	Europe (Paris)
eu-south-2	Europe (Espagne)
eu-north-1	Europe (Stockholm)
eu-central-2	Europe (Zurich)
il-central-1	Israël (Tel Aviv)
me-south-1	Middle East (Bahrain)
me-central-1	Moyen-Orient (EAU)
sa-east-1	Amérique du Sud (São Paulo)

Création d'un compartiment Amazon S3 pour Amazon MWAA

Ce guide décrit les étapes de création d'un compartiment Amazon S3 pour stocker vos graphes acycliques dirigés Apache Airflow (DAGs), vos plugins personnalisés dans un `plugins.zip` fichier et vos dépendances Python dans un fichier `requirements.txt`

Table des matières

- [Avant de commencer](#)
- [Créez le bucket](#)
- [Quelle est la prochaine étape ?](#)

Avant de commencer

- Le nom du compartiment Amazon S3 ne peut pas être modifié une fois que vous l'avez créé. Pour en savoir plus, consultez [les règles relatives à la dénomination des compartiments](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.
- Un compartiment Amazon S3 utilisé pour un environnement Amazon MWAA doit être configuré pour bloquer tout accès public, le contrôle de version des compartiments étant activé.
- Un compartiment Amazon S3 utilisé pour un environnement Amazon MWAA doit être situé au même endroit Région AWS qu'un environnement Amazon MWAA. Pour accéder à la Régions AWS liste des [points de terminaison et quotas Amazon MWAA destinés à Amazon MWAA](#) dans le. Références générales AWS

Créez le bucket

Cette section décrit les étapes de création du compartiment Amazon S3 pour votre environnement.

Pour créer un compartiment

1. Connectez-vous à la console Amazon S3 AWS Management Console et ouvrez-la à l'adresse <https://console.aws.amazon.com/s3/>.
2. Choisissez Créer un compartiment.
3. Dans Nom du compartiment, saisissez un nom compatible DNS pour votre compartiment.

Le nom du compartiment doit présenter les caractéristiques suivantes :

- Il doit être unique sur l'ensemble d'Amazon S3.
- Il doit comporter entre 3 et 63 caractères.
- Ne contient pas de majuscules.
- Il doit commencer par une minuscule ou un chiffre.

Important

Évitez d'inclure des informations sensibles, notamment des numéros de compte, dans le nom du compartiment. Le nom du compartiment est disponible en URLs pointant vers les objets du compartiment.

4. Choisissez une Région AWS région. Il doit être Région AWS identique à votre environnement Amazon MWAA.
 - Nous vous recommandons de choisir une région proche de chez vous afin de minimiser la latence et les coûts et de répondre aux exigences réglementaires.
5. Choisissez Block all public access (Bloquer tous les accès publics).
6. Choisissez Activer dans la gestion des versions des compartiments.
7. Facultatif - Balises. Ajoutez des paires de balises clé-valeur pour identifier votre compartiment Amazon S3 dans Tags. Par exemple, Bucket : Staging.
8. Facultatif - Chiffrement côté serveur. Vous pouvez éventuellement activer l'une des options de chiffrement suivantes sur votre compartiment Amazon S3.
 - a. Choisissez la clé Amazon S3 (SSE-S3) dans le chiffrement côté serveur pour activer le chiffrement côté serveur pour le compartiment.
 - b. Choisissez une AWS Key Management Service clé (SSE-KMS) pour utiliser une AWS KMS clé de chiffrement sur votre compartiment Amazon S3 :
 - i. AWS clé gérée (aws/s3) - Si vous choisissez cette option, vous pouvez soit utiliser une [cléAWS appartenant à Amazon MWAA, soit spécifier une clé gérée par le client pour](#) le chiffrement de votre environnement Amazon MWAA.
 - ii. Choisissez parmi vos AWS KMS clés ou entrez l'ARN de la AWS KMS clé : si vous choisissez de spécifier une [clé gérée par le client](#) à cette étape, vous devez spécifier un ID de AWS KMS clé ou un ARN. [AWS KMS les alias et les clés multirégionales ne sont pas pris en charge par Amazon MWAA](#). La AWS KMS clé que vous spécifiez doit également être utilisée pour le chiffrement de votre environnement Amazon MWAA.
9. Facultatif - Paramètres avancés. Si vous souhaitez activer Amazon S3 Object Lock :
 - a. Choisissez Paramètres avancés, puis Activer.

 Important

L'activation du verrouillage des objets permet de verrouiller définitivement les objets de ce compartiment. Pour en savoir plus, reportez-vous à la section [Verrouillage d'objets à l'aide d'Amazon S3 Object Lock](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

- b. Choisissez l'accusé de réception.

10. Choisissez Créer un compartiment.

Quelle est la prochaine étape ?

- Découvrez comment créer le réseau Amazon VPC requis pour un environnement dans. [Création du réseau VPC](#)
- Découvrez comment gérer les autorisations d'accès dans [Comment définir les autorisations du bucket ACL ?](#)
- Découvrez comment supprimer un compartiment de stockage dans [Comment supprimer un compartiment S3 ?](#).

Création du réseau VPC

Amazon Managed Workflows pour Apache Airflow nécessite un Amazon VPC et des composants réseau spécifiques pour prendre en charge un environnement. Ce guide décrit les différentes options permettant de créer le réseau Amazon VPC pour un environnement Amazon Managed Workflows for Apache Airflow.

Note

Apache Airflow fonctionne mieux dans un environnement réseau à faible latence. Si vous utilisez un Amazon VPC existant qui achemine le trafic vers une autre région ou vers un environnement sur site, nous vous recommandons d'ajouter des points de AWS PrivateLink terminaison pour Amazon SQS, CloudWatch Amazon S3 et. AWS KMS Pour plus d'informations sur la configuration AWS PrivateLink pour Amazon MWAA, consultez [Création d'un réseau Amazon VPC sans](#) accès à Internet.

Table des matières

- [Prérequis](#)
- [Avant de commencer](#)
- [Options pour créer le réseau Amazon VPC](#)
 - [Première option : créer le réseau VPC sur la console Amazon MWAA](#)
 - [Deuxième option : créer un réseau Amazon VPC avec accès à Internet](#)

- [Troisième option : créer un réseau Amazon VPC sans accès à Internet](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Avant de commencer

- Le [réseau VPC](#) que vous spécifiez pour votre environnement ne peut pas être modifié une fois l'environnement créé.
- Vous pouvez utiliser un routage privé ou public pour votre Amazon VPC et votre serveur Web Apache Airflow. Pour accéder à la liste des options, reportez-vous à [the section called “Exemples de cas d'utilisation pour un Amazon VPC et le mode d'accès Apache Airflow”](#).

Options pour créer le réseau Amazon VPC

La section suivante décrit les options disponibles pour créer le réseau Amazon VPC pour un environnement.

Note

Amazon MWAA ne prend pas en charge l'utilisation de `use1-az3` la zone de disponibilité (AZ) dans la région USA Est (Virginie du Nord). Lorsque vous créez le VPC pour Amazon MWAA dans la région USA Est (Virginie du Nord), vous devez l'attribuer explicitement `AvailabilityZone` dans le modèle CloudFormation (CFN). Le nom de zone de disponibilité attribué ne doit pas être mappé à `use1-az3`. Vous pouvez récupérer le mappage détaillé des noms d'AZ avec leur AZ correspondant IDs en exécutant la commande suivante :

```
aws ec2 describe-availability-zones --region us-east-1
```

Première option : créer le réseau VPC sur la console Amazon MWAA

La section suivante explique comment créer un réseau Amazon VPC sur la console Amazon MWAA. Cette option utilise [Routage public sur Internet](#). Il peut être utilisé pour un serveur Web Apache Airflow avec les modes d'accès réseau privé ou réseau public.

L'image suivante montre où se trouve le bouton Create MWAA VPC sur la console Amazon MWAA.

Deuxième option : créer un réseau Amazon VPC avec accès à Internet

Le CloudFormation modèle suivant crée un réseau Amazon VPC avec un accès Internet par défaut. Région AWS Cette option utilise [Routage public sur Internet](#). Ce modèle peut être utilisé pour un serveur Web Apache Airflow avec les modes d'accès réseau privé ou réseau public.

1. Copiez le contenu du modèle suivant et enregistrez-le localement sous `cf-n-vpc-public-private.yaml`. Vous pouvez également [télécharger le modèle](#).

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::VPCEGatewayAttachment

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. Dans votre invite de commande, accédez au répertoire dans lequel `cfn-vpc-public-private.yaml` est stocké. Par exemple :

```
cd mwaaproject
```

3. Utilisez la [aws cloudformation create-stack](#) commande pour créer la pile à l'aide du AWS CLI.

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

Note

La création de l'infrastructure Amazon VPC prend environ 30 minutes.

Troisième option : créer un réseau Amazon VPC sans accès à Internet

Le CloudFormation modèle suivant crée un réseau Amazon VPC sans accès à Internet par défaut.
Région AWS

Cette option utilise [Routage privé sans accès à Internet](#). Ce modèle peut être utilisé pour un serveur Web Apache Airflow avec le mode d'accès réseau privé uniquement. Il crée les [points de terminaison VPC requis pour les AWS services utilisés par](#) un environnement.

1. Copiez le contenu du modèle suivant et enregistrez-le localement sous `cfn-vpc-private.yaml`. Vous pouvez également [télécharger le modèle](#).

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

```
Parameters:
```

```
  VpcCIDR:
```

```
    Description: The IP range (CIDR notation) for this VPC
```

```
    Type: String
```

Default: 10.192.0.0/16

PrivateSubnet1CIDR:

Description: The IP range (CIDR notation) for the private subnet in the first Availability Zone

Type: String

Default: 10.192.10.0/24

PrivateSubnet2CIDR:

Description: The IP range (CIDR notation) for the private subnet in the second Availability Zone

Type: String

Default: 10.192.11.0/24

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref AWS::StackName

RouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: !Sub "\${AWS::StackName}-route-table"

PrivateSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [0, !GetAZs '']

CidrBlock: !Ref PrivateSubnet1CIDR

MapPublicIpOnLaunch: false

Tags:

- Key: Name

Value: !Sub "\${AWS::StackName} Private Subnet (AZ1)"

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

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

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

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}-mwaa-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
```

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. Dans votre invite de commande, accédez au répertoire dans lequel `cfn-vpc-private.yml` est stocké. Par exemple :

```
cd mwaaproject
```

3. Utilisez la [aws cloudformation create-stack](#) commande pour créer la pile à l'aide du AWS CLI.

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

Note

La création de l'infrastructure Amazon VPC prend environ 30 minutes.

4. Vous devez créer un mécanisme pour accéder à ces points de terminaison VPC depuis votre ordinateur. Pour en savoir plus, reportez-vous à [Gestion de l'accès aux points de terminaison Amazon VPC spécifiques à un service sur Amazon MWAA](#).

Note

Vous pouvez restreindre davantage l'accès sortant dans le CIDR de votre groupe de sécurité Amazon MWAA. Par exemple, vous pouvez vous limiter à elle-même en ajoutant une règle d'autoréférencement sortant, la liste de [préfixes pour Amazon](#) S3 et le CIDR de votre Amazon VPC.

Quelle est la prochaine étape ?

- Découvrez comment créer un environnement Amazon MWAA dans [Création d'un environnement Amazon MWAA](#).
- Découvrez comment créer un tunnel VPN entre votre ordinateur et votre Amazon VPC avec un routage privé intégré. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#)

Création d'un environnement Amazon MWAA

Amazon Managed Workflows pour Apache Airflow configure Apache Airflow dans un environnement de la version que vous avez choisie en utilisant le même Apache Airflow open source et la même interface utilisateur disponibles auprès d'Apache. Ce guide décrit les étapes de création d'un environnement Amazon MWAA.

Table des matières

- [Avant de commencer](#)
- [Versions d'Apache Airflow](#)
- [Création d'un environnement](#)
 - [Première étape : Spécifier les détails](#)
 - [Deuxième étape : configurer les paramètres avancés](#)
 - [Troisième étape : révision et création](#)

Avant de commencer

- Le [réseau VPC](#) que vous spécifiez pour votre environnement ne peut pas être modifié une fois l'environnement créé.
- Vous avez besoin d'un compartiment Amazon S3 configuré pour bloquer tout accès public, avec activation de la gestion des versions des compartiments.
- Vous avez besoin d'un Compte AWS d'une [autorisation pour utiliser Amazon MWAA](#) et d'une autorisation in Gestion des identités et des accès AWS (IAM) pour créer des rôles IAM. Si vous choisissez le mode d'accès au réseau privé pour le serveur Web Apache Airflow, qui limite l'accès à Apache Airflow au sein de votre Amazon VPC, vous aurez besoin d'une autorisation dans IAM pour créer des points de terminaison Amazon VPC.

Note

Amazon MWAA détermine le réseau de manière dynamique lors de la création. Si vous utilisez des IPv6 sous-réseaux, Amazon MWAA crée une connectivité par lien IPv6 privé vers la base de données et le serveur Web. Amazon MWAA ne prend pas en charge la transition entre les types de réseaux et ne peut pas mettre à niveau les environnements existants vers IPv6.

Versions d'Apache Airflow

Les versions d'Apache Airflow suivantes sont prises en charge sur Amazon Managed Workflows pour Apache Airflow.

Note

- À compter du 30 décembre 2025, Amazon MWAA cessera de prendre en charge les versions 2.4.3, 2.5.1 et 2.6.3 d'Apache Airflow. Pour plus d'informations, consultez [Support des versions d'Apache Airflow et FAQ](#).
- À partir de la version 2.2.2 d'Apache Airflow, Amazon MWAA prend en charge l'installation des exigences relatives à Python, des packages de fournisseurs et des plug-ins personnalisés directement sur le serveur Web Apache Airflow.
- À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. --constraint Si vous ne fournissez aucune contrainte, Amazon MWAA vous en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Pour plus d'informations sur la configuration des contraintes dans votre fichier d'exigences, reportez-vous à la section [Installation des dépendances Python](#).

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.11.0	20 mai 2025	7 janvier 2026	fichier de contraintes v2.11.0	Python 3.12
v3.0.6	29 août 2025	1er octobre 2025	fichier de contraintes v3.0.6	Python 3.12
v2.10.3	4 novembre 2024	18 décembre 2024	fichier de contraintes v2.10.3	Python 3.11

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.10.1	5 septembre 2024	26 septembre 2024	fichier de contraintes v2.10.1	Python 3.11
v2.9.2	10 juin 2024	9 juillet 2024	fichier de contraintes v2.9.2	Python 3.11
v2.8.1	19 janvier 2024	23 février 2024	fichier de contraintes v2.8.1	Python 3.11
v2.7.2	12 octobre 2023	6 novembre 2023	fichier de contraintes v2.7.2	Python 3.11

[Pour plus d'informations sur la migration de vos déploiements Apache Airflow autogérés ou sur la migration d'un environnement Amazon MWAA existant, y compris les instructions pour sauvegarder votre base de données de métadonnées, consultez le guide de migration Amazon MWAA.](#)

Création d'un environnement

La section suivante décrit les étapes de création d'un environnement Amazon MWAA.

Première étape : Spécifier les détails

Pour spécifier les détails de l'environnement

1. Ouvrez la console [Amazon MWAA](#).
2. Sélectionnez votre Région AWS.
3. Choisissez Create environment.
4. Sur la page Spécifier les détails, sous Détails de l'environnement :

- a. Entrez un nom unique pour votre environnement dans Nom.
- b. Choisissez la version Apache Airflow dans la Version Airflow.

 Note

Si aucune valeur n'est spécifiée, la dernière version d'Apache Airflow est utilisée par défaut. La dernière version disponible est Apache Airflow v3.0.6.

5. Sous le code DAG dans Amazon S3, spécifiez les éléments suivants :
 - a. Seau S3. Choisissez Browse S3 et sélectionnez votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - b. DAGs folder. Choisissez Browse S3 et sélectionnez le dags dossier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - c. Fichier de plugins : facultatif. Choisissez Browse S3 et sélectionnez le `plugins.zip` fichier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - d. Fichier des exigences : facultatif. Choisissez Browse S3 et sélectionnez le `requirements.txt` fichier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - e. Fichier de script de démarrage : facultatif. Choisissez Browse S3 et sélectionnez le fichier de script dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
6. Choisissez Suivant.

Deuxième étape : configurer les paramètres avancés

Pour configurer les paramètres avancés

1. Sur la page Configurer les paramètres avancés, sous Mise en réseau :
 - Choisissez votre [Amazon VPC](#).

Cette étape permet de remplir deux des sous-réseaux privés de votre Amazon VPC.

2. Sous Accès au serveur Web, sélectionnez votre mode d'accès [Apache Airflow](#) préféré :
 - a. Réseau privé. Cela limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement. Pour

cette étape, vous devez disposer d'une autorisation pour créer des points de terminaison Amazon VPC.

 Note

Choisissez l'option Réseau privé si l'interface utilisateur d'Apache Airflow n'est accessible qu'au sein d'un réseau d'entreprise et si vous n'avez pas besoin d'accéder aux référentiels publics pour l'installation des exigences du serveur Web. Si vous choisissez cette option de mode d'accès, vous devez créer un mécanisme pour accéder à votre serveur Web Apache Airflow dans votre Amazon VPC. Pour plus d'informations, consultez [Accès au point de terminaison VPC de votre serveur Web Apache Airflow \(accès réseau privé\)](#).

- b. Réseau public. Cela permet aux utilisateurs autorisés à accéder à l'interface utilisateur d'Apache Airflow via Internet à la [politique IAM de](#) votre environnement.
3. Sous Groupes de sécurité, choisissez le groupe de sécurité utilisé pour sécuriser votre [Amazon VPC](#) :
 - a. Par défaut, Amazon MWAA crée un groupe de sécurité dans votre Amazon VPC avec des règles d'entrée et de sortie spécifiques dans Créer un nouveau groupe de sécurité.
 - b. Facultatif. Décochez la case dans Créer un nouveau groupe de sécurité pour sélectionner jusqu'à 5 groupes de sécurité.

 Note

Un groupe de sécurité Amazon VPC existant doit être configuré avec des règles entrantes et sortantes spécifiques pour autoriser le trafic réseau. Pour en savoir plus, reportez-vous à [Sécurité de votre VPC sur Amazon MWAA](#).

4. Sous Classe d'environnement, choisissez une [classe d'environnement](#).

Nous vous recommandons de choisir la plus petite taille nécessaire pour supporter votre charge de travail. Vous pouvez modifier la classe d'environnement à tout moment.

5. Pour le nombre maximal de travailleurs, spécifiez le nombre maximum de travailleurs Apache Airflow à exécuter dans l'environnement.

Pour plus d'informations, consultez [Exemple de cas d'utilisation à hautes performances](#).

6. Spécifiez le nombre maximal de serveurs Web et le nombre minimal de serveurs Web pour configurer la manière dont Amazon MWAA adapte les serveurs Web Apache Airflow dans votre environnement.

Pour plus d'informations sur le dimensionnement automatique du serveur Web, reportez-vous à [the section called “Configuration de la mise à l'échelle automatique du serveur Web”](#).

7. Sous Chiffrement, choisissez une option de chiffrement des données :
 - a. Par défaut, Amazon MWAA utilise une clé AWS appartenant à Amazon pour chiffrer vos données.
 - b. Facultatif. Choisissez Personnaliser les paramètres de chiffrement (avancés) pour choisir une autre AWS KMS clé. Si vous choisissez de spécifier une [clé gérée par le client](#) à cette étape, vous devez spécifier un ID de AWS KMS clé ou un ARN. [AWS KMS les alias et les clés multirégionales ne sont pas pris en charge par Amazon MWAA](#). Si vous avez spécifié une clé Amazon S3 pour le chiffrement côté serveur sur votre compartiment Amazon S3, vous devez spécifier la même clé pour votre environnement Amazon MWAA.

 Note

Vous devez être autorisé à accéder à la clé pour la sélectionner sur la console Amazon MWAA. Vous devez également autoriser Amazon MWAA à utiliser la clé en joignant la politique décrite dans [Joindre une politique clé](#).

8. Recommandé Sous Surveillance, choisissez une ou plusieurs catégories de journaux pour la configuration de journalisation Airflow afin d'envoyer les journaux Apache Airflow à Logs CloudWatch :
- a. Journaux des tâches Airflow. Choisissez le type de journaux de tâches Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- b. Journaux du serveur Web Airflow. Choisissez le type de journaux du serveur Web Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- c. Journaux du planificateur de débit d'air. Choisissez le type de journaux du planificateur Apache Airflow à envoyer au niveau Logs in CloudWatch Log.
- d. Journaux des travailleurs de Airflow. Choisissez le type de journaux de travail d'Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- e. Journaux de traitement Airflow DAG. Choisissez le type de journaux de traitement du DAG Apache Airflow à envoyer au niveau CloudWatch Logs in Log.

9. Facultatif. Pour les options de configuration Airflow, choisissez l'option Ajouter une configuration personnalisée.

Vous pouvez choisir parmi la liste déroulante suggérée des options de [configuration d'Apache Airflow](#) pour votre version d'Apache Airflow, ou spécifier des options de configuration personnalisées. Par exemple, `core.default_task_retries : 3`.
10. Facultatif. Sous Balises, choisissez Ajouter une nouvelle balise pour associer des balises à votre environnement. Par exemple, `Environment : Staging`.
11. Sous Autorisations, choisissez un rôle d'exécution :
 - a. Par défaut, Amazon MWAA crée un [rôle d'exécution](#) dans Create a new role. Vous devez être autorisé à créer des rôles IAM pour utiliser cette option.
 - b. Facultatif. Choisissez Enter role ARN pour saisir le Amazon Resource Name (ARN) d'un rôle d'exécution existant.
12. Choisissez Suivant.

Troisième étape : révision et création

Pour consulter un résumé de l'environnement

- Consultez le résumé de l'environnement, puis choisissez Créer un environnement.

Note

Il faut environ vingt à trente minutes pour créer un environnement.

Quelle est la prochaine étape ?

- Découvrez comment créer un compartiment Amazon S3 dans [Création d'un compartiment Amazon S3 pour Amazon MWAA](#).

Gestion de l'accès à un environnement Amazon MWAA

Amazon Managed Workflows for Apache Airflow doit être autorisé à utiliser d'autres AWS services et ressources utilisés par un environnement. Vous devez également être autorisé à accéder à un environnement Amazon MWAA et à votre interface utilisateur Apache Airflow Gestion des identités et des accès AWS dans (IAM). Cette section décrit le rôle d'exécution utilisé pour accorder l'accès aux AWS ressources de votre environnement et explique comment ajouter des autorisations, ainsi que les autorisations dont vous avez besoin pour accéder à votre environnement Amazon MWAA et à l'interface utilisateur d'Apache Airflow. Compte AWS

Rubriques

- [Accès à un environnement Amazon MWAA](#)
- [Rôle lié à un service pour Amazon MWAA](#)
- [Rôle d'exécution Amazon MWAA](#)
- [Prévention du cas de figure de l'adjoint désorienté entre services](#)
- [Modes d'accès Apache Airflow](#)

Accès à un environnement Amazon MWAA

Pour utiliser Amazon Managed Workflows pour Apache Airflow, vous devez utiliser un compte et des entités IAM disposant des autorisations nécessaires. Cette rubrique décrit les politiques d'accès que vous pouvez appliquer à votre équipe de développement Apache Airflow et aux utilisateurs d'Apache Airflow pour votre environnement Amazon Managed Workflows for Apache Airflow.

Nous vous recommandons d'utiliser des informations d'identification temporaires et de configurer des identités fédérées avec des groupes et des rôles pour accéder à vos ressources Amazon MWAA. Il est recommandé d'éviter d'associer des politiques directement à vos utilisateurs IAM. Définissez plutôt des groupes ou des rôles pour fournir un accès temporaire aux AWS ressources.

Un [rôle](#) IAM est une identité IAM que vous pouvez créer dans votre compte et qui dispose d'autorisations spécifiques. Un rôle IAM est similaire à un utilisateur IAM dans la mesure où il s'agit d'une AWS identité dotée de politiques d'autorisation qui déterminent ce que l'identité peut et ne peut pas faire. AWS En revanche, au lieu d'être associé de manière unique à une personne, un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. En outre, un rôle ne dispose pas d'informations d'identification standard à long terme comme un mot de passe ou des clés

d'accès associées. Au lieu de cela, lorsque vous adoptez un rôle, il vous fournit des informations d'identification de sécurité temporaires pour votre session de rôle.

Pour attribuer des autorisations à une identité fédérée, vous devez créer un rôle et définir des autorisations pour ce rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center.

Vous pouvez utiliser un rôle IAM dans votre compte pour accorder à d'autres Compte AWS autorisations d'accès aux ressources de votre compte. Pour un exemple, consultez le [didacticiel IAM : Déléguer l'accès à Comptes AWS l'aide de rôles IAM](#) dans le guide de l'utilisateur IAM.

Sections

- [Comment ça marche](#)
- [Politique d'accès complète à la console : Amazon MWAAFull ConsoleAccess](#)
- [Politique complète d'accès à l'API et à la console : Amazon MWAAFull ApiAccess](#)
- [Politique d'accès à la console en lecture seule : Amazon MWAARead OnlyAccess](#)
- [Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#)
- [Politique d'accès à l'API Rest d'Apache Airflow : Amazon MWAARest APIAccess](#)
- [Politique de la CLI Apache Airflow : Amazon MWAAAirflow CliAccess](#)
- [Création d'une politique JSON](#)
- [Exemple de cas d'utilisation pour associer des politiques à un groupe de développeurs](#)
- [Quelle est la prochaine étape ?](#)

Comment ça marche

Les ressources et services utilisés dans un environnement Amazon MWAA ne sont pas accessibles à toutes les entités Gestion des identités et des accès AWS (IAM). Vous devez créer une politique qui autorise les utilisateurs d'Apache Airflow à accéder à ces ressources. Par exemple, vous devez autoriser l'accès à votre équipe de développement Apache Airflow.

Amazon MWAA utilise ces politiques pour vérifier si un utilisateur dispose des autorisations nécessaires pour effectuer une action sur la AWS console ou via un environnement APIs utilisé.

Vous pouvez utiliser les politiques JSON décrites dans cette rubrique pour créer une stratégie pour vos utilisateurs Apache Airflow dans IAM, puis associer la politique à un utilisateur, un groupe ou un rôle dans IAM.

- [Amazon MWAAFull ConsoleAccess](#) — Utilisez cette politique pour autoriser la configuration d'un environnement sur la console Amazon MWAA.
- [Amazon MWAAFull ApiAccess](#) — Utilisez cette politique pour accorder l'accès à tous les Amazon MWAA APIs utilisés pour gérer un environnement.
- [Amazon MWAARead OnlyAccess](#) — Utilisez cette politique pour accorder l'accès aux ressources utilisées par un environnement sur la console Amazon MWAA.
- [Amazon MWAAWeb ServerAccess](#) — Utilisez cette politique pour accorder l'accès au serveur Web Apache Airflow.
- [Amazon MWAAAirflow CliAccess](#) — Utilisez cette politique pour autoriser l'accès à l'exécution des commandes de la CLI Apache Airflow.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

Créez un jeu d'autorisations. Suivez les instructions de la rubrique [Création d'un jeu d'autorisations](#) du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
 - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique [Création d'un rôle pour un utilisateur IAM](#) dans le Guide de l'utilisateur IAM.
 - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique [Ajout d'autorisations à un utilisateur \(console\)](#) du Guide de l'utilisateur IAM.

Politique d'accès complète à la console : Amazon MWAAFullConsoleAccess

Un utilisateur peut avoir besoin d'accéder à la politique d'AmazonMWAAFullConsoleAccess autorisations s'il doit configurer un environnement sur la console Amazon MWAA.

Note

Votre politique d'accès à la console complète doit inclure des autorisations d'exécution `iam:PassRole`. Cela permet à l'utilisateur de transmettre des [rôles liés aux services et des rôles d'exécution](#) à Amazon MWAA. Amazon MWAA assume chaque rôle pour appeler d'autres AWS services en votre nom. L'exemple suivant utilise la clé de `iam:PassedToService` condition pour spécifier le principal de service Amazon MWAA (`airflow.amazonaws.com`) en tant que service auquel un rôle peut être transmis. Pour plus d'informations `iam:PassRole`, reportez-vous à la section [Octroi à un utilisateur des autorisations pour transmettre un rôle à un AWS service](#) dans le Guide de l'utilisateur IAM.

Appliquez la politique suivante si vous souhaitez créer et gérer vos environnements Amazon MWAA à l'aide d'un système [Clé détenue par AWS](#) de [chiffrement au repos](#).

À l'aide d'un Clé détenue par AWS

JSON

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

```

        "Resource": "*",
        "Condition": {
            "StringLike": {
                "iam:PassedToService": "airflow.amazonaws.com"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:ListRoles"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:CreatePolicy"
        ],
        "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:AttachRolePolicy",
            "iam:CreateRole"
        ],
        "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",

```

```

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

```

```

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

```

Appliquez la politique suivante si vous souhaitez créer et gérer vos environnements Amazon MWAA à l'aide d'une [clé gérée par le client](#) pour le chiffrement au repos. Pour utiliser une clé gérée par le client, le principal IAM doit être autorisé à accéder aux AWS KMS ressources à l'aide de la clé enregistrée dans votre compte.

Utilisation d'une clé gérée par le client

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {

```

```

        "StringLike": {
            "iam:PassedToService": "airflow.amazonaws.com"
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:ListRoles"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:CreatePolicy"
        ],
        "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:AttachRolePolicy",
            "iam:CreateRole"
        ],
        "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:ListBucketVersions"
        ],

```

```

        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:CreateBucket",
            "s3:PutObject",
            "s3:GetEncryptionConfiguration"
        ],
        "Resource": "arn:aws:s3:::*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:CreateSecurityGroup"
        ],
        "Resource": "arn:aws:ec2:*:*:security-group/airflow-security-group-*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:ListAliases"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:DescribeKey",
            "kms:ListGrants",
            "kms:CreateGrant",
            "kms:RevokeGrant",
            "kms:Decrypt",

```



```

        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:ReEncrypt*"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
},
{
    "Effect": "Allow",
    "Action": "ec2:CreateVpcEndpoint",
    "Resource": [
        "arn:aws:ec2:*:vpc-endpoint/*",
        "arn:aws:ec2:*:vpc/*",
        "arn:aws:ec2:*:subnet/*",
        "arn:aws:ec2:*:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:subnet/*",
        "arn:aws:ec2:*:network-interface/*"
    ]
}
]
}

```

Politique complète d'accès à l'API et à la console : Amazon MWAAFullApiAccess

Un utilisateur peut avoir besoin d'accéder à la politique AmazonMWAAFullApiAccess d'autorisations s'il a besoin d'accéder à tous les Amazon MWAA APIs utilisés pour gérer un environnement. Il n'accorde pas d'autorisations pour accéder à l'interface utilisateur d'Apache Airflow.

Note

Une politique d'accès complète à l'API doit inclure des autorisations d'exécution `iam:PassRole`. Cela permet à l'utilisateur de transmettre des [rôles liés aux services et des rôles d'exécution](#) à Amazon MWAA. Amazon MWAA assume chaque

rôle pour appeler d'autres AWS services en votre nom. L'exemple suivant utilise la clé de `iam:PassedToService` condition pour spécifier le principal de service Amazon MWAA (`airflow.amazonaws.com`) en tant que service auquel un rôle peut être transmis. Pour plus d'informations `iam:PassRole`, reportez-vous à la section [Octroi à un utilisateur des autorisations pour transmettre un rôle à un AWS service](#) dans le Guide de l'utilisateur IAM.

Appliquez la politique suivante si vous souhaitez créer et gérer vos environnements Amazon MWAA à l'aide d'un système Clé détenue par AWS de chiffrement au repos.

À l'aide d'un Clé détenue par AWS

JSON

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

```

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

```

Appliquez la politique suivante si vous souhaitez créer et gérer vos environnements Amazon MWAA à l'aide d'une clé gérée par le client pour le chiffrement au repos. Pour utiliser une clé gérée par le client, le principal IAM doit être autorisé à accéder aux AWS KMS ressources à l'aide de la clé enregistrée dans votre compte.

Utilisation d'une clé gérée par le client

JSON

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

```

        "ec2:DescribeVpcs",
        "ec2:DescribeRouteTables"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "kms:DescribeKey",
        "kms:ListGrants",
        "kms:CreateGrant",
        "kms:RevokeGrant",
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:ReEncrypt*"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetEncryptionConfiguration"
    ],
    "Resource": "arn:aws:s3:::*"
},
{
    "Effect": "Allow",
    "Action": "ec2:CreateVpcEndpoint",
    "Resource": [
        "arn:aws:ec2:*:*:vpc-endpoint/*",
        "arn:aws:ec2:*:*:vpc/*",
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:network-interface/*"
    ]
}

```

```

    ]
  }
]
}

```

Politique d'accès à la console en lecture seule : Amazon MWAARead OnlyAccess

Un utilisateur peut avoir besoin d'accéder à la politique d'AmazonMWAAReadOnlyAccess autorisations s'il a besoin d'accéder aux ressources utilisées par un environnement sur la page de détails de l'environnement de la console Amazon MWAA. Il ne permet pas à un utilisateur de créer de nouveaux environnements, de modifier des environnements existants ou d'accéder à l'interface utilisateur d'Apache Airflow.

JSON

```

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

```

Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAServerAccess

Un utilisateur peut avoir besoin d'accéder à la politique AmazonMWAAServerAccess d'autorisations pour accéder à l'interface utilisateur d'Apache Airflow. Il n'autorise pas l'utilisateur à accéder aux environnements de la console Amazon MWAA ni à utiliser l'Amazon MWAA APIs pour

effectuer des actions. Spécifiez le AdminOp, User, Viewer ou le Public rôle dans {airflow-role} pour personnaliser le niveau d'accès pour l'utilisateur du jeton Web. Pour plus d'informations, reportez-vous à la section [Rôles par défaut](#) du guide de référence d'Apache Airflow.

JSON

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

Note

- Amazon MWAA fournit une intégration IAM avec les cinq rôles de contrôle d'[accès basé sur les rôles \(RBAC\) par défaut d'Apache Airflow](#). Pour plus d'informations sur l'utilisation de rôles Apache Airflow personnalisés, reportez-vous à [the section called "Tutoriel : Restreindre les utilisateurs à un sous-ensemble de DAGs"](#)
- Le Resource champ de cette politique peut être utilisé pour spécifier les rôles de contrôle d'accès basés sur les rôles d'Apache Airflow pour l'environnement Amazon MWAA. Toutefois, il ne prend pas en charge l'ARN (Amazon Resource Name) de l'environnement Amazon MWAA dans le Resource champ de la politique.

Politique d'accès à l'API Rest d'Apache Airflow : Amazon MWAA Rest APIAccess

Pour accéder à l'API REST Apache Airflow, vous devez accorder l'`airflow:InvokeRestApi` autorisation dans votre politique IAM. Dans l'exemple de politique suivant, spécifiez le `AdminOp`, `User`, `Viewer` ou le `Public` rôle dans `{airflow-role}` pour personnaliser le niveau d'accès des utilisateurs. Pour plus d'informations, reportez-vous à la section [Rôles par défaut](#) du guide de référence d'Apache Airflow.

JSON

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

Note

- Lors de la configuration d'un serveur Web privé, l'`InvokeRestApi` action ne peut pas être invoquée depuis l'extérieur d'un Virtual Private Cloud (VPC). Vous pouvez utiliser la `aws:SourceVpc` clé pour appliquer un contrôle d'accès plus précis à cette opération. Pour plus d'informations, consultez [aws : SourceVpc](#)
- Le `Resource` champ de cette politique peut être utilisé pour spécifier les rôles de contrôle d'accès basés sur les rôles d'Apache Airflow pour l'environnement Amazon MWAA. Toutefois, il ne prend pas en charge l'ARN (Amazon Resource Name) de l'environnement Amazon MWAA dans le `Resource` champ de la politique.

Politique de la CLI Apache Airflow : Amazon MWAAirflow CliAccess

Un utilisateur peut avoir besoin d'accéder à la politique d'AmazonMWAAirflowCliAccess autorisations s'il doit exécuter des commandes CLI d'Apache Airflow (telles que `trigger_dag`). Il n'autorise pas l'utilisateur à accéder aux environnements de la console Amazon MWAA ni à utiliser l'Amazon MWAA APIs pour effectuer des actions.

JSON

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

Création d'une politique JSON

Vous pouvez créer la politique JSON et l'associer à votre utilisateur, rôle ou groupe sur la console IAM. Les étapes suivantes décrivent comment créer une politique JSON dans IAM.

Pour créer la politique JSON

1. Ouvrez la [page Politiques](#) sur la console IAM.
2. Choisissez Create Policy (Créer une politique).
3. Choisissez l'onglet JSON.
4. Ajoutez votre politique JSON.
5. Choisissez Examiner une politique.
6. Entrez une valeur dans le champ de texte pour le nom et la description (facultatif).

Par exemple, vous pouvez donner un nom à la politique `AmazonMWAAReadOnlyAccess`.

7. Choisissez Create Policy (Créer une politique).

Exemple de cas d'utilisation pour associer des politiques à un groupe de développeurs

Supposons que vous utilisiez un groupe dans IAM nommé `AirflowDevelopmentGroup` pour appliquer des autorisations à tous les développeurs de votre équipe de développement Apache Airflow. Ces utilisateurs doivent avoir accès aux politiques de `AmazonMWAAFullConsoleAccess`, `AmazonMWAAirflowCliAccess`, et `AmazonMWAAWebServerAccess` d'autorisation. Cette section décrit comment créer un groupe dans IAM, créer et associer ces politiques, et comment associer le groupe à un utilisateur IAM. Les étapes supposent que vous utilisez une [cléAWS qui vous appartient](#).

Pour créer la `MWAAFull ConsoleAccess` politique Amazon

1. Téléchargez la [politique d'MWAAFullConsoleAccess accès d'Amazon](#).
2. Ouvrez la [page Politiques](#) sur la console IAM.
3. Choisissez Create Policy (Créer une politique).
4. Choisissez l'onglet JSON.
5. Collez la politique JSON pour `AmazonMWAAFullConsoleAccess`.
6. Remplacez les valeurs suivantes :
 - a. `123456789012`— Votre Compte AWS identifiant (tel que `0123456789`)
 - b. `{your-kms-id}`— L'identifiant unique d'une clé gérée par le client, applicable uniquement si vous utilisez une clé gérée par le client pour le chiffrement au repos.
7. Choisissez la politique de révision.
8. Tapez `AmazonMWAAFullConsoleAccess` le nom.
9. Choisissez Create Policy (Créer une politique).

Pour créer la `MWAAWeb ServerAccess` politique Amazon

1. Téléchargez la [politique d'MWAAWebServerAccess accès d'Amazon](#).
2. Ouvrez la [page Politiques](#) sur la console IAM.
3. Choisissez Create Policy (Créer une politique).

4. Choisissez l'onglet JSON.
5. Collez la politique JSON pour AmazonMWAAServerAccess.
6. Remplacez les valeurs suivantes :
 - a. `us-east-1`— la région de votre environnement Amazon MWAAServerAccess (telle que `us-east-1`)
 - b. `123456789012`— votre Compte AWS identifiant (tel que `0123456789`)
 - c. `{your-environment-name}`— le nom de votre environnement Amazon MWAAServerAccess (tel que `MyAirflowEnvironment`)
 - d. `{airflow-role}`— [le rôle par défaut d'Apache Airflow](#)
7. Choisissez Examiner une politique.
8. Tapez AmazonMWAAServerAccess le nom.
9. Choisissez Create Policy (Créer une politique).

Pour créer la MWAAAirflowCliAccess politique Amazon

1. Téléchargez la [politique d'MWAAAirflowCliAccess accès d'Amazon](#).
2. Ouvrez la [page Politiques](#) sur la console IAM.
3. Choisissez Create Policy (Créer une politique).
4. Choisissez l'onglet JSON.
5. Collez la politique JSON pour AmazonMWAAirflowCliAccess.
6. Choisissez la politique de révision.
7. Tapez AmazonMWAAirflowCliAccess le nom.
8. Choisissez Create Policy (Créer une politique).

Pour créer le groupe

1. Ouvrez la [page Groupes](#) sur la console IAM.
2. Entrez le nom de AirflowDevelopmentGroup.
3. Choisissez Étape suivante.
4. Tapez AmazonMWAA pour filtrer les résultats dans Filtrer.
5. Sélectionnez les trois politiques que vous avez créées.
6. Choisissez Étape suivante.
7. Choisissez Create Group.

Pour l'associer à un utilisateur

1. Ouvrez la [page Utilisateurs](#) sur la console IAM.
2. Choisissez un utilisateur.
3. Choisissez Groupes.
4. Choisissez Ajouter l'utilisateur aux groupes.
5. Sélectionnez la AirflowDevelopmentGroup.
6. Choisissez Add to Groups (Ajouter aux groupes).

Quelle est la prochaine étape ?

- Découvrez comment générer un jeton pour accéder à l'interface utilisateur d'Apache Airflow dans [Accès à Apache Airflow](#)
- Pour en savoir plus sur la création de politiques IAM, consultez la section [Création de politiques IAM](#).

Rôle lié à un service pour Amazon MWAA

[Amazon Managed Workflows pour Apache Airflow utilise des rôles liés à un Gestion des identités et des accès AWS service \(IAM\)](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon MWAA. Les rôles liés à un service sont prédéfinis par Amazon MWAA et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon MWAA, car il n'est pas nécessaire d'ajouter manuellement les autorisations nécessaires. Amazon MWAA définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon MWAA peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon MWAA car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés aux services, reportez-vous à la section [AWS Services qui fonctionnent avec IAM](#) et recherchez les services dont la

valeur est Oui dans la colonne Rôles liés aux services. Cliquez sur Oui avec un lien pour accéder à la documentation des rôles liés au service correspondant à ce service.

Autorisations de rôle liées à un service pour Amazon MWAA

Amazon MWAA utilise le rôle lié au service nommé `AWSServiceRoleForAmazonMWAA` — Le rôle lié au service créé dans votre compte permet à Amazon MWAA d'accéder aux services suivants : AWS

- Amazon CloudWatch Logs (CloudWatch Logs) — Pour créer des groupes de journaux pour les journaux Apache Airflow.
- Amazon CloudWatch (CloudWatch) — Pour publier des statistiques relatives à votre environnement et à ses composants sous-jacents sur votre compte.
- Amazon Elastic Compute Cloud (Amazon EC2) — Pour créer les ressources suivantes :
 - Un point de terminaison Amazon VPC dans votre VPC pour un cluster de bases de données Amazon AWS Aurora PostgreSQL géré à utiliser par le planificateur et le programme de travail Apache Airflow.
 - Un point de terminaison Amazon VPC supplémentaire pour permettre l'accès réseau au serveur Web si vous choisissez l'option [réseau privé](#) pour votre serveur Web Apache Airflow.
 - [Interfaces réseau élastiques \(ENIs\)](#) dans votre Amazon VPC pour permettre l'accès réseau aux AWS ressources hébergées dans votre Amazon VPC.

La politique de confiance suivante permet au directeur du service d'assumer le rôle lié au service. Le principe du service pour Amazon MWAA est `airflow.amazonaws.com` tel que démontré par la politique.

JSON

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

```
}
```

La politique d'autorisations de rôle nommée `AmazonMWAAServiceRolePolicy` permet à Amazon MWAA d'effectuer les actions suivantes sur les ressources spécifiées :

JSON

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

```

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

```

```
{
  "Effect": "Allow",
  "Action": "cloudwatch:PutMetricData",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "cloudwatch:namespace": [
        "AWS/MWAA"
      ]
    }
  }
}
```

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, reportez-vous à la section [Autorisations relatives aux rôles liés à un service](#) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon MWAA

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un nouvel environnement Amazon MWAA à l'aide de, de AWS Management Console AWS CLI, ou de l'AWS API, Amazon MWAA crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un autre environnement, Amazon MWAA crée à nouveau le rôle lié au service pour vous.

Modification d'un rôle lié à un service pour Amazon MWAA

Amazon MWAA n'autorise pas la modification du rôle lié au service AWSService RoleForAmazon MWAA. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, reportez-vous à la section [Modification d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Supprimer un rôle lié à un service pour Amazon MWAA

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement.

Lorsque vous supprimez un environnement Amazon MWAA, Amazon MWAA supprime toutes les ressources associées qu'il utilise dans le cadre du service. Toutefois, vous devez attendre qu'Amazon MWAA ait terminé de supprimer votre environnement avant de tenter de supprimer le rôle lié au service. Si vous supprimez le rôle lié au service avant qu'Amazon MWAA ne supprime l'environnement, Amazon MWAA risque de ne pas être en mesure de supprimer toutes les ressources associées à l'environnement.

Pour supprimer manuellement le rôle lié au service à l'aide d'IAM

Utilisez la console IAM AWS CLI, le ou l' AWS API pour supprimer le rôle lié au service AWSServiceRoleForAmazon MWAA. Pour plus d'informations, reportez-vous à la section [Suppression d'un rôle lié à un service](#) dans le guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés aux services Amazon MWAA

Amazon MWAA prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez [Amazon Managed Workflows pour les points de terminaison et les quotas Apache Airflow](#).

Mises à jour des politiques

Modifier	Description	Date
Amazon MWAA met à jour sa politique d'autorisation des rôles liés à un service	AmazonMWAAServiceRolePolicy — Amazon MWAA met à jour la politique d'autorisation relative à son rôle lié à un service afin d'autoriser Amazon MWAA à publier des statistiques supplémentaires relatives aux ressources sous-jacentes	18 novembre 2022

Modifier	Description	Date
	du service sur les comptes clients. Ces nouvelles mesures sont publiées dans le AWS/MWAA	
Amazon MWAA a commencé à suivre les modifications	Amazon MWAA a commencé à suivre les modifications apportées à sa politique d'AWS autorisation relative aux rôles liés aux services gérés.	18 novembre 2022

Rôle d'exécution Amazon MWAA

Un rôle d'exécution est un rôle Gestion des identités et des accès AWS (IAM) doté d'une politique d'autorisations qui autorise Amazon Managed Workflows for Apache Airflow à invoquer les ressources d'AWS autres services en votre nom. Cela peut inclure des ressources telles que votre compartiment Amazon S3, votre [cléAWS détenue](#) et vos CloudWatch journaux. Les environnements Amazon MWAA ont besoin d'un rôle d'exécution par environnement. Cette rubrique décrit comment utiliser et configurer le rôle d'exécution pour votre environnement afin de permettre à Amazon MWAA d'accéder à d'autres AWS ressources utilisées par votre environnement.

Table des matières

- [Vue d'ensemble des rôles d'exécution](#)
 - [Autorisations attachées par défaut](#)
 - [Comment ajouter l'autorisation d'utiliser d'autres AWS services](#)
 - [Comment associer un nouveau rôle d'exécution](#)
- [Créer un rôle](#)
- [Accéder à une politique de rôle d'exécution et la mettre à jour](#)
 - [Joindre une politique JSON pour utiliser d'autres AWS services](#)
- [Accorder l'accès au compartiment Amazon S3 avec un blocage d'accès public au niveau du compte](#)
- [Utiliser les connexions Apache Airflow](#)
- [Exemples de politiques JSON pour un rôle d'exécution](#)

- [Exemple de politique pour une clé gérée par le client](#)
- [Exemple de politique pour une clé AWS détenue par un propriétaire](#)
- [Quelle est la prochaine étape ?](#)

Vue d'ensemble des rôles d'exécution

L'autorisation accordée à Amazon MWAA d'utiliser d'autres AWS services utilisés par votre environnement provient du rôle d'exécution. Un rôle d'exécution Amazon MWAA doit être autorisé à accéder aux AWS services suivants utilisés par un environnement :

- Amazon CloudWatch (CloudWatch) — pour envoyer les métriques et les journaux d'Apache Airflow.
- Amazon Simple Storage Service (Amazon S3) : pour analyser le code DAG de votre environnement et les fichiers de support (tels que `a). requirements.txt`)
- Amazon Simple Queue Service (Amazon SQS) : pour mettre en file d'attente les tâches Apache Airflow de votre environnement dans une file d'attente Amazon SQS appartenant à Amazon MWAA.
- AWS Key Management Service (AWS KMS) — pour le chiffrement des données de votre environnement (à l'aide d'une [clé AWS détenue](#) ou [gérée par le client](#)).

Note

Si vous avez choisi qu'Amazon MWAA utilise une clé KMS AWS détenue pour chiffrer vos données, vous devez définir des autorisations dans une politique attachée à votre rôle d'exécution Amazon MWAA qui autorise l'accès à des clés KMS arbitraires stockées en dehors de votre compte via Amazon SQS. Les deux conditions suivantes sont requises pour que le rôle d'exécution de votre environnement puisse accéder à des clés KMS arbitraires :

- Une clé KMS dans un compte tiers doit autoriser cet accès entre comptes via sa politique de ressources.
 - Votre code DAG doit accéder à une file d'attente Amazon SQS qui commence `airflow-celery-` par le compte tiers et utilise la même clé KMS pour le chiffrement.
- Pour atténuer les risques associés à l'accès aux ressources entre comptes, nous vous recommandons de consulter le code placé dans votre dossier afin de vous assurer que vos flux de travail n'accèdent pas DAGs à des files d'attente Amazon SQS arbitraires en dehors

de votre compte. En outre, vous pouvez utiliser une clé KMS gérée par le client et stockée dans votre propre compte pour gérer le chiffrement sur Amazon MWAA. Cela limite le rôle d'exécution de votre environnement à accéder uniquement à la clé KMS de votre compte. N'oubliez pas qu'une fois que vous avez choisi une option de chiffrement, vous ne pouvez pas modifier votre sélection pour un environnement existant.

Un rôle d'exécution doit également être autorisé à effectuer les actions IAM suivantes :

- `airflow:PublishMetrics`— pour permettre à Amazon MWAA de surveiller l'état de santé d'un environnement.

Autorisations attachées par défaut

Vous pouvez utiliser les options par défaut de la console Amazon MWAA pour créer un rôle d'exécution et une [cléAWS détenue](#), puis suivre les étapes de cette page pour ajouter des politiques d'autorisation à votre rôle d'exécution.

- Lorsque vous choisissez l'option Créer un nouveau rôle sur la console, Amazon MWAA attache les autorisations minimales requises par un environnement à votre rôle d'exécution.
- Dans certains cas, Amazon MWAA attache le maximum d'autorisations. Par exemple, nous vous recommandons de choisir l'option sur la console Amazon MWAA pour créer un rôle d'exécution lorsque vous créez un environnement. Amazon MWAA ajoute automatiquement les politiques d'autorisation pour tous les groupes de CloudWatch journaux en utilisant le modèle regex dans le rôle d'exécution `as. "arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-name - *"`

Comment ajouter l'autorisation d'utiliser d'autres AWS services

Amazon MWAA ne peut pas ajouter ou modifier des politiques d'autorisation à un rôle d'exécution existant après la création d'un environnement. Vous devez mettre à jour votre rôle d'exécution avec les politiques d'autorisation supplémentaires requises par votre environnement. Par exemple, si votre DAG a besoin d'accéder à AWS Glue, Amazon MWAA ne peut pas détecter automatiquement que ces autorisations sont requises par votre environnement, ni les ajouter à votre rôle d'exécution.

Vous pouvez ajouter des autorisations à un rôle d'exécution de deux manières :

- En modifiant la politique JSON pour votre rôle d'exécution en ligne. Vous pouvez utiliser les exemples de [documents de stratégie JSON](#) présentés sur cette page pour compléter ou remplacer la politique JSON de votre rôle d'exécution sur la console IAM.
- En créant une politique JSON pour un AWS service et en l'associant à votre rôle d'exécution. Vous pouvez suivre les étapes de cette page pour associer un nouveau document de politique JSON pour un AWS service à votre rôle d'exécution sur la console IAM.

En supposant que le rôle d'exécution soit déjà associé à votre environnement, Amazon MWAA peut commencer à utiliser les politiques d'autorisation ajoutées immédiatement. Cela signifie également que si vous supprimez les autorisations requises d'un rôle d'exécution, vos DAGs risquent d'échouer.

Comment associer un nouveau rôle d'exécution

Vous pouvez modifier le rôle d'exécution de votre environnement à tout moment. Si aucun nouveau rôle d'exécution n'est déjà associé à votre environnement, suivez les étapes de cette page pour créer une nouvelle politique de rôle d'exécution et associez le rôle à votre environnement.

Créer un rôle

Par défaut, Amazon MWAA crée une [cléAWS détenue](#) pour le chiffrement des données et un rôle d'exécution en votre nom. Vous pouvez choisir les options par défaut sur la console Amazon MWAA lorsque vous créez un environnement. L'image suivante montre l'option par défaut permettant de créer un rôle d'exécution pour un environnement.

Important

Lorsque vous créez un nouveau rôle d'exécution, ne réutilisez pas le nom d'un rôle d'exécution supprimé. Les noms uniques peuvent contribuer à prévenir les conflits et à garantir une gestion appropriée des ressources.

Accéder à une politique de rôle d'exécution et la mettre à jour

Vous pouvez accéder au rôle d'exécution de votre environnement sur la console Amazon MWAA et mettre à jour la politique JSON pour le rôle sur la console IAM.

Pour mettre à jour une politique de rôle d'exécution

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez le rôle d'exécution dans le volet Autorisations pour ouvrir la page des autorisations dans IAM.
4. Choisissez le nom du rôle d'exécution pour ouvrir la politique d'autorisations.
5. Choisissez Modifier la politique.
6. Choisissez l'onglet JSON.
7. Mettez à jour votre politique JSON.
8. Choisissez Examiner une politique.
9. Sélectionnez Enregistrer les modifications.

Joindre une politique JSON pour utiliser d'autres AWS services

Vous pouvez créer une politique JSON pour un AWS service et l'associer à votre rôle d'exécution. Par exemple, vous pouvez joindre la politique JSON suivante pour accorder un accès en lecture seule à toutes les ressources de AWS Secrets Manager

JSON

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

```
}
```

Pour associer une politique à votre rôle d'exécution

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez votre rôle d'exécution dans le volet Autorisations.
4. Choisissez Attacher des politiques.
5. Choisissez Create Policy (Créer une politique).
6. Choisissez JSON.
7. Collez la politique JSON.
8. Choisissez Suivant : Tags, Suivant : Révision.
9. Entrez un nom descriptif (tel que `SecretsManagerReadPolicy`) et une description pour la politique.
10. Choisissez Create Policy (Créer une politique).

Accorder l'accès au compartiment Amazon S3 avec un blocage d'accès public au niveau du compte

Vous souhaitez peut-être bloquer l'accès à tous les compartiments de votre compte en utilisant l'opération [PutPublicAccessBlock](#) Amazon S3. Lorsque vous bloquez l'accès à tous les compartiments de votre compte, votre rôle d'exécution de l'environnement doit inclure `s3:GetAccountPublicAccessBlock` dans une politique d'autorisation.

L'exemple suivant illustre la politique que vous devez associer à votre rôle d'exécution lorsque vous bloquez l'accès à tous les compartiments Amazon S3 de votre compte.

JSON

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

```
    "Resource": "*"
  }
]
}
```

Pour plus d'informations sur la restriction de l'accès à vos compartiments Amazon S3, consultez la section [Blocage de l'accès public à votre espace de stockage Amazon S3](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Utiliser les connexions Apache Airflow

Vous pouvez également créer une connexion Apache Airflow et spécifier votre rôle d'exécution et son ARN dans votre objet de connexion Apache Airflow. Pour en savoir plus, reportez-vous à [Gestion des connexions à Apache Airflow](#).

Exemples de politiques JSON pour un rôle d'exécution

Vous pouvez utiliser les deux exemples de politiques d'autorisation de cette section pour remplacer la politique d'autorisation utilisée pour votre rôle d'exécution existant, ou pour créer un nouveau rôle d'exécution à utiliser pour votre environnement. Ces politiques contiennent des espaces réservés [Resource ARN](#) pour les groupes de journaux Apache Airflow, un compartiment [Amazon S3](#) et un environnement [Amazon MWAA](#).

Nous vous recommandons de copier l'exemple de politique, de remplacer l'exemple ARNs ou les espaces réservés, puis d'utiliser la politique JSON pour créer ou mettre à jour un rôle d'exécution.

Exemple de politique pour une clé gérée par le client

L'exemple suivant présente une politique de rôle d'exécution que vous pouvez utiliser pour une clé [gérée par le client](#).

JSON

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



```

    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ],
},
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
  ],
  "Resource": [
    "arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-
name:*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "logs:DescribeLogGroups"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Effect": "Allow",
  "Action": [

```

```

    "s3:GetAccountPublicAccessBlock"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Effect": "Allow",
  "Action": "cloudwatch:PutMetricData",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",
    "sqs:GetQueueAttributes",
    "sqs:GetQueueUrl",
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
  ],
  "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
},
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:GenerateDataKey*",
    "kms:Encrypt"
  ],
  "Resource": "arn:aws:kms:us-east-1:111122223333:key/your-kms-cmk-id",
  "Condition": {
    "StringLike": {
      "kms:ViaService": [
        "sqs.us-east-1.amazonaws.com",
        "s3.us-east-1.amazonaws.com"
      ]
    }
  }
}
]
}

```

Ensuite, vous devez autoriser Amazon MWAA à assumer ce rôle pour effectuer des actions en votre nom. Cela peut être fait en ajoutant "airflow.amazonaws.com" des principes de "airflow-env.amazonaws.com" service à la liste des entités fiables pour ce rôle d'exécution à l'[aide de la console IAM](#), ou en plaçant ces principaux de service dans le document de politique d'assume le rôle pour ce rôle d'exécution via la commande IAM [create-role](#) à l'aide du. AWS CLI Reportez-vous à l'exemple de document de politique d'acceptation des rôles suivant :

JSON

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

Attachez ensuite la politique JSON suivante à votre clé [gérée par le client](#). Cette politique utilise le préfixe de clé de [kms:EncryptionContext](#) condition pour autoriser l'accès à votre groupe de journaux Apache Airflow dans Logs. CloudWatch

```
{
  "Sid": "Allow logs access",
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
}
```

```

"Condition": {
  "ArnLike": {
    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:*"
  }
}
}

```

Exemple de politique pour une clé AWS détenue par un propriétaire

L'exemple suivant présente une politique de rôle d'exécution que vous pouvez utiliser pour une [cléAWS appartenant à](#) une personne.

JSON

```

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

```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents",
        "logs:GetLogEvents",
        "logs:GetLogRecord",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:airflow-{your-environment-name}-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "cloudwatch:PutMetricData",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ChangeMessageVisibility",

```

```

        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
},
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey*",
        "kms:Encrypt"
    ],
    "NotResource": "arn:aws:kms:*:111122223333:key/*",
    "Condition": {
        "StringLike": {
            "kms:ViaService": [
                "sqs.us-east-1.amazonaws.com"
            ]
        }
    }
}
]
}

```

Quelle est la prochaine étape ?

- Découvrez les autorisations requises dont vous et les utilisateurs d'Apache Airflow avez besoin pour accéder à votre environnement. [Accès à un environnement Amazon MWAA](#)
- En savoir plus sur [Utilisation de clés gérées par le client pour le chiffrement](#).
- Découvrez d'autres exemples de [politiques gérées par le client](#).

Prévention du cas de figure de l'adjoint désorienté entre services

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS,

l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service d'appel peut être manipulé pour utiliser ses autorisations afin d'agir sur les ressources d'un autre client d'une manière à laquelle il n'est pas autorisé à accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services auprès des principaux fournisseurs de services qui ont obtenu l'accès aux ressources de votre compte.

Nous vous recommandons d'utiliser les clés de contexte de condition [aws:SourceAccount](#) globale [aws:SourceArn](#) et les clés de contexte dans le rôle d'exécution de votre environnement afin de limiter les autorisations accordées par Amazon MWAA à un autre service pour accéder à la ressource. Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource ou si vous spécifiez plusieurs ressources, utilisez la clé de contexte de condition globale `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:airflow:*:123456789012:environment/*`.

La valeur de `aws:SourceArn` doit être l'ARN de votre environnement Amazon MWAA, pour lequel vous créez un rôle d'exécution.

Utilisez l'exemple suivant pour appliquer les clés de contexte `aws:SourceArn` et les clés de contexte de condition `aws:SourceAccount` globale dans la politique de confiance des rôles d'exécution de votre environnement afin d'éviter le problème de confusion des adjoints. Vous pouvez utiliser la politique de confiance suivante lorsque vous créez un nouveau rôle d'exécution.

JSON

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

```

        "airflow.amazonaws.com",
        "airflow-env.amazonaws.com"
    ],
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:airflow:us-
east-1:123456789012:environment/your-environment-name"
        },
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        }
    }
}
]
}

```

Modes d'accès Apache Airflow

La console Amazon Managed Workflows for Apache Airflow contient des options intégrées permettant de configurer le routage privé ou public vers le serveur Web Apache Airflow de votre environnement. Ce guide décrit les modes d'accès disponibles pour le serveur Web Apache Airflow sur votre environnement Amazon Managed Workflows for Apache Airflow, ainsi que les ressources supplémentaires que vous devrez configurer dans votre Amazon VPC si vous choisissez l'option réseau privé.

Table des matières

- [Modes d'accès Apache Airflow](#)
 - [Réseau public](#)
 - [réseau privé](#)
- [Vue d'ensemble des modes d'accès](#)
 - [Mode d'accès au réseau public](#)
 - [Mode d'accès au réseau privé](#)
- [Configuration pour les modes d'accès privé et public](#)
 - [Configuration pour le réseau public](#)
 - [Configuration pour le réseau privé](#)

- [Accès au point de terminaison VPC de votre serveur Web Apache Airflow \(accès réseau privé\)](#)

Modes d'accès Apache Airflow

Vous pouvez choisir un routage privé ou public pour votre serveur Web Apache Airflow. Pour activer le routage privé, choisissez Réseau privé. Cela limite l'accès des utilisateurs à un serveur Web Apache Airflow au sein d'un Amazon VPC. Pour activer le routage public, choisissez Réseau public. Cela permet aux utilisateurs d'accéder au serveur Web Apache Airflow via Internet.

Réseau public

Le schéma architectural suivant décrit un environnement Amazon MWAA avec un serveur Web public.

Le mode d'accès au réseau public permet aux utilisateurs autorisés à accéder à la [politique IAM](#) de votre environnement via Internet à l'interface utilisateur d'Apache Airflow.

L'image suivante indique où trouver l'option Réseau public sur la console Amazon MWAA.

réseau privé

Le schéma architectural suivant décrit un environnement Amazon MWAA avec un serveur Web privé.

Le mode d'accès au réseau privé limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement.

Lorsque vous créez un environnement avec accès privé à un serveur Web, vous devez empaqueter toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans votre `requirements.txt`. Pour obtenir des instructions sur l'empaquetage et l'installation de vos dépendances à l'aide de wheel, reportez-vous à [la section Gestion des dépendances à l'aide de Python Wheel](#).

L'image suivante indique où trouver l'option Réseau privé sur la console Amazon MWAA.

Vue d'ensemble des modes d'accès

Cette section décrit les points de terminaison VPC (AWS PrivateLink) créés dans votre Amazon VPC lorsque vous choisissez le mode d'accès au réseau public ou au réseau privé.

Mode d'accès au réseau public

Si vous avez choisi le mode d'accès au réseau public pour votre serveur Web Apache Airflow, le trafic réseau est routé publiquement sur Internet.

- Amazon MWAA crée un point de terminaison d'interface VPC pour votre base de données de métadonnées Amazon Aurora PostgreSQL. Le point de terminaison est créé dans les zones de disponibilité mappées à vos sous-réseaux privés et est indépendant des autres. Comptes AWS
- Amazon MWAA lie ensuite une adresse IP de vos sous-réseaux privés aux points de terminaison de l'interface. Ceci est conçu pour soutenir la meilleure pratique qui consiste à lier une adresse IP unique à chaque zone de disponibilité d'Amazon VPC.

Mode d'accès au réseau privé

Si vous avez choisi le mode d'accès réseau privé pour votre serveur Web Apache Airflow, le trafic réseau est acheminé de manière privée au sein de votre Amazon VPC.

- Amazon MWAA crée un point de terminaison d'interface VPC pour votre serveur Web Apache Airflow et un point de terminaison d'interface pour votre base de données de métadonnées Amazon Aurora PostgreSQL. Les points de terminaison sont créés dans les zones de disponibilité mappées à vos sous-réseaux privés et sont indépendants des autres. Comptes AWS
- Amazon MWAA lie ensuite une adresse IP de vos sous-réseaux privés aux points de terminaison de l'interface. Ceci est conçu pour soutenir la meilleure pratique qui consiste à lier une adresse IP unique à chaque zone de disponibilité d'Amazon VPC.

Pour en savoir plus, reportez-vous à [the section called “Exemples de cas d'utilisation pour un Amazon VPC et le mode d'accès Apache Airflow”](#).

Configuration pour les modes d'accès privé et public

La section suivante décrit l'installation et les configurations supplémentaires dont vous aurez besoin en fonction du mode d'accès Apache Airflow que vous avez choisi pour votre environnement.

Configuration pour le réseau public

Si vous choisissez l'option Réseau public pour votre serveur Web Apache Airflow, vous pouvez commencer à utiliser l'interface utilisateur d'Apache Airflow après avoir créé votre environnement.

Vous devrez suivre les étapes suivantes pour configurer l'accès de vos utilisateurs et l'autorisation pour votre environnement d'utiliser d'autres AWS services.

1. Ajoutez des autorisations. Amazon MWAA a besoin d'une autorisation pour utiliser d'autres AWS services. Lorsque vous créez un environnement, Amazon MWAA crée un [rôle lié à un service](#) qui lui permet d'utiliser certaines actions IAM pour Amazon Elastic Container Registry (Amazon ECR), Logs et Amazon EC2 CloudWatch .

Vous pouvez ajouter l'autorisation d'utiliser des actions supplémentaires pour ces services, ou d'utiliser d'autres AWS services en ajoutant des autorisations à votre rôle d'exécution. Pour en savoir plus, reportez-vous à [Rôle d'exécution Amazon MWAA](#).

2. Créez des politiques utilisateur. Vous devrez peut-être créer plusieurs politiques IAM pour vos utilisateurs afin de configurer l'accès à votre environnement et à l'interface utilisateur d'Apache Airflow. Pour en savoir plus, reportez-vous à [Accès à un environnement Amazon MWAA](#).

Configuration pour le réseau privé

Si vous choisissez l'option Réseau privé pour votre serveur Web Apache Airflow, vous devez configurer l'accès pour vos utilisateurs, autoriser votre environnement à utiliser d'autres AWS services et créer un mécanisme pour accéder aux ressources de votre Amazon VPC depuis votre ordinateur.

1. Ajoutez des autorisations. Amazon MWAA a besoin d'une autorisation pour utiliser d'autres AWS services. Lorsque vous créez un environnement, Amazon MWAA crée un [rôle lié à un service](#) qui lui permet d'utiliser certaines actions IAM pour Amazon Elastic Container Registry (Amazon ECR), Logs et Amazon EC2 CloudWatch .

Vous pouvez ajouter l'autorisation d'utiliser des actions supplémentaires pour ces services, ou d'utiliser d'autres AWS services en ajoutant des autorisations à votre rôle d'exécution. Pour en savoir plus, reportez-vous à [Rôle d'exécution Amazon MWAA](#).

2. Créez des politiques utilisateur. Vous devrez peut-être créer plusieurs politiques IAM pour vos utilisateurs afin de configurer l'accès à votre environnement et à l'interface utilisateur d'Apache Airflow. Pour en savoir plus, reportez-vous à [Accès à un environnement Amazon MWAA](#).

3. Activez l'accès au réseau. Vous devez créer un mécanisme dans votre Amazon VPC pour vous connecter au point de terminaison VPC (AWS PrivateLink) de votre serveur Web Apache Airflow. Par exemple, en créant un tunnel VPN à partir de votre ordinateur à l'aide d'un AWS Client VPN.

Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)

Si vous avez choisi l'option Réseau privé, vous devez créer un mécanisme dans votre Amazon VPC pour accéder au point de terminaison VPC (AWS PrivateLink) de votre serveur Web Apache Airflow. Nous vous recommandons d'utiliser le même Amazon VPC, le même groupe de sécurité VPC et les mêmes sous-réseaux privés que votre environnement Amazon MWAA pour ces ressources.

Pour en savoir plus, reportez-vous à la section [Gestion de l'accès pour les points de terminaison VPC](#).

Accès à Apache Airflow

Amazon MWAA vous permet d'accéder à votre environnement Apache Airflow à l'aide de plusieurs méthodes : la console d'interface utilisateur (UI) Apache Airflow, la CLI Apache Airflow et l'API REST Apache Airflow. Vous pouvez utiliser la console Amazon MWAA pour accéder à un DAG et l'invoquer dans votre interface utilisateur Apache Airflow, ou utiliser Amazon APIs MWAA pour obtenir un jeton et invoquer un DAG. Cette section décrit les autorisations nécessaires pour accéder à l'interface utilisateur d'Apache Airflow, comment générer un jeton pour effectuer des appels d'API Amazon MWAA directement dans votre interface de commande et les commandes prises en charge dans l'interface de ligne de commande d'Apache Airflow.

Rubriques

- [Conditions préalables](#)
- [Ouvrez l'interface utilisateur d'Apache Airflow](#)
- [Connectez-vous à Apache Airflow](#)
- [Création d'un jeton d'accès au serveur Web Apache Airflow](#)
- [Configuration d'un domaine personnalisé pour le serveur Web Apache Airflow](#)
- [Création d'un jeton CLI Apache Airflow](#)
- [Utilisation de l'API REST Apache Airflow](#)
- [Référence des commandes de la CLI Apache Airflow](#)

Conditions préalables

La section suivante décrit les étapes préliminaires requises pour utiliser les commandes et les scripts de cette section.

Accès

- Compte AWS access in Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#)
- Compte AWS accès Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : Amazon MWAAFull ApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Ouvrez l'interface utilisateur d'Apache Airflow

L'image suivante affiche le lien vers votre interface utilisateur Apache Airflow sur la console Amazon MWAA.

Connectez-vous à Apache Airflow

Vous avez besoin d'[Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#) autorisations pour que votre Compte AWS identifiant Gestion des identités et des accès AWS (IAM) puisse accéder à votre interface utilisateur Apache Airflow.

Pour accéder à votre interface utilisateur Apache Airflow

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Open Airflow UI.

Création d'un jeton d'accès au serveur Web Apache Airflow

Vous pouvez utiliser les commandes de cette page pour créer un jeton d'accès au serveur Web. Un jeton d'accès permet d'accéder à votre environnement Amazon MWAA. Par exemple, vous pouvez obtenir un jeton, puis le déployer DAGs par programmation à l'aide d'Amazon MWAA. APIs La section suivante décrit les étapes à suivre pour créer un jeton de connexion Web Apache Airflow à l' AWS CLI aide d'un script bash, d'une requête d'API POST ou d'un script Python. Le jeton renvoyé dans la réponse est valide pendant 60 secondes.

Important

À compter du 19 août 2025, Amazon MWAA a ajouté la prise en charge des points de IPv6 terminaison, et désormais des supports IPv4 et IPv6 des points de terminaison. À compter de cette date, tous les environnements nouvellement créés utiliseront des `.on.aws` domaines pour l'interface utilisateur (UI) Airflow. Les clients doivent migrer leur interface utilisateur Airflow depuis `.on.aws` des domaines `.amazonaws.com` pour ces environnements nouvellement créés. Les services de point de terminaison Virtual Private Cloud (VPC) pour les serveurs Web et les bases de données conserveront leurs `.amazonaws.com` domaines actuels sans aucune modification requise.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [À l'aide du AWS CLI](#)
- [Utiliser un script bash](#)
- [Utilisation d'un script Python](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

La section suivante décrit les étapes préliminaires requises pour utiliser les commandes et les scripts de cette page.

Accès

- Compte AWS access in Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#)
- Compte AWS accès Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : Amazon MWAAFull ApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

À l'aide du AWS CLI

L'exemple suivant utilise la [create-web-login-token](#) commande du AWS CLI pour créer un jeton de connexion Web Apache Airflow.

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

Utiliser un script bash

L'exemple suivant utilise un script bash pour appeler la [create-web-login-token](#) commande dans le AWS CLI afin de créer un jeton de connexion Web Apache Airflow.

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `get-web-token.sh`.

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

2. Remplacez les espaces réservés par *red* `YOUR_HOST_NAME` et `YOUR_ENVIRONMENT_NAME`. Par exemple, le nom d'hôte d'un réseau public ressemble (sans le `https ://`) à :

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

3. (facultatif) Les utilisateurs de macOS et Linux devront peut-être exécuter la commande suivante pour s'assurer que le script est exécutable.


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

4. Exécutez le script suivant pour obtenir un jeton de connexion Web.

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

Votre invite de commande affiche :

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

Utilisation d'un script Python

L'exemple suivant utilise la méthode [boto3 create_web_login_token dans un script Python pour créer un jeton](#) de connexion Web Apache Airflow. Vous pouvez exécuter ce script en dehors d'Amazon MWAA. La seule chose que vous devez faire est d'installer la bibliothèque boto3. Vous souhaitez peut-être créer un environnement virtuel pour installer la bibliothèque. Cela suppose que vous avez [configuré les informations d'AWS authentication](#) pour votre compte.

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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. Remplacez l'espace *red* réservé `YOUR_ENVIRONMENT_NAME` par.
3. Exécutez le script suivant pour obtenir un jeton de connexion Web.

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

Quelle est la prochaine étape ?

- Découvrez l'opération d'API Amazon MWAA utilisée pour créer un jeton de connexion Web à l'[CreateWebLoginToken](#)adresse.

Configuration d'un domaine personnalisé pour le serveur Web Apache Airflow

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) vous permet de configurer un domaine personnalisé pour le serveur Web géré Apache Airflow. À l'aide d'un domaine personnalisé, vous pouvez accéder au serveur Web Apache Airflow géré par Amazon MWAA de votre environnement à l'aide de l'interface utilisateur Apache Airflow, de la CLI Apache Airflow ou du serveur Web Apache Airflow.

Note

Vous ne pouvez utiliser un domaine personnalisé qu'avec un serveur Web privé sans accès à Internet.

Cas d'utilisation d'un domaine personnalisé sur Amazon MWAA

1. Partagez le domaine du serveur Web dans votre application cloud sur AWS : l'utilisation d'un domaine personnalisé vous permet de définir une URL conviviale pour accéder au serveur Web, au lieu du nom de domaine de service généré. Vous pouvez stocker ce domaine personnalisé et le partager en tant que variable d'environnement dans vos applications.
2. Accès à un serveur Web privé : si vous souhaitez configurer l'accès à un serveur Web dans un VPC sans accès à Internet, l'utilisation d'un domaine personnalisé simplifie le flux de redirection d'URL.

Rubriques

- [Configuration du domaine personnalisé](#)
- [Configuration de l'infrastructure réseau](#)

Configuration du domaine personnalisé

Pour configurer la fonctionnalité de domaine personnalisé, vous devez fournir la valeur de domaine personnalisée via la configuration `webserver.base_url` Apache Airflow lors de la création ou de la mise à jour de votre environnement Amazon MWAA. Les contraintes suivantes s'appliquent à votre nom de domaine personnalisé :

- La valeur doit être un nom de domaine complet (FQDN) sans protocole ni chemin. Par exemple, `your-custom-domain.com`.
- Amazon MWAA n'autorise pas de chemin dans l'URL. Par exemple, `your-custom-domain.com/dags/` il ne s'agit pas d'un nom de domaine personnalisé valide.
- La longueur de l'URL est limitée à 255 caractères ASCII.
- Si vous fournissez une chaîne vide, par défaut, l'environnement sera créé avec une URL de serveur Web générée par Amazon MWAA.

Utilisez l'exemple suivant pour créer un environnement avec un nom de domaine de serveur Web personnalisé à l'aide du AWS CLI.

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

Une fois l'environnement créé ou mis à jour, vous devez configurer l'infrastructure réseau de votre ordinateur Compte AWS pour accéder au serveur Web privé via le domaine personnalisé.

Pour revenir à l'URL générée par le service par défaut, mettez à jour votre environnement privé et supprimez l'`webserver.base_url` option de configuration.

Configuration de l'infrastructure réseau

Suivez les étapes ci-dessous pour configurer l'infrastructure réseau requise à utiliser avec votre domaine personnalisé dans votre Compte AWS.

1. Obtenez les adresses IP des interfaces réseau (ENI) Amazon VPC Endpoint. Pour ce faire, utilisez d'abord le [get-environment](#) pour trouver celui qui `WebserverVpcEndpointService` convient à votre environnement.

```
aws mwa get-environment --name your-environment-name
```

En cas de succès, vous obtiendrez un résultat similaire à ce qui suit.

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

Notez la `WebserverVpcEndpointService` valeur et utilisez-la `web-server-vpc-endpoint-service` dans la EC2 `describe-vpc-endpoints` commande Amazon suivante. `--filters Name=service-name,Values=web-server-vpc-endpoint-service-id` dans la commande suivante.

2. Récupérez les détails du point de terminaison Amazon VPC. Cette commande récupère des informations sur les points de terminaison Amazon VPC qui correspondent à un nom de service spécifique, en renvoyant l'ID du point de terminaison et l'IDs interface réseau associée au format texte.

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

3. Obtenez les détails de l'interface réseau. Cette commande récupère les adresses IP privées pour chaque interface réseau associée aux points de terminaison Amazon VPC identifiés à l'étape précédente.

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

4. `create-target-group` À utiliser pour créer un nouveau groupe cible. Vous utiliserez ce groupe cible pour enregistrer les adresses IP des points de terminaison Amazon VPC de votre serveur Web.

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

Enregistrez les adresses IP à l'aide de la `register-targets` commande.

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

5. Demandez un certificat ACM. Ignorez cette étape si vous utilisez un certificat existant.

```
aws acm request-certificate \
```

```
--domain-name my-custom-domain.com \
--validation-method DNS
```

6. Configurez un Application Load Balancer. Créez d'abord l'équilibreur de charge, puis créez un écouteur pour l'équilibreur de charge. Spécifiez le certificat ACM que vous avez créé à l'étape précédente.

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

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

Si vous utilisez un Network Load Balancer dans un sous-réseau privé, configurez un [hôte bastion](#) ou un [Site-to-Site VPN tunnel](#) pour accéder au serveur Web.

7. Créez une zone hébergée en utilisant Route 53 pour le domaine.

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

Créez un enregistrement A pour le domaine. Pour ce faire, utilisez le AWS CLI, obtenez l'ID de zone hébergée en utilisant `list-hosted-zones-by-name` puis appliquez l'enregistrement avec `change-resource-record-sets`.

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

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

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

8. Mettez à jour les règles du groupe de sécurité pour le point de terminaison Amazon VPC du serveur Web afin de respecter le principe du moindre privilège en autorisant le trafic HTTPS uniquement depuis les sous-réseaux publics où se trouve l'Application Load Balancer. Enregistrez le code JSON suivant localement. Par exemple, comme `sg-ingress-ip-permissions.json`.

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

Exécutez la EC2 commande Amazon suivante pour mettre à jour les règles de votre groupe de sécurité d'entrée. Spécifiez le fichier JSON pour `--ip-permissions`.

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

Exécutez la EC2 commande Amazon suivante pour mettre à jour vos règles de sortie.

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

Ouvrez la console Amazon MWAA et accédez à l'interface utilisateur d'Apache Airflow. Si vous configurez un Network Load Balancer dans un sous-réseau privé au lieu de l'Application Load Balancer utilisé ici, vous devez accéder au serveur Web avec l'une des options suivantes.

- [the section called “Tutoriel : Linux Bastion Host”](#)
- [the section called “Tutoriel : AWS Client VPN”](#)

Création d'un jeton CLI Apache Airflow

Tip

L'API REST est plus moderne que la CLI et est conçue pour une intégration programmatique avec des systèmes externes. REST est la méthode préférée pour interagir avec Apache Airflow.

Vous pouvez utiliser les commandes de cette page pour générer un jeton CLI, puis effectuer des appels d'API Amazon Managed Workflows for Apache Airflow directement dans votre interface de commande. Par exemple, vous pouvez obtenir un jeton, puis le déployer DAGs par programmation à l'aide d'Amazon MWAA. APIs La section suivante décrit les étapes de création d'un jeton CLI Apache

Airflow à l'aide d'un script curl AWS CLI, d'un script Python ou d'un script bash. Le jeton renvoyé dans la réponse est valide pendant 60 secondes.

Le AWS CLI jeton est destiné à remplacer les actions du shell synchrone, et non les commandes d'API asynchrones. Par conséquent, la simultanéité disponible est limitée. Pour garantir que le serveur Web reste réactif pour les utilisateurs, nous recommandons de ne pas ouvrir de nouvelle AWS CLI demande tant que la précédente n'est pas terminée avec succès.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [En utilisant le AWS CLI](#)
- [Utilisation d'un script curl](#)
- [Utiliser un script bash](#)
- [Utilisation d'un script Python](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

La section suivante décrit les étapes préliminaires requises pour utiliser les commandes et les scripts de cette page.

Accès

- Compte AWS access in Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#)
- Compte AWS accès Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : Amazon MWAAFull ApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

En utilisant le AWS CLI

L'exemple suivant utilise la [create-cli-token](#) commande du AWS CLI pour créer un jeton d'interface de ligne de commande Apache Airflow.

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

Utilisation d'un script curl

L'exemple suivant utilise un script curl pour appeler la [create-web-login-token](#) commande dans le afin d'appeler la AWS CLI CLI Apache Airflow via un point de terminaison sur le serveur Web Apache Airflow.

Apache Airflow v3

1. Copiez l'instruction curl depuis votre fichier texte et collez-la dans votre interface de commande.

Note

Après l'avoir copié dans votre presse-papiers, vous devrez peut-être utiliser Edition > Coller dans le menu du shell.

```
CLI_JSON=$(aws mwa --region us-east-1 create-cli-token --  
name YOUR_ENVIRONMENT_NAME) \  
&& CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \  
&& WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \  
&& CLI_RESULTS=$(curl -L --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwa/  
cli" \  
--header "Authorization: Bearer $CLI_TOKEN" \  
--header "Content-Type: text/plain" \  
--data-raw "dags trigger YOUR_DAG_NAME --logical-date $(date -u +"%Y-%m-%dT%H:  
%M:%SZ")") \  
&& echo "Output:" \  
&& echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \  

```

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

2. Remplacez les espaces réservés *red* par ceux Région AWS de votre environnement *YOUR_DAG_NAME*, et *YOUR_ENVIRONMENT_NAME*. Par exemple, le nom d'hôte d'un réseau public ressemble (sans le https ://) à :

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

Votre invite de commande affiche :

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

Apache Airflow v2

1. Copiez l'instruction curl depuis votre fichier texte et collez-la dans votre interface de commande.

Note

Après l'avoir copié dans votre presse-papiers, vous devrez peut-être utiliser Edition > Coller dans le menu du shell.

```
CLI_JSON=$(aws mwa --region us-east-1 create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
&& CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
&& WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
&& CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_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. Remplacez les espaces réservés *red* par ceux Région AWS de votre environnement `YOUR_DAG_NAME`, et `YOUR_ENVIRONMENT_NAME`. Par exemple, le nom d'hôte d'un réseau public ressemble (sans le `https ://`) à :

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

Votre invite de commande affiche :

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

Utiliser un script bash

L'exemple suivant utilise un script bash pour appeler la [create-cli-token](#) commande dans le afin de créer un jeton AWS CLI d'interface de ligne de commande Apache Airflow.

Apache Airflow v3

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `get-cli-token.sh`.

```
# brew install jq
aws mwa create-cli-token --name YOUR_ENVIRONMENT_NAME | export
CLI_TOKEN=$(jq -r .CliToken) && curl -L --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 --logical-date $(date -u +"%Y-%m-%dT%H:%M:%SZ")"
```

2. Remplacez les espaces réservés *red* par `YOUR_ENVIRONMENT_NAME` `YOUR_HOST_NAME`, et `YOUR_DAG_NAME`. Par exemple, le nom d'hôte d'un réseau public ressemble (sans le `https ://`) à :

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

3. (facultatif) Les utilisateurs de macOS et Linux devront peut-être exécuter la commande suivante pour s'assurer que le script est exécutable.

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

4. Exécutez le script suivant pour créer un jeton d'interface de ligne de commande Apache Airflow.

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

Apache Airflow v2

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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. Remplacez les espaces réservés *red* par `YOUR_ENVIRONMENT_NAME`, `YOUR_HOST_NAME`, et `YOUR_DAG_NAME`. Par exemple, le nom d'hôte d'un réseau public ressemble (sans le `https ://`) à :

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

3. (facultatif) Les utilisateurs de macOS et Linux peuvent exécuter la commande suivante pour s'assurer que le script est exécutable.

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

4. Exécutez le script suivant pour créer un jeton d'interface de ligne de commande Apache Airflow.

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

Utilisation d'un script Python

L'exemple suivant utilise la méthode [boto3 create_cli_token](#) dans un script Python pour créer un jeton CLI Apache Airflow et déclencher un DAG. Vous pouvez exécuter ce script en dehors d'Amazon MWAA. La seule chose que vous devez faire est d'installer la bibliothèque boto3. Vous souhaitez peut-être créer un environnement virtuel pour installer la bibliothèque. Cela suppose que vous avez [configuré les informations d'AWS authentication](#) pour votre compte.

Apache Airflow v3

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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
```

```

)

mwa_auth_token = 'Bearer ' + mwa_cli_token['CliToken']
mwa_webserver_hostname = 'https://{0}/aws_mwa/
cli'.format(mwa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwa_cli_command, dag_name)

mwa_response = requests.post(
    mwa_webserver_hostname,
    headers={
        'Authorization': mwa_auth_token,
        'Content-Type': 'text/plain'
    },
    data=raw_data
)

mwa_std_err_message = base64.b64decode(mwa_response.json()
['stderr']).decode('utf8')
mwa_std_out_message = base64.b64decode(mwa_response.json()
['stdout']).decode('utf8')

print(mwa_response.status_code)
print(mwa_std_err_message)
print(mwa_std_out_message)

```

2. Remplacez les espaces réservés par YOUR_ENVIRONMENT_NAME et. YOUR_DAG_NAME
3. Exécutez le script suivant pour créer un jeton d'interface de ligne de commande Apache Airflow.

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

Apache Airflow v2

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous 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. Remplacez les espaces réservés par YOUR_ENVIRONMENT_NAME et. YOUR_DAG_NAME
3. Exécutez le script suivant pour créer un jeton d'interface de ligne de commande Apache Airflow.

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

Quelle est la prochaine étape ?

- Découvrez l'opération d'API Amazon MWAA utilisée pour créer un jeton CLI sur [CreateCliToken](#).

Utilisation de l'API REST Apache Airflow

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) permet d'interagir avec vos environnements Apache Airflow directement à l'aide de l'API REST Apache Airflow pour les environnements exécutant Apache Airflow v2.4.3 et versions ultérieures. Cela vous permet d'accéder à vos environnements Amazon MWAA et de les gérer par programmation, fournissant ainsi un moyen standardisé d'invoquer des flux de travail d'orchestration de données, de gérer vos différents composants d'Apache Airflow et de surveiller leur état DAGs, tels que la base de données de métadonnées, le déclencheur et le planificateur.

Pour garantir l'évolutivité lors de l'utilisation de l'API REST d'Apache Airflow, Amazon MWAA vous offre la possibilité de dimensionner horizontalement la capacité du serveur Web pour faire face à une demande accrue, qu'il s'agisse de demandes d'API REST, de l'utilisation de l'interface de ligne de commande (CLI) ou d'un plus grand nombre d'utilisateurs simultanés de l'interface utilisateur (UI) Apache Airflow. Pour plus d'informations sur la manière dont Amazon MWAA fait évoluer les serveurs Web, consultez [the section called “Configuration de la mise à l'échelle automatique du serveur Web”](#)

Vous pouvez utiliser l'API REST Apache Airflow pour implémenter les cas d'utilisation suivants pour vos environnements :

- Accès par programmation : vous pouvez désormais démarrer les exécutions d'Apache Airflow DAG, gérer des ensembles de données et récupérer l'état de divers composants tels que la

base de données de métadonnées, les déclencheurs et les planificateurs sans avoir à vous fier à l'interface utilisateur ou à la CLI d'Apache Airflow.

- **Intégration à des applications externes et à des microservices** : vous pouvez utiliser le support des API REST pour créer des solutions personnalisées qui intègrent vos environnements Amazon MWAA à d'autres systèmes. Par exemple, vous pouvez démarrer des flux de travail en réponse à des événements provenant de systèmes externes, tels que des tâches de base de données terminées ou l'inscription de nouveaux utilisateurs.
- **Surveillance centralisée** : vous pouvez créer des tableaux de bord de surveillance qui regroupent votre statut DAGs sur plusieurs environnements Amazon MWAA, permettant ainsi une surveillance et une gestion centralisées.

Pour plus d'informations sur l'API REST d'Apache Airflow, reportez-vous au manuel de référence de l'API REST [d'Apache Airflow](#).

En utilisant `InvokeRestApi`, vous pouvez accéder à l'API REST Apache Airflow à l'aide AWS d'informations d'identification. Vous pouvez également y accéder en obtenant un jeton d'accès au serveur Web, puis en utilisant le jeton pour l'appeler.

Si vous rencontrez une erreur avec le message `Update your environment to use InvokeRestApi` lors de l'utilisation de l'`InvokeRestApi` opération, cela indique que vous devez mettre à jour votre environnement Amazon MWAA. Cette erreur se produit lorsque votre environnement Amazon MWAA n'est pas compatible avec les dernières modifications liées à `InvokeRestApi` cette fonctionnalité. Pour résoudre ce problème, mettez à jour votre environnement Amazon MWAA afin d'intégrer les modifications nécessaires à la `InvokeRestApi` fonctionnalité.

Le délai d'expiration par défaut de l'`InvokeRestApi` opération est de 10 secondes. Si l'opération ne se termine pas dans ce délai de 10 secondes, elle est automatiquement interrompue et une erreur est générée. Assurez-vous que vos appels d'API REST sont conçus pour être exécutés dans ce délai afin d'éviter de rencontrer des erreurs.

Pour garantir l'évolutivité lors de l'utilisation de l'API REST d'Apache Airflow, Amazon MWAA vous offre la possibilité de dimensionner horizontalement la capacité du serveur Web pour faire face à une demande accrue, qu'il s'agisse de demandes d'API REST, de l'utilisation de l'interface de ligne de commande (CLI) ou d'un plus grand nombre d'utilisateurs simultanés de l'interface utilisateur (UI) Apache Airflow. Pour plus d'informations sur la manière dont Amazon MWAA fait évoluer les serveurs Web, consultez [the section called "Configuration de la mise à l'échelle automatique du serveur Web"](#)

Vous pouvez utiliser l'API REST Apache Airflow pour implémenter les cas d'utilisation suivants pour vos environnements :

- Accès par programmation : vous pouvez désormais démarrer les exécutions d'Apache Airflow DAG, gérer des ensembles de données et récupérer l'état de divers composants tels que la base de données de métadonnées, les déclencheurs et les planificateurs sans avoir à vous fier à l'interface utilisateur ou à la CLI d'Apache Airflow.
- Intégration à des applications externes et à des microservices : vous pouvez utiliser le support des API REST pour créer des solutions personnalisées qui intègrent vos environnements Amazon MWAA à d'autres systèmes. Par exemple, vous pouvez démarrer des flux de travail en réponse à des événements provenant de systèmes externes, tels que des tâches de base de données terminées ou l'inscription de nouveaux utilisateurs.
- Surveillance centralisée : vous pouvez créer des tableaux de bord de surveillance qui regroupent votre statut DAGs sur plusieurs environnements Amazon MWAA, permettant ainsi une surveillance et une gestion centralisées.

Pour plus d'informations sur l'API REST d'Apache Airflow, reportez-vous au manuel de référence de [l'API REST d'Apache Airflow](#).

En utilisant `InvokeRestApi`, vous pouvez accéder à l'API REST Apache Airflow à l'aide AWS d'informations d'identification. Vous pouvez également y accéder en obtenant un jeton d'accès au serveur Web, puis en utilisant le jeton pour l'appeler.

- Si vous rencontrez une erreur avec le message `Update your environment to use InvokeRestApi` lors de l'utilisation de l'`InvokeRestApi` opération, cela indique que vous devez mettre à jour votre environnement Amazon MWAA. Cette erreur se produit lorsque votre environnement Amazon MWAA n'est pas compatible avec les dernières modifications liées à `InvokeRestApi` cette fonctionnalité. Pour résoudre ce problème, mettez à jour votre environnement Amazon MWAA afin d'intégrer les modifications nécessaires à la `InvokeRestApi` fonctionnalité.
- Le délai d'expiration par défaut de l'`InvokeRestApi` opération est de 10 secondes. Si l'opération ne se termine pas dans ce délai de 10 secondes, elle est automatiquement interrompue et une erreur est générée. Assurez-vous que vos appels d'API REST sont conçus pour être exécutés dans ce délai afin d'éviter de rencontrer des erreurs.

⚠ Important

La taille de la charge utile de réponse ne peut pas dépasser 6 Mo. Vous RestApi échouez si cette limite est dépassée.

Utilisez les exemples suivants pour effectuer des appels d'API vers l'API REST Apache Airflow et démarrer une nouvelle exécution du DAG :

Rubriques

- [Octroi de l'accès à l'API REST Apache Airflow : airflow:InvokeRestApi](#)
- [Appel de l'API REST Apache Airflow](#)
- [Création d'un jeton de session de serveur Web et appel de l'API REST Apache Airflow](#)

Octroi de l'accès à l'API REST Apache Airflow :**airflow:InvokeRestApi**

Pour accéder à l'API REST Apache Airflow à l'aide AWS d'informations d'identification, vous devez accorder l'`airflow:InvokeRestApi` autorisation dans votre politique IAM. Dans l'exemple de politique suivant, spécifiez le `AdminOp`, `User`, `Viewer`, ou `Public` le rôle dans `{airflow-role}` pour personnaliser le niveau d'accès des utilisateurs. Pour plus d'informations, reportez-vous à la section [Rôles par défaut](#) du guide de référence d'Apache Airflow.

JSON

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

```
]
}
```

Note

Lors de la configuration d'un serveur Web privé, l'`InvokeRestApi` action ne peut pas être invoquée depuis l'extérieur d'un Virtual Private Cloud (VPC). Vous pouvez utiliser la `aws:SourceVpc` clé pour appliquer un contrôle d'accès plus précis à cette opération. Pour plus d'informations, consultez [aws : SourceVpc](#).

Appel de l'API REST Apache Airflow

L'exemple de script suivant explique comment utiliser l'API REST d'Apache Airflow pour répertorier les éléments disponibles DAGs dans votre environnement et comment créer une variable Apache Airflow :

```
import boto3

env_name = "MyAirflowEnvironment"

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

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

def create_variable(client):
    request_params = {
        "Name": env_name,
        "Path": "/variables",
```

```
"Method": "POST",
"Body": {
    "key": "test-restapi-key",
    "value": "test-restapi-value",
    "description": "Test variable created by MWA InvokeRestApi API",
}
}
response = client.invoke_rest_api(
    **request_params
)

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

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

Création d'un jeton de session de serveur Web et appel de l'API REST Apache Airflow

Pour créer un jeton d'accès au serveur Web, utilisez la fonction Python suivante. Cette fonction appelle d'abord l'API Amazon MWA pour obtenir un jeton de connexion Web. Le jeton de connexion Web, qui expire au bout de 60 secondes, est ensuite échangé contre un jeton de session Web, qui vous permet d'accéder au serveur Web et d'utiliser l'API REST Apache Airflow. Si vous avez besoin d'une capacité de régulation supérieure à 10 transactions par seconde (TPS), vous pouvez utiliser cette méthode pour accéder à l'API REST d'Apache Airflow.

Le jeton de session expire au bout de 12 heures.

Tip

Les principales modifications apportées aux exemples de code suivants entre Apache Airflow v2 et v3 sont les suivantes :

- Le chemin de l'API REST est passé de `/api/v1` à `/api/v2`
- Le chemin de connexion est passé de `/aws_maa/login` à `/pluginsv2/aws_mwa/login`
- La réponse de connexion `response.cookies["_token"]` contient des informations de jeton que vous devez utiliser pour les appels d'API suivants

- Pour un appel d'API REST, vous devez transmettre les `jwt_token` informations dans les en-têtes sous la forme suivante :

```
headers = {  
    "Authorization": f"Bearer {jwt_token}",  
    "Content-Type": "application/json"  
}
```

Apache Airflow v3

```
def get_token_info(region, env_name):  
    logging.basicConfig(level=logging.INFO)  
  
    try:  
        # Initialize MWS client and request a web login token  
        mws = boto3.client('mws', region_name=region)  
        response = mws.create_web_login_token(Name=env_name)  
  
        # Extract the web server hostname and login token  
        web_server_host_name = response["WebServerHostname"]  
        web_token = response["WebToken"]  
  
        # Construct the URL needed for authentication  
        login_url = f"https://{web_server_host_name}/pluginsv2/aws_mws/login"  
        login_payload = {"token": web_token}  
  
        # Make a POST request to the MWS login url using the login payload  
        response = requests.post(  
            login_url,  
            data=login_payload,  
            timeout=10  
        )  
  
        # Check if login was successful  
        if response.status_code == 200:  
  
            # Return the hostname and the session cookie  
            return (  
                web_server_host_name,  
                response.cookies['_token']  
            )
```

```
else:
    # Log an error
    logging.error("Failed to log in: HTTP %d", response.status_code)
    return None
except requests.RequestException as e:

    # Log any exceptions raised during the request to the MWA login endpoint
    logging.error("Request failed: %s", str(e))
    return None
except Exception as e:

    # Log any other unexpected exceptions
    logging.error("An unexpected error occurred: %s", str(e))
    return None
```

Apache Airflow v2

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

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

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

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

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

        # Check if login was successful
        if response.status_code == 200:
```



```

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

```

Une fois l'authentification terminée, vous disposez des informations d'identification nécessaires pour commencer à envoyer des demandes aux points de terminaison de l'API. Dans l'exemple de la section suivante, utilisez le point de terminaison `dags/{dag_name}/dagRuns` .

Apache Airflow v3

```

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

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

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

    # Retrieve the web server hostname and token for authentication
    try:
        web_server_host_name, jwt_token = get_token_info(region, env_name)
        if not jwt_token:
            logging.error("Authentication failed, no jwt token retrieved.")

```

```
    return
except Exception as e:
    logging.error(f"Error retrieving token info: {str(e)}")
    return

# Prepare headers and payload for the request
request_headers = {
    "Authorization": f"Bearer {jwt_token}",
    "Content-Type": "application/json" # Good practice to include, even for GET
}

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

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

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

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

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

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

    # Trigger the DAG with the provided arguments
```

```
trigger_dag(region, env_name, dag_id)
```

Apache Airflow v2

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

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

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

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

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

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

    # Send the POST request to trigger the DAG
    try:
        response = requests.post(url, cookies=cookies, json=json_body)
    # Check the response status code to determine if the DAG was triggered
    successfully
    if response.status_code == 200:
        logging.info("DAG triggered successfully.")
    else:
```

```
logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
{response.text}")
except requests.RequestException as e:
    logging.error(f"Request to trigger DAG failed: {str(e)}")

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

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

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

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

Référence des commandes de la CLI Apache Airflow

Cette rubrique décrit les commandes de la CLI Apache Airflow prises en charge et non prises en charge sur Amazon Managed Workflows pour Apache Airflow.

Tip

L'API REST est plus moderne que la CLI et est conçue pour une intégration programmatique avec des systèmes externes. REST est la méthode préférée pour interagir avec Apache Airflow.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [Qu'est-ce qui a changé ?](#)

- [Commandes CLI prises en charge](#)
 - [Commandes prises en charge](#)
 - [Utilisation de commandes qui analysent DAGs](#)
- [Exemple de code](#)
 - [Définir, obtenir ou supprimer une variable Apache Airflow v2](#)
 - [Ajouter une configuration lors du déclenchement d'un DAG](#)
 - [Exécuter des commandes CLI sur un tunnel SSH vers un hôte bastion](#)

Prérequis

La section suivante décrit les étapes préliminaires requises pour utiliser les commandes et les scripts de cette page.

Accès

- Compte AWS access in Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#)
- Compte AWS accès Gestion des identités et des accès AWS (IAM) à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : Amazon MWAAFull ApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Qu'est-ce qui a changé ?

- v3 : architecture Airflow. Apache Airflow v3 introduit des modifications architecturales majeures afin d'améliorer la sécurité et l'évolutivité, ainsi que de faciliter la maintenance. Pour en savoir plus, reportez-vous à la section [Mise à niveau vers Airflow 3](#).

- v2 : structure de commande Airflow CLI. La CLI Apache Airflow v2 est organisée de telle sorte que les commandes associées sont regroupées sous forme de sous-commandes, ce qui signifie que vous devez mettre à jour les scripts Apache Airflow v1 si vous souhaitez passer à Apache Airflow v2. Par exemple, `unpause` dans Apache Airflow v1, c'est `dag unpause` dans Apache Airflow v2. Pour en savoir plus, reportez-vous aux [modifications apportées à la CLI Airflow dans la version 2.0](#).

Commandes CLI prises en charge

La section suivante répertorie les commandes de la CLI Apache Airflow disponibles sur Amazon MWAA.

Commandes prises en charge

Apache Airflow v3

Versions mineures	Commande
v3.0.6	détails des actifs
v3.0.6	liste des actifs
v3.0.6	les actifs se matérialisent
v3.0.6	créer un remblai
v3.0.6	aide-mémoire
v3.0.6	connexions ajouter
v3.0.6	suppression des connexions
v3.0.6	jours supprimer

Versions mineures	Commande
v3.0.6	liste des jours
v3.0.6	liste de jours-jobs
v3.0.6	jours list-import-errors
v3.0.6	jours list-runs
v3.0.6	jours de prochaine exécution
v3.0.6	pause de jours
v3.0.6	rapport de jours
v3.0.6	jours resérialisés
v3.0.6	affichage des jours
v3.0.6	état des jours
v3.0.6	test de jours
v3.0.6	déclencheur Dags
v3.0.6	jours d'inpause
v3.0.6	base de données propre
v3.0.6	comportements des fournisseurs

Versions mineures	Commande
v3.0.6	<u>les fournisseurs obtiennent</u>
v3.0.6	<u>crochets pour fournisseurs</u>
v3.0.6	<u>liens vers les fournisseurs</u>
v3.0.6	<u>liste des fournisseurs</u>
v3.0.6	<u>notifications des fournisseurs</u>
v3.0.6	<u>secrets des fournisseurs</u>
v3.0.6	<u>déclencheur des fournisseurs</u>
v3.0.6	<u>widgets des fournisseurs</u>
v3.0.6	<u>rôles add perms</u>
v3.0.6	<u>rôles del-perms</u>
v3.0.6	<u>les rôles créent</u>
v3.0.6	<u>liste des rôles</u>
v3.0.6	<u>tâches claires</u>
v3.0.6	<u>tâches échouées-deps</u>

Versions mineures	Commande
v3.0.6	liste des tâches
v3.0.6	rendu des tâches
v3.0.6	état des tâches
v3.0.6	tâches states-for-dag-run
v3.0.6	test de tâches
v3.0.6	suppression de variables
v3.0.6	les variables obtiennent
v3.0.6	ensemble de variables
v3.0.6	liste de variables
v3.0.6	Version

Apache Airflow v2

Versions mineures	Commande
version 2.0 et versions ultérieures	aide-mémoire
version 2.0 et versions ultérieures	connexions ajouter
version 2.0 et versions ultérieures	suppression des connexions

Versions mineures	Commande
v2.2+ (remarque)	remblai de Dags
version 2.0 et versions ultérieures	jours supprimer
v2.2+ (remarque)	liste des jours
version 2.0 et versions ultérieures	liste de jours-jobs
v2.6 et versions ultérieures	jours list-import-errors
v2.2+ (remarque)	jours list-runs
v2.2+ (remarque)	jours de prochaine exécution
version 2.0 et versions ultérieures	pause de jours
version 2.0 et versions ultérieures	rapport de jours
v2.4 et versions ultérieures	jours resérialisés
version 2.0 et versions ultérieures	affichage des jours
version 2.0 et versions ultérieures	état des jours
version 2.0 et versions ultérieures	test de jours
version 2.0 et versions ultérieures	déclencheur Dags

Versions mineures	Commande
version 2.0 et versions ultérieures	jours d'inpase
v2.4 et versions ultérieures	base de données propre
version 2.0 et versions ultérieures	comportements des fournisseurs
version 2.0 et versions ultérieures	les fournisseurs obtiennent
version 2.0 et versions ultérieures	crochets pour fournisseurs
version 2.0 et versions ultérieures	liens vers les fournisseurs
version 2.0 et versions ultérieures	liste des fournisseurs
v2.8 et versions ultérieures	notifications des fournisseurs
v2.6 et versions ultérieures	secrets des fournisseurs
v2.7+	déclencheur des fournisseurs
version 2.0 et versions ultérieures	widgets des fournisseurs
v2.6 et versions ultérieures	rôles add perms
v2.6 et versions ultérieures	rôles del-perms
v2.6 et versions ultérieures	les rôles créent

Versions mineures	Commande
version 2.0 et versions ultérieures	liste des rôles
version 2.0 et versions ultérieures	tâches claires
version 2.0 et versions ultérieures	tâches échouées-deps
version 2.0 et versions ultérieures	liste des tâches
version 2.0 et versions ultérieures	rendu des tâches
version 2.0 et versions ultérieures	tâches exécutées
version 2.0 et versions ultérieures	état des tâches
version 2.0 et versions ultérieures	tâches states-for-dag-run
version 2.0 et versions ultérieures	test de tâches
version 2.0 et versions ultérieures	suppression de variables
version 2.0 et versions ultérieures	les variables obtiennent
version 2.0 et versions ultérieures	ensemble de variables
version 2.0 et versions ultérieures	liste de variables
version 2.0 et versions ultérieures	Version

Utilisation de commandes qui analysent DAGs

Si votre environnement exécute Apache Airflow v2.0.2, les commandes CLI qui DAGs analysent échoueront si le DAG utilise des plugins qui dépendent de packages installés via un : `requirements.txt`

Apache Airflow v2.0.2

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

Vous pouvez utiliser ces commandes CLI si vous DAGs n'utilisez pas de plugins qui dépendent de packages installés via `unrequirements.txt`.

Exemple de code

La section suivante contient des exemples de différentes manières d'utiliser la CLI Apache Airflow.

Définir, obtenir ou supprimer une variable Apache Airflow v2

Vous pouvez utiliser l'exemple de code suivant pour définir, obtenir ou supprimer une variable au format `<script> <mwa env name> get | set | delete <variable> <variable value> </variable> </variable>`.

```
[ $# -eq 0 ] && echo "Usage: $0 MWA 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 mwa --region $AWS_REGION create-cli-token --name $1) \
```

```

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

```

Ajouter une configuration lors du déclenchement d'un DAG

Vous pouvez utiliser l'exemple de code suivant avec Apache Airflow v2 pour ajouter une configuration lors du déclenchement d'un DAG, telle que `airflow trigger_dag 'dag_name' -conf '{"key":"value"}'`

```

import boto3
import json
import requests
import base64

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

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

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

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

```

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

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

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

Exécuter des commandes CLI sur un tunnel SSH vers un hôte bastion

Utilisez l'exemple suivant pour exécuter des commandes de la CLI Airflow à l'aide d'un proxy de tunnel SSH vers un hôte Linux Bastion.

Utilisation de curl

1.

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

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

Gestion des connexions à Apache Airflow

Ce chapitre explique comment configurer une connexion Apache Airflow pour un environnement Amazon Managed Workflows for Apache Airflow.

Rubriques

- [Vue d'ensemble des variables et des connexions Apache Airflow](#)
- [Packages du fournisseur Apache Airflow installés sur les environnements Amazon MWAA](#)
- [Vue d'ensemble des types de connexion](#)
- [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)

Vue d'ensemble des variables et des connexions Apache Airflow

Dans certains cas, vous souhaitez peut-être spécifier des connexions ou des variables supplémentaires pour un environnement, tel qu'un AWS profil, ou ajouter votre rôle d'exécution dans un objet de connexion du métastore Apache Airflow, puis faire référence à la connexion depuis un DAG.

- Apache Airflow autogéré. Sur une installation Apache Airflow autogérée, vous définissez les options de configuration d'[Apache Airflow](#) dans `airflow.cfg`

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

- Apache Airflow sur Amazon MWAA. Sur Amazon MWAA, vous devez ajouter ces paramètres de configuration en tant qu'options de configuration [Apache Airflow sur la console](#) Amazon MWAA. Les options de configuration d'Apache Airflow sont écrites sous forme de variables d'environnement dans votre environnement et remplacent toutes les autres configurations existantes pour le même paramètre.

Packages du fournisseur Apache Airflow installés sur les environnements Amazon MWAA

Cette page répertorie les packages du fournisseur Apache Airflow installés par Amazon MWAA pour tous les environnements Apache Airflow pris en charge. Pour plus d'informations sur ces packages, reportez-vous à la [référence Apache Airflow pour les](#) modules supplémentaires.

Note

Pour garantir que la compatibilité avec la CloudWatch journalisation n'est pas annulée par d'autres installations de bibliothèques Python, Amazon MWAA installe la version 2.0.1 de [Watchtower](#) après l'avoir exécutée. `pip3 install -r requirements.txt`

Rubriques

- [Fichier de contraintes](#)
- [Packages de fournisseurs spécifiques à la version](#)

Fichier de contraintes

À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. `--constraint` Si vous ne fournissez aucune contrainte, Amazon MWAA vous en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Les fichiers de contraintes d'Apache Airflow spécifient les versions des fournisseurs disponibles au moment de la publication d'Apache Airflow. Dans de nombreux cas, cependant, les nouveaux fournisseurs sont compatibles avec cette version d'Apache Airflow. Comme vous devez utiliser des contraintes, pour spécifier une version plus récente d'un package de fournisseur, vous pouvez modifier le fichier de contraintes pour une version de fournisseur spécifique :

1. Téléchargez le fichier de contraintes spécifique à la version depuis GitHub, par exemple <https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt> (remplacez « 2.7.2 » par la version que vous souhaitez utiliser).

2. Enregistrez le fichier de contraintes modifié dans le dossier Amazon S3 dags de votre environnement Amazon MWAA, par exemple sous le nom de. `constraints-3.11-updated.txt`
3. Spécifiez vos besoins comme indiqué ci-dessous.

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

Note

Si vous utilisez un serveur Web privé, nous vous recommandons d'[empaqueter les bibliothèques requises sous forme de fichiers WHL](#) en utilisant. [aws-mwaa-docker-images](#)

Packages de fournisseurs spécifiques à la version

Installation de packages de fournisseurs que vous pouvez utiliser pour accéder à un type de connexion dans l'interface utilisateur d'Apache Airflow. Cela signifie également que vous n'avez pas besoin de spécifier ces packages en tant que dépendance Python dans votre `requirements.txt` fichier. Cette page répertorie les packages du fournisseur Apache Airflow installés par Amazon MWAA pour tous les environnements Apache Airflow pris en charge.

Note

Pour Apache Airflow v2 et versions ultérieures, Amazon MWAA installe la [version 2.0.1 de Watchtower](#) après l'avoir exécuté `pip3 install -r requirements.txt`, afin de garantir que la compatibilité avec la CloudWatch journalisation ne soit pas annulée par d'autres installations de bibliothèques Python.

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur.

Versions d'Apache Airflow prises en charge :

v3.0.6

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==9.9.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==6.2.1</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3,13,1</u>
Fab Connection	<u>apache-airflow-providers-fab==2,3.0</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3,12.1</u>
Connexion HTTP	<u>apache-airflow-providers-http==5.3.2</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3,9.1</u>
SQL commun	<u>apache-airflow-providers-common-sql= =1,27,3</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==4.1.1</u>

v2.11.0

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==9.8.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==6,2,0</u>

Type de connexion	Package
Connexion FTP	<u>apache-airflow-providers-ftp==3,13,0</u>
Fab Connection	<u>apache-airflow-providers-fab==1,5,3</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3,11,0</u>
Connexion HTTP	<u>apache-airflow-providers-http==5,3,0</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3,9,0</u>
SQL commun	<u>apache-airflow-providers-common-sql= =1,27.1</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==4,10</u>
Connexion SMTP	<u>apache-airflow-providers-smtp==2,10</u>

v2.10.3

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==9.0.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,13,1</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3.11.1</u>
Fab Connection	<u>apache-airflow-providers-fab==1,5,0</u>

Type de connexion	Package
Connexion au céleri	<u>apache-airflow-providers-celery==3,8,3</u>
Connexion HTTP	<u>apache-airflow-providers-http==4,13,2</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3,7,0</u>
SQL commun	<u>apache-airflow-providers-common-sql= =1,19,0</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3,9,0</u>
Connexion SMTP	<u>apache-airflow-providers-smtp==1,8,0</u>

v2.10.1

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiob otocore] ==8,28,0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,12,0</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3,11,0</u>
Fab Connection	<u>apache-airflow-providers-fab==1,3,0</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3,8,1</u>
Connexion HTTP	<u>apache-airflow-providers-http==4,13,0</u>

Type de connexion	Package
Connexion IMAP	<u>apache-airflow-providers-imap==3,7.0</u>
SQL commun	<u>apache-airflow-providers-common-sql= =1,16,0</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3,9,0</u>
Connexion SMTP	<u>apache-airflow-providers-smtp==1,8.0</u>

v2.9.2

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==8,24.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,11</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3,9.1</u>
Fab Connection	<u>apache-airflow-providers-fab==1.1.1</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3.7.2</u>
Connexion HTTP	<u>apache-airflow-providers-http==4.11.1</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3,6.1</u>
SQL commun	

Type de connexion	Package
	<u>apache-airflow-providers-common-sql==1,14.0</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3,8.1</u>
Connexion SMTP	<u>apache-airflow-providers-smtp==1,7.1</u>

v2.8.1

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==8.16.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,1,0</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3,7.0</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3.5.1</u>
Connexion HTTP	<u>apache-airflow-providers-http==4,8,0</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3,5,0</u>
SQL commun	<u>apache-airflow-providers-common-sql==1,1,0</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3,7.0</u>

v2.7.2

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==8.7.1</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,6.1</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3.5.2</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3.3.4</u>
Connexion HTTP	<u>apache-airflow-providers-http==4,5.2</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3.3.2</u>
SQL commun	<u>apache-airflow-providers-common-sql==1,7.2</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3.4.3</u>

v2.6.3

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon[aiobotocore] ==8.2.0</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,5.1</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3.4.2</u>

Type de connexion	Package
Connexion au céleri	<u>apache-airflow-providers-celery==3.2.1</u>
Connexion HTTP	<u>apache-airflow-providers-http==4.4.2</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3.2.2</u>
SQL commun	<u>apache-airflow-providers-common-sql==1,5.2</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3.4.2</u>

v2.5.1

Type de connexion	Package
AWS Raccordement	<u>apache-airflow-providers-amazon==7,10</u>
Connexion Postgres	<u>apache-airflow-providers-postgres==5,4.0</u>
Connexion FTP	<u>apache-airflow-providers-ftp==3,3,0</u>
Connexion au céleri	<u>apache-airflow-providers-celery==3,10</u>
Connexion HTTP	<u>apache-airflow-providers-http==4.1.1</u>
Connexion IMAP	<u>apache-airflow-providers-imap==3.1.1</u>
SQL commun	<u>apache-airflow-providers-common-sql==1.3.3</u>
SQLite Raccordement	<u>apache-airflow-providers-sqlite==3.3.1</u>

Type de connexion	Package

v2.4.3

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon==6.0.0
Connexion Postgres	apache-airflow-providers-postgres==5.2.2
Connexion FTP	apache-airflow-providers-ftp==3.10
Connexion au céleri	apache-airflow-providers-celery==3.0.0
Connexion HTTP	apache-airflow-providers-http==4.0.0
Connexion IMAP	apache-airflow-providers-imap==3.0.0
SQL commun	apache-airflow-providers-common-sql==1.2.0
SQLite Raccordement	apache-airflow-providers-sqlite==3.2.1

Vue d'ensemble des types de connexion

Apache Airflow enregistre les connexions sous forme de chaîne d'URI de connexion. Il fournit un modèle de connexion dans l'interface utilisateur d'Apache Airflow pour générer la chaîne d'URI de connexion, quel que soit le type de connexion. Si aucun modèle de connexion n'est disponible dans l'interface utilisateur d'Apache Airflow, un autre modèle de connexion peut être utilisé pour générer cette chaîne d'URI de connexion, par exemple en utilisant le modèle de connexion HTTP. La principale différence réside dans le préfixe d'URI, par exemple `my-conn-type://`, que les fournisseurs Apache Airflow ignorent généralement pour une connexion. Cette page explique

comment utiliser les modèles de connexion dans l'interface utilisateur d'Apache Airflow de manière interchangeable pour différents types de connexion.

Warning

Ne remplacez pas la [aws_default](#) connexion dans Amazon MWAA. Amazon MWAA utilise cette connexion pour effectuer diverses tâches critiques, telles que la collecte de journaux de tâches. Le remplacement de cette connexion peut entraîner une perte de données et perturber la disponibilité de votre environnement.

Rubriques

- [Exemple de chaîne d'URI de connexion](#)
- [Exemple de modèle de connexion](#)
- [Exemple d'utilisation d'un modèle de connexion HTTP pour une connexion Jdbc](#)

Exemple de chaîne d'URI de connexion

L'exemple suivant présente une chaîne d'URI de connexion pour le type de connexion MySQL.

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

Exemple de modèle de connexion

Les exemples suivants illustrent le modèle de connexion HTTP dans l'interface utilisateur d'Apache Airflow.

Apache Airflow v3

Apache Airflow v2

Exemple d'utilisation d'un modèle de connexion HTTP pour une connexion Jdbc

Utilisez l'exemple suivant pour appliquer le modèle de connexion HTTP pour un type de connexion Jdbc dans l'interface utilisateur d'Apache Airflow.

Apache Airflow v3

L'exemple suivant affiche la chaîne d'URI de connexion générée par Apache Airflow pour l'exemple présenté dans cette section.

```
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
```

Utilisez l'exemple suivant pour appliquer le modèle de connexion HTTP pour une connexion Jdbc pour Apache Airflow v3 dans l'interface utilisateur d'Apache Airflow.

Apache Airflow v2

L'exemple suivant affiche la chaîne d'URI de connexion générée par Apache Airflow pour l'exemple présenté dans cette section.

```
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
```

Utilisez l'exemple suivant pour appliquer le modèle de connexion HTTP pour une connexion Jdbc pour Apache Airflow v2 dans l'interface utilisateur d'Apache Airflow.

Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager

AWS Secrets Manager est un backend Apache Airflow alternatif pris en charge sur un environnement Amazon Managed Workflows pour Apache Airflow. Cette rubrique explique comment stocker en toute

sécurité AWS Secrets Manager les secrets des variables Apache Airflow et d'une connexion Apache Airflow sur Amazon Managed Workflows pour Apache Airflow.

 Note

- Les secrets que vous créez vous sont facturés. Pour plus d'informations sur les tarifs de Secrets Manager, reportez-vous à la section [AWS Tarification](#).
- [AWS Systems Manager Parameter Store](#) est également pris en charge en tant que backend de secrets dans Amazon MWAA. Pour plus d'informations, consultez la [documentation Amazon Provider Package](#).

Table des matières

- [Étape 1 : donnez à Amazon MWAA l'autorisation d'accéder aux clés secrètes de Secrets Manager](#)
- [Deuxième étape : créer le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow](#)
- [Troisième étape : générer une chaîne d'URI de AWS connexion Apache Airflow](#)
- [Étape 4 : ajouter les variables dans Secrets Manager](#)
- [Étape 5 : ajouter la connexion dans Secrets Manager](#)
- [Exemple de code](#)
- [Ressources](#)
- [Quelle est la prochaine étape ?](#)

Étape 1 : donnez à Amazon MWAA l'autorisation d'accéder aux clés secrètes de Secrets Manager

Le [rôle d'exécution](#) de votre environnement Amazon MWAA nécessite un accès en AWS Secrets Manager lecture à la clé secrète. La politique IAM suivante autorise l'accès en lecture-écriture à l'aide de la AWS stratégie -managed. [SecretsManagerReadWrite](#)

Pour associer la politique à votre rôle d'exécution

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez votre rôle d'exécution dans le volet Autorisations.

4. Choisissez Attacher des politiques.
5. Tapez SecretsManagerReadWrite dans le champ de texte Politiques de filtrage.
6. Choisissez Attach policy (Attacher la politique).

Si vous ne souhaitez pas utiliser de politique d'autorisation AWS gérée, vous pouvez directement mettre à jour le rôle d'exécution de votre environnement pour autoriser n'importe quel niveau d'accès à vos ressources de Secrets Manager. Par exemple, la déclaration de politique suivante accorde un accès en lecture à tous les secrets que vous créez dans un espace spécifique Région AWS dans Secrets Manager.

JSON

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

Deuxième étape : créer le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow

La section suivante décrit comment créer une option de configuration Apache Airflow sur la console Amazon MWAA pour le backend. AWS Secrets Manager Si vous utilisez un paramètre de

configuration du même nom dans `airflow.cfg`, la configuration que vous créez dans les étapes suivantes est prioritaire et remplace les paramètres de configuration.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Choisissez Ajouter une configuration personnalisée dans le volet des options de configuration d'Airflow. Ajoutez les paires clé-valeur suivantes :
 - a. **secrets.backend:**
`airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend`
 - b. **secrets.backend_kwargs:** **`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`** Cela permet de configurer Apache Airflow pour rechercher des chaînes de connexion et des variables `airflow/connections/*` sur `airflow/variables/*` et des chemins.

Vous pouvez utiliser un [modèle de recherche](#) pour réduire le nombre d'appels d'API qu'Amazon MWAA envoie à Secrets Manager en votre nom. Si vous ne spécifiez aucun modèle de recherche, Apache Airflow recherche toutes les connexions et variables dans le backend configuré. En spécifiant un modèle, vous réduisez les chemins possibles recherchés par Apache Airflow. Cela réduit vos coûts lorsque vous utilisez Secrets Manager avec Amazon MWAA.

Pour spécifier un modèle de recherche, spécifiez les `variables_lookup_pattern` paramètres `connections_lookup_pattern` et. Ces paramètres acceptent une RegEx chaîne en entrée. Par exemple, pour rechercher des secrets commençant par `test`, entrez ce qui suit pour `secrets.backend_kwargs` :

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

Note

Pour utiliser `connections_lookup_pattern` et `variables_lookup_pattern`, vous devez installer `apache-airflow-providers-amazon` la version 7.3.0 ou supérieure. Pour plus d'informations sur la mise à jour des packages des fournisseurs vers des versions plus récentes, reportez-vous à [Fichier de contraintes](#)

6. Choisissez Enregistrer.

Troisième étape : générer une chaîne d'URI de AWS connexion Apache Airflow

Pour créer une chaîne de connexion, utilisez la touche « tab » de votre clavier pour mettre en retrait les paires clé-valeur dans l'objet `Connection`. Nous vous recommandons également de créer une variable pour l'extraobjet dans votre session shell. La section suivante explique les étapes à suivre pour générer une chaîne d'URI de connexion Apache Airflow pour un environnement Amazon MWAA à l'aide d'Apache Airflow ou d'un script Python.

Apache Airflow CLI

La session shell suivante utilise votre CLI Airflow locale pour générer une chaîne de connexion. Si la CLI n'est pas installée, nous vous recommandons d'utiliser le script Python.

1. Ouvrez une session shell Python :

```
python3
```

2. Entrez la commande suivante :

```
>>> import json
```

3. Entrez la commande suivante :

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

4. Créez une variable pour l'extraobjet dans votre session shell. Remplacez les valeurs d'échantillon `YOUR_EXECUTION_ROLE_ARN` par le rôle d'exécution ARN et la région par `us-east-1` (tel que `us-east-1`).


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

5. Créez l'objet de connexion. Remplacez la valeur d'échantillon `myconn` par le nom de la connexion Apache Airflow.

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

6. Utilisez la touche « tab » de votre clavier pour mettre en retrait chacune des paires clé-valeur suivantes dans votre objet de connexion. Remplacez les valeurs d'échantillon par `red`.

- a. Spécifiez le type de AWS connexion :

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

- b. Spécifiez l'option de base de données Apache Airflow :

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

- c. Spécifiez l'URL de l'interface utilisateur Apache Airflow sur Amazon MWAA :

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

- d. Spécifiez l'ID de clé d' AWS accès (nom d'utilisateur) pour vous connecter à Amazon MWAA :

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

- e. Spécifiez la clé d'accès AWS secrète (mot de passe) pour vous connecter à Amazon MWAA :

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

- f. Spécifiez la variable de session extra shell :

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

- g. Fermez l'objet de connexion.

```
... )
```

7. Imprimez la chaîne d'URI de connexion :

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

Reportez-vous à la chaîne d'URI de connexion dans la réponse :

```
'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

Le script Python suivant ne nécessite pas la CLI Apache Airflow.

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `mwaa_connection.py`.

```
import urllib.parse

conn_type = 'YOUR_DB_OPTION'
host = 'YOUR_MWAA_AIRFLOW_UI_URL'
port = 'YOUR_PORT'
login = 'YOUR_AWS_ACCESS_KEY_ID'
password = 'YOUR_AWS_SECRET_ACCESS_KEY'
role_arn = urllib.parse.quote_plus('YOUR_EXECUTION_ROLE_ARN')
region_name = 'us-east-1'

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

2. Remplacez les espaces réservés par *red*
3. Exécutez le script suivant pour générer une chaîne de connexion.

```
python3 mwaa_connection.py
```

Étape 4 : ajouter les variables dans Secrets Manager

La section suivante décrit comment créer le secret d'une variable dans Secrets Manager.

Pour créer le secret

1. Ouvrez la [AWS Secrets Manager console](#).
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Choisissez Autre type de secret.
4. Dans le volet Spécifier les key/value paires à stocker dans ce secret, choisissez Plaintext.
5. Ajoutez la valeur de la variable en texte brut au format suivant.

```
"YOUR_VARIABLE_VALUE"
```

Par exemple, pour spécifier un entier :

```
14
```

Par exemple, pour spécifier une chaîne :

```
"mystring"
```

6. Pour Clé de chiffrement, choisissez une option de AWS KMS clé dans la liste déroulante.
7. Entrez un nom dans le champ de texte pour le nom du secret au format suivant.

```
airflow/variables/YOUR_VARIABLE_NAME
```

Par exemple :

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

8. Choisissez Suivant.
9. Sur la page Configurer le secret, dans le volet Nom et description du secret, procédez comme suit.
 - a. Dans Nom du secret, saisissez le nom de votre secret.
 - b. (Facultatif) Dans Description, fournissez une description de votre secret.

Choisissez Suivant.

10. Dans le champ Configurer la rotation - facultatif, laissez les options par défaut et choisissez Next.
11. Répétez ces étapes dans Secrets Manager pour toutes les variables supplémentaires que vous souhaitez ajouter.
12. Sur la page Révision, vérifiez votre secret, puis choisissez Store.

Étape 5 : ajouter la connexion dans Secrets Manager

La section suivante décrit comment créer le secret pour l'URI de votre chaîne de connexion dans Secrets Manager.

Pour créer le secret

1. Ouvrez la [AWS Secrets Manager console](#).
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Choisissez Autre type de secret.
4. Dans le volet Spécifier les key/value paires à stocker dans ce secret, choisissez Plaintext.
5. Ajoutez la chaîne d'URI de connexion en texte brut au format suivant.

YOUR_CONNECTION_URI_STRING

Par exemple :

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

Warning

Apache Airflow analyse chacune des valeurs de la chaîne de connexion. Vous ne devez pas utiliser de guillemets simples ou doubles, sinon cela analyse la connexion comme une chaîne unique.

6. Pour Clé de chiffrement, choisissez une option de AWS KMS clé dans la liste déroulante.

- Entrez un nom dans le champ de texte pour le nom du secret au format suivant.

```
airflow/connections/YOUR_CONNECTION_NAME
```

Par exemple :

```
airflow/connections/myconn
```

- Choisissez Suivant.
- Sur la page Configurer le secret, dans le volet Nom et description du secret, procédez comme suit.
 - Dans Nom du secret, saisissez le nom de votre secret.
 - (Facultatif) Dans Description, fournissez une description de votre secret.

Choisissez Suivant.

- Dans le champ Configurer la rotation - facultatif, laissez les options par défaut et choisissez Next.
- Répétez ces étapes dans Secrets Manager pour toutes les variables supplémentaires que vous souhaitez ajouter.
- Sur la page Révision, vérifiez votre secret, puis choisissez Store.

Exemple de code

- Découvrez comment utiliser la clé secrète pour la connexion Apache Airflow (myconn) sur cette page à l'aide de l'exemple de code disponible à l'adresse. [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow](#)
- Découvrez comment utiliser la clé secrète de la variable Apache Airflow (test-variable) sur cette page à l'aide de l'exemple de code disponible à l'adresse. [Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow](#)

Ressources

- Pour plus d'informations sur la configuration des secrets de Secrets Manager à l'aide de la console et du AWS CLI, reportez-vous à la section [Créer un secret](#) dans le Guide de AWS Secrets Manager l'utilisateur.

- Utilisez un script Python pour migrer un grand volume de variables et de connexions Apache Airflow vers Secrets Manager dans [Move your Apache Airflow connections](#) and variables to. AWS Secrets Manager

Quelle est la prochaine étape ?

- Découvrez comment générer un jeton pour accéder à l'interface utilisateur d'Apache Airflow dans. [Accès à Apache Airflow](#)

Gestion des environnements Amazon MWAA

La console Amazon Managed Workflows pour Apache Airflow contient des options intégrées permettant de configurer un accès privé ou public à l'interface utilisateur d'Apache Airflow. Il contient également des options intégrées pour configurer la taille de l'environnement, le moment où il convient de dimensionner les travailleurs, ainsi que des options de configuration d'Apache Airflow que vous pouvez utiliser pour remplacer les configurations d'Apache Airflow qui ne sont normalement accessibles que dans `airflow.cfg`. Ce chapitre explique comment utiliser ces configurations sur la console Amazon MWAA.

Rubriques

- [Configuration de la classe d'environnement Amazon MWAA](#)
- [Configuration du dimensionnement automatique d'Amazon MWAA Worker](#)
- [Configuration du dimensionnement automatique du serveur Web Amazon MWAA](#)
- [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#)
- [Mettre à jour un environnement Amazon MWAA](#)
- [Modification de la version d'Apache Airflow](#)
- [Utilisation d'un script de démarrage avec Amazon MWAA](#)

Configuration de la classe d'environnement Amazon MWAA

La classe d'environnement que vous choisissez pour votre environnement Amazon MWAA détermine la taille des AWS Fargate conteneurs AWS gérés dans lesquels le [Celery Executor](#) s'exécute, ainsi que la base de données de métadonnées AWS Amazon Aurora PostgreSQL gérée dans laquelle les planificateurs Apache Airflow créent des instances de tâches. Cette rubrique décrit chaque classe d'environnement Amazon MWAA et explique comment mettre à jour la classe d'environnement sur la console Amazon MWAA.

Sections

- [Capacités environnementales](#)
- [Planificateurs Apache Airflow](#)

Capacités environnementales

La section suivante contient les tâches Apache Airflow simultanées par défaut, la mémoire vive (RAM) et les unités de traitement centralisées virtuelles (vCPUs) pour chaque classe d'environnement. Les tâches simultanées répertoriées supposent que la simultanéité des tâches ne dépasse pas la capacité de travail d'Apache Airflow dans l'environnement.

Dans le tableau suivant, la capacité du DAG fait référence aux définitions du DAG, et non aux exécutions, et suppose que vous DAGs êtes [dynamique](#) dans un seul fichier Python écrit selon les [meilleures pratiques d'Apache Airflow](#).

Les exécutions de tâches dépendent du nombre de tâches planifiées simultanément et supposent que le nombre d'exécutions DAG définies pour démarrer en même temps ne dépasse pas le nombre par défaut [max_dagruns_per_loop_to_schedule](#), ainsi que la taille et le nombre de travailleurs, comme indiqué dans cette rubrique.

mw1.micro

- Capacité jusqu'à 25 DAG
- 3 tâches simultanées (par défaut)
- Composants :
 - Serveur Web : 1 vCPU, 3 Go de RAM
 - Travailleur et planificateur : 1 vCPU, 3 Go de RAM
 - Base de données : 2 vCPU, 4 Go de RAM



Note

mw1.micro ne prend pas en charge l'auto-scaling.

mw1.small

- Capacité jusqu'à 50 DAG
- 5 tâches simultanées (par défaut)
- Composants :
 - Serveurs Web : 1 vCPU, 2 Go de RAM chacun
 - Travailleurs : 1 vCPU, 2 Go de RAM chacun

- Planificateurs : 1 vCPU, 2 Go de RAM chacun
- Base de données : 2 vCPU, 4 Go de RAM

mw1.medium

- Capacité jusqu'à 250 DAG
- 10 tâches simultanées (par défaut)
- Composants :
 - Serveurs Web : 1 vCPU de 2 Go de RAM chacun
 - Travailleurs : 2 vCPU de 4 Go de RAM chacun
 - Planificateurs : 2 vCPU de 4 Go de RAM chacun
 - Base de données : 2 vCPU 8 Go de RAM

mw1.large

- Capacité jusqu'à 1000 DAG
- 20 tâches simultanées (par défaut)
- Composants :
 - Serveurs Web : 2 vCPU de 4 Go de RAM chacun
 - Travailleurs : 4 vCPU de 8 Go de RAM chacun
 - Planificateurs : 4 vCPU de 8 Go de RAM chacun
 - Base de données : 2 vCPU 8 Go de RAM

mw1.xlarge

- Capacité jusqu'à 2000 DAG
- 40 tâches simultanées (par défaut)
- Composants :
 - Serveurs Web : 4 vCPU de 12 Go de RAM chacun
 - Travailleurs : 8 vCPU de 24 Go de RAM chacun
 - Planificateurs : 8 vCPU (24 Go de RAM chacun)
 - Base de données : 4 vCPU 32 Go de RAM

mw1.2xlarge

- Capacité jusqu'à 4000 DAG
- 80 tâches simultanées (par défaut)
- Composants :
 - Serveurs Web : 8 vCPU de 24 Go de RAM chacun
 - Travailleurs : 16 vCPU de 48 Go de RAM chacun
 - Planificateurs : 16 vCPU (48 Go de RAM chacun)
 - Base de données : 8 vCPU 64 Go de RAM

Vous pouvez l'utiliser `celery.worker_autoscale` pour augmenter le nombre de tâches par travailleur. Pour plus d'informations, reportez-vous au [the section called “Exemple de cas d'utilisation à hautes performances”](#).

Planificateurs Apache Airflow

La section suivante décrit les options du planificateur Apache Airflow disponibles sur Amazon MWAA et explique comment le nombre de planificateurs affecte le nombre de déclencheurs.

Dans Apache Airflow, un [déclencheur](#) gère les tâches qu'il reporte jusqu'à ce que certaines conditions spécifiées à l'aide d'un déclencheur soient remplies. Dans Amazon MWAA, le déclencheur s'exécute parallèlement au planificateur sur la même tâche Fargate. L'augmentation du nombre de planificateurs augmente en conséquence le nombre de déclencheurs disponibles, optimisant ainsi la façon dont l'environnement gère les tâches différées. Cela garantit une gestion efficace des tâches, en les planifiant rapidement pour qu'elles s'exécutent lorsque les conditions sont satisfaites.

Apache Airflow v3

- v3 - Pour les environnements supérieurs à mw1.micro, accepte les valeurs comprises entre 1 et 5. La valeur par défaut est 2 pour toutes les tailles d'environnement sauf mw1.micro, qui est définie par défaut sur 1.

Apache Airflow v2

- v2 - Pour les environnements supérieurs à mw1.micro, accepte les valeurs comprises entre 1 et 5. La valeur par défaut est 2 pour toutes les tailles d'environnement sauf mw1.micro, qui est définie par défaut sur 1.

Configuration du dimensionnement automatique d'Amazon MWAA Worker

Le mécanisme de dimensionnement automatique augmente automatiquement le nombre de travailleurs Apache Airflow en réponse aux tâches en cours d'exécution et en file d'attente dans votre environnement Amazon Managed Workflows for Apache Airflow et élimine les travailleurs supplémentaires lorsqu'il n'y a plus de tâches en file d'attente ou en cours d'exécution. Cette rubrique décrit comment configurer le dimensionnement automatique en spécifiant le nombre maximum de travailleurs Apache Airflow exécutés sur votre environnement à l'aide de la console Amazon MWAA.

Note

Amazon MWAA utilise les métriques d'Apache Airflow pour déterminer quand [des travailleurs supplémentaires de Celery](#) Executor sont nécessaires et, le cas échéant, augmente le nombre de travailleurs Fargate jusqu'à la valeur spécifiée par `max-workers`. Au fur et à mesure que les travailleurs supplémentaires terminent leur travail et que la charge de travail diminue, Amazon MWAA les supprime, réduisant ainsi la taille à la valeur définie par `min-workers`.

Si les travailleurs prennent en charge de nouvelles tâches lors de la réduction d'échelle, Amazon MWAA conserve la ressource Fargate et ne supprime pas le travailleur. Pour plus d'informations, consultez [Comment fonctionne le dimensionnement automatique d'Amazon MWAA](#).

Sections

- [Comment fonctionne Worker Scaling](#)
- [Utilisation de la console Amazon MWAA](#)
- [Exemple de cas d'utilisation à hautes performances](#)
- [Tâches de dépannage bloquées en cours d'exécution](#)
- [Quelle est la prochaine étape ?](#)

Comment fonctionne Worker Scaling

Usages `RunningTasks` et `QueuedTasks` [statistiques](#) d'Amazon MWAA, où (tâches en cours d'exécution + tâches en file d'attente)/([tâches par travailleur](#)) = (travailleurs requis). Si le nombre

de travailleurs requis est supérieur au nombre actuel de travailleurs, Amazon MWAA ajoutera les conteneurs de travailleurs Fargate à cette valeur, jusqu'à la valeur maximale spécifiée par `max-workers`

À mesure que la charge de travail diminue et que la `RunningTasks` somme des `QueuedTasks` mesures diminue, Amazon MWAA demande à Fargate de réduire le nombre de travailleurs en fonction de l'environnement. Tous les travailleurs qui terminent encore leur travail restent protégés pendant la réduction d'échelle jusqu'à ce qu'ils aient terminé leur travail. En fonction de la charge de travail, les tâches peuvent être mises en file d'attente pendant que les employés réduisent leur effectif.

Utilisation de la console Amazon MWAA

Vous pouvez choisir le nombre maximum de travailleurs pouvant s'exécuter simultanément sur votre environnement sur la console Amazon MWAA. Par défaut, vous pouvez spécifier une valeur maximale de 25.

Pour configurer le nombre de travailleurs

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Dans le volet Classe d'environnement, entrez une valeur dans Nombre maximal de travailleurs.
6. Choisissez Enregistrer.

Note

Quelques minutes peuvent s'écouler avant que les modifications ne prennent effet sur votre environnement.

Exemple de cas d'utilisation à hautes performances

La section suivante décrit le type de configuration que vous pouvez utiliser pour activer les hautes performances et le parallélisme dans un environnement.

Apache Airflow sur site

Généralement, sur une plateforme Apache Airflow sur site, vous configurez les paramètres de parallélisme des tâches, de mise à l'échelle automatique et de simultanéité dans votre fichier : `airflow.cfg`

- `core.parallelism`— Le nombre maximum d'instances de tâches pouvant être exécutées simultanément par planificateur.
- `core.dag_concurrency`— La simultanéité maximale pour DAGs (et non pour les travailleurs).
- `celery.worker_autoscale`— Le nombre maximum et minimum de tâches pouvant être exécutées simultanément sur n'importe quel travailleur.

Par exemple, si ce `core.parallelism` paramètre était défini sur 100 et `core.dag_concurrency` était défini sur 7, vous ne pouvez exécuter un total de 14 tâches simultanément que si vous en avez 2 DAGs. Cela étant dit, chaque DAG est configuré pour exécuter uniquement sept tâches simultanément (`core.dag_concurrency`), même si le parallélisme global est défini sur 100 (`core.parallelism`).

Note

`core.dag_concurrency` n'est pas disponible dans Apache Airflow v3.

Dans un environnement Amazon MWAA

Dans un environnement Amazon MWAA, vous pouvez configurer ces paramètres directement sur la console Amazon MWAA à l'aide [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#) du [Configuration de la classe d'environnement Amazon MWAA](#) mécanisme de dimensionnement automatique du nombre maximal de travailleurs. Bien qu'elle ne `core.dag_concurrency` soit pas disponible dans la liste déroulante en tant qu'option de configuration Apache Airflow sur la console Amazon MWAA, vous pouvez l'ajouter en tant qu'option de configuration [Apache](#) Airflow personnalisée.

Supposons que lorsque vous avez créé votre environnement, vous avez choisi les paramètres suivants :

1. La [classe d'environnement](#) `mw1.small` qui contrôle le nombre maximum de tâches simultanées que chaque travailleur peut exécuter par défaut, ainsi que le vCPU des conteneurs.

2. Le paramètre par défaut des 10 travailleurs dans Nombre maximal de travailleurs.
3. Une [option de configuration d'Apache Airflow](#) pour le nombre `celery.worker_autoscale` de 5, 5 tâches par travailleur.

Cela signifie que vous pouvez exécuter 50 tâches simultanément dans votre environnement. Toutes les tâches supérieures à 50 sont mises en file d'attente et attendent que les tâches en cours soient terminées.

Exécutez davantage de tâches simultanées. Vous pouvez modifier votre environnement pour exécuter davantage de tâches simultanément à l'aide des configurations suivantes :

1. Augmentez le nombre maximum de tâches simultanées que chaque travailleur peut exécuter par défaut et le nombre de vCPU des conteneurs en choisissant la classe d'[environnement `mw1.medium`](#) (10 tâches simultanées par défaut).
2. Ajouter `celery.worker_autoscale` en tant qu'option de [configuration d'Apache Airflow](#).
3. Augmentez le nombre maximum de travailleurs. Dans cet exemple, l'augmentation du nombre maximum de travailleurs de 10 à 20 double le nombre de tâches simultanées que l'environnement peut exécuter.

Spécifiez le nombre minimum de travailleurs. Vous pouvez également spécifier le nombre minimum et maximum de travailleurs Apache Airflow exécutés dans votre environnement à l'aide du AWS Command Line Interface (AWS CLI). Par exemple :

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

Pour en savoir plus, reportez-vous à la commande [update-environment](#) dans le AWS CLI

Tâches de débannage bloquées en cours d'exécution

Dans de rares cas, Apache Airflow peut penser que des tâches sont toujours en cours d'exécution. Pour résoudre ce problème, vous devez effacer la tâche bloquée dans l'interface utilisateur d'Apache Airflow. Pour plus d'informations, reportez-vous à la rubrique de [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#) résolution des problèmes.

Quelle est la prochaine étape ?

- Découvrez les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement [Optimisation des performances pour Apache Airflow sur Amazon MWAA](#).

Configuration du dimensionnement automatique du serveur Web Amazon MWAA

Pour les environnements exécutant Apache Airflow v2.2.2 et versions ultérieures, Amazon MWAA adapte dynamiquement vos serveurs Web pour gérer les charges de travail fluctuantes, ce qui permet d'éviter les problèmes de performances lors des pics de charge. En dimensionnant automatiquement le nombre de serveurs Web en fonction de l'utilisation du processeur et du nombre de connexions actives, Amazon MWAA garantit que votre environnement Apache Airflow peut facilement répondre à une demande accrue, qu'il s'agisse de demandes d'API REST, de l'utilisation de CLI ou d'un plus grand nombre d'utilisateurs simultanés de l'interface utilisateur Apache Airflow.

Sections

- [Comment fonctionne la mise à l'échelle du serveur Web](#)
- [Utilisation de la console Amazon MWAA](#)

Comment fonctionne la mise à l'échelle du serveur Web

Amazon MWAA utilise la métrique du conteneur et la métrique de l'équilibreur de charge pour déterminer si le dimensionnement des serveurs Web est nécessaire en fonction de la quantité de trafic. [CPUUtilizationActiveConnectionCount](#) S'il CPUUtilization est supérieur à 70 ou ActiveConnectionCount supérieur à 15, Amazon MWAA ajoutera des conteneurs de serveur Web Fargate supplémentaires jusqu'à la valeur maximale spécifiée par. MaxWebServers

À mesure que le trafic diminue et que ActiveConnectionCount les valeurs CPUUtilization et diminuent, Amazon MWAA demande à Fargate de réduire le nombre de conteneurs de serveurs Web pour l'environnement à la valeur minimale définie par. MinimumWebServers

Utilisation de la console Amazon MWAA

Vous pouvez choisir le nombre de serveurs Web qui peuvent s'exécuter simultanément sur votre environnement sur la console Amazon MWAA. Par défaut, le nombre minimum de serveurs Web est de deux, et le nombre maximum de serveurs Web est de cinq.

Pour configurer le nombre de serveurs Web

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Dans le volet Classe d'environnement, entrez une valeur dans Nombre maximal de serveurs Web.
6. Entrez ensuite une valeur dans le champ Nombre minimal de serveurs Web.
7. Choisissez Enregistrer.

Note

Quelques minutes peuvent s'écouler avant que les modifications ne prennent effet sur votre environnement.

Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA

Les options de configuration d'Apache Airflow peuvent être associées à votre environnement Amazon Managed Workflows for Apache Airflow en tant que variables d'environnement. Vous pouvez choisir dans la liste déroulante suggérée ou spécifier des options de configuration personnalisées pour votre version d'Apache Airflow sur la console Amazon MWAA. Cette rubrique décrit les options de configuration d'Apache Airflow disponibles et explique comment utiliser ces options pour remplacer les paramètres de configuration d'Apache Airflow dans votre environnement.

Table des matières

- [Prérequis](#)

- [Comment ça marche](#)
- [Utilisation des options de configuration pour charger des plugins](#)
- [Vue d'ensemble des options de configuration](#)
 - [Options de configuration Apache Airflow](#)
 - [Référence Apache Airflow](#)
 - [Utilisation de la console Amazon MWAA](#)
- [Référence de configuration](#)
 - [Configurations des e-mails](#)
 - [Configurations des tâches](#)
 - [Configurations du planificateur](#)
 - [Configurations des travailleurs](#)
 - [Configurations du serveur Web](#)
 - [Configurations du déclencheur](#)
- [Exemples et exemple de code](#)
 - [Exemple de DAG](#)
 - [Exemples de paramètres de notification par e-mail](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Vous aurez besoin des éléments suivants avant de pouvoir effectuer les étapes indiquées sur cette page.

- **Autorisations** — Votre administrateur Compte AWS doit vous avoir accordé l'accès à la politique de contrôle MWAAFull ConsoleAccess d'accès [Amazon](#) pour votre environnement. En outre, votre environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder aux AWS ressources utilisées par votre environnement.
- **Accès** : si vous avez besoin d'accéder à des référentiels publics pour installer des dépendances directement sur le serveur Web, votre environnement doit être configuré avec un accès au serveur Web du réseau public. Pour plus d'informations, consultez [the section called "Modes d'accès Apache Airflow"](#).

- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAGs plugins personnalisés et vos dépendances Python `requirements.txt` doit être configuré avec l'accès public bloqué et le versionnage activé. `plugins.zip`

Comment ça marche

Lorsque vous créez un environnement, Amazon MWAA joint les paramètres de configuration que vous spécifiez sur la console Amazon MWAA dans les options de configuration d'Airflow en tant que variables d'environnement au AWS Fargate conteneur de votre environnement. Si vous utilisez un paramètre du même nom dans `airflow.cfg`, les options que vous spécifiez sur la console Amazon MWAA remplacent les valeurs dans `airflow.cfg`

Bien que nous ne les exposons pas `airflow.cfg` dans l'interface utilisateur Apache Airflow d'un environnement Amazon MWAA par défaut, vous pouvez modifier les options de configuration d'Apache Airflow directement sur la console Amazon MWAA, y compris les paramètres permettant d'exposer les configurations. `webserver.expose_config`

Utilisation des options de configuration pour charger des plugins

Par défaut, dans Apache Airflow v2 et versions ultérieures, les plugins sont configurés pour être chargés « paresseusement » à l'aide de ce paramètre. `core.lazy_load_plugins : True` Si vous utilisez des plugins personnalisés, vous devez les ajouter `core.lazy_load_plugins : False` en tant qu'option de configuration d'Apache Airflow pour charger les plugins au début de chaque processus Airflow afin de remplacer le paramètre par défaut.

Vue d'ensemble des options de configuration

Lorsque vous ajoutez une configuration sur la console Amazon MWAA, Amazon MWAA écrit la configuration en tant que variable d'environnement.

- Options répertoriées. Vous pouvez choisir l'un des paramètres de configuration disponibles pour votre version d'Apache Airflow dans la liste déroulante. Par exemple, `dag_concurrency : 16`. Le paramètre de configuration est traduit dans le conteneur Fargate de votre environnement sous la forme `AIRFLOW__CORE__DAG_CONCURRENCY : 16`
- Options personnalisées. Vous pouvez également spécifier les options de configuration d'Airflow qui ne sont pas répertoriées pour votre version d'Apache Airflow dans la liste déroulante. Par exemple, `foo.user : YOUR_USER_NAME`. Le paramètre de configuration est traduit dans le conteneur Fargate de votre environnement sous la forme `AIRFLOW__FOO__USER : YOUR_USER_NAME`

Options de configuration Apache Airflow

L'image suivante montre où vous pouvez personnaliser les options de configuration d'Apache Airflow sur la console Amazon MWAA.

Référence Apache Airflow

Pour obtenir la liste des options de configuration prises en charge par Apache Airflow, reportez-vous à la section [Configuration Reference du guide de référence](#) d'Apache Airflow. Pour accéder aux options de la version d'Apache Airflow que vous utilisez sur Amazon MWAA, sélectionnez la version dans la liste déroulante.

Utilisation de la console Amazon MWAA

La procédure suivante explique les étapes à suivre pour ajouter une option de configuration Airflow à votre environnement.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Choisissez Ajouter une configuration personnalisée dans le volet des options de configuration d'Airflow.
6. Choisissez une configuration dans la liste déroulante et entrez une valeur, ou entrez une configuration personnalisée et entrez une valeur.
7. Choisissez Ajouter une configuration personnalisée pour chaque configuration que vous souhaitez ajouter.
8. Choisissez Enregistrer.

Référence de configuration

La section suivante contient la liste des configurations Apache Airflow disponibles dans la liste déroulante de la console Amazon MWAA.

Configurations des e-mails

La liste suivante répertorie les options de configuration des notifications par e-mail Airflow disponibles sur Amazon MWAA pour Apache Airflow v2 et v3.

Nous recommandons d'utiliser le port 587 pour le trafic SMTP. AWS Bloque par défaut le trafic SMTP sortant sur le port 25 de toutes les instances Amazon EC2 . Si vous souhaitez envoyer du trafic sortant sur le port 25, vous pouvez [demander la suppression de cette restriction](#).

Option de configuration du flux d'air	Description	Exemple de valeur
email.email_backend	L'utilitaire Apache Airflow utilisé pour les notifications par e-mail dans email_backend.	airflow.utils.email.send_email_smtp
smtp.smtp_host	Nom du serveur sortant utilisé pour l'adresse e-mail dans smtp_host .	localhost
smtp.smtp_starttls	Le protocole TLS (Transport Layer Security) est utilisé pour chiffrer le courrier électronique sur Internet dans smtp_starttls.	False
smtp.smtp_ssl	Le protocole SSL (Secure Sockets Layer) est utilisé pour connecter le serveur et le client de messagerie dans smtp_ssl .	True
smtp.smtp_port	Le port TCP (Transmission Control Protocol) désigné pour le serveur dans smtp_port .	587

Option de configuration du flux d'air	Description	Exemple de valeur
smtp.smtp_mail_from	Adresse e-mail sortante dans smtp_mail_from .	myemail@domain.com

Configurations des tâches

La liste suivante affiche les configurations disponibles dans la liste déroulante pour les tâches Airflow sur Amazon MWAA pour Apache Airflow v2 et v3.

Option de configuration du flux d'air	Description	Exemple de valeur
core.default_task_retries	Nombre de tentatives d'exécution d'une tâche Apache Airflow dans default_task_retries.	3
core.parallélisme	Nombre maximal d'instances de tâches pouvant être exécutées simultanément dans l'ensemble de l'environnement en parallèle (parallélisme).	40

Configurations du planificateur

La liste suivante répertorie les configurations du planificateur Apache Airflow disponibles dans la liste déroulante d'Amazon MWAA pour Apache Airflow v2 et v3.

Option de configuration du flux d'air	Description	Exemple de valeur
scheduler.catchup_by_default		False

Option de configuration du flux d'air	Description	Exemple de valeur
	Indique au planificateur de créer un DAG exécuté pour « rattraper » l'intervalle de temps spécifique indiqué dans <code>catchup_by_default</code>.	
scheduler.scheduler_zombie_task_threshold  Note Non disponible dans Apache Airflow v3.	Indique au planificateur s'il convient de marquer l'instance de tâche comme ayant échoué et de replanifier la tâche dans <code>scheduler_zombie_task_threshold</code>.	300

Configurations des travailleurs

La liste suivante affiche les configurations de travail Airflow disponibles dans la liste déroulante d'Amazon MWAA pour Apache Airflow v2 et v3.

Option de configuration du flux d'air	Description	Exemple de valeur
celery.worker_autoscale	Le nombre maximum et minimum de tâches pouvant être exécutées simultanément sur n'importe quel travailleur utilisant le Celery Executor dans <code>worker_autoscale</code>. Les valeurs doivent être séparées par des virgules dans l'ordre suivant :	16,12

Option de configuration du flux d'air	Description	Exemple de valeur

`max_concurrency, min_concurrency`

Configurations du serveur Web

La liste suivante répertorie les configurations du serveur Web Apache Airflow disponibles dans la liste déroulante d'Amazon MWAA pour Apache Airflow v2 et v3.

Option de configuration du flux d'air	Description	Exemple de valeur

`webserver.default_ui_timezone`

[Le paramètre de date/heure par défaut de l'interface utilisateur d'Apache Airflow dans `default\_ui\_timezone`.](#)

Amérique/New_York

Note

Non disponible dans Apache Airflow v3.

Note

La définition de `default_ui_timezone` cette option ne modifie pas le fuseau horaire dans lequel votre DAGs course est prévue. Pour modifier le fuseau horaire de votre ordinateur DAGs, vous pouvez utiliser un plugin personnalisé. Pour plus d'informations, consultez [the section](#)

Option de configuration du flux d'air	Description	Exemple de valeur
	called “Modifier le fuseau horaire d'un DAG” .	

Configurations du déclencheur

La liste suivante répertorie les configurations du [déclencheur Apache Airflow disponibles](#) sur Amazon MWAA pour Apache Airflow v2 et v3.

Option de configuration du flux d'air	Description	Exemple de valeur
<code>mwaa.triggerer_enabled</code>	Utilisé pour activer et désactiver le déclencheur sur Amazon MWAA. Par défaut, cette valeur indique <code>True</code> . Si ce paramètre est défini sur <code>False</code> , Amazon MWAA ne lancera aucun processus déclencheur sur les planificateurs.	<code>True</code>
<code>triggerer.default_capacity</code> (dans la version 2) <code>triggerer.capacity</code> (dans la version 3)	Définit le nombre de déclencheurs que chaque déclencheur peut exécuter en parallèle. Sur Amazon MWAA, cette capacité est définie pour chaque déclencheur et pour chaque planificateur, car les deux composants fonctionnent côte à côte. La valeur par défaut par planifica	125

Option de configuration du flux d'air	Description	Exemple de valeur
	teur est définie sur 60, 125 250 500, et 1000 pour les petites, moyennes et grandes instances, xlarge et 2xlarge, respectivement.	

Exemples et exemple de code

Exemple de DAG

Vous pouvez utiliser le DAG suivant pour imprimer vos options de configuration d'email_backend Apache Airflow. Pour l'exécuter en réponse aux événements Amazon MWAA, copiez le code dans le DAGs dossier de votre environnement sur votre compartiment de stockage Amazon S3.

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

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

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

    print_variable_test = print_variable_dag()
```

Exemples de paramètres de notification par e-mail

Les options de configuration d'Apache Airflow suivantes peuvent être utilisées pour un compte de messagerie Gmail.com à l'aide d'un mot de passe d'application. Pour plus d'informations, reportez-vous à la section Se [connecter à l'aide des mots de passe des applications](#) dans le guide de référence de l'aide Gmail.

Quelle est la prochaine étape ?

- Découvrez comment télécharger votre dossier DAG dans votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Mettre à jour un environnement Amazon MWAA

Note

Les mises à jour progressives d'Amazon MWAA ne sont pas encore prises en charge dans les régions de l'Ouest du Canada (Calgary) et de l'Asie-Pacifique (Malaisie).

Les mises à jour de l'environnement Amazon MWAA appliquent les dernières modifications et correctifs de sécurité. Vous pouvez également modifier les configurations existantes et mettre à niveau la version d'Apache Airflow. Ce guide décrit les étapes de mise à jour d'un environnement Amazon MWAA.

Table des matières

- [Avant de commencer](#)
- [Stratégie de remplacement des travailleurs](#)
- [Mettre à jour les ressources environnementales](#)
- [Mise à jour d'un environnement](#)
 - [Première étape : Spécifier les détails](#)
 - [Deuxième étape : configurer les paramètres avancés](#)
 - [Troisième étape : révision et mise à jour](#)

Avant de commencer

- Le [réseau VPC](#) que vous avez spécifié pour votre environnement ne peut pas être modifié une fois l'environnement créé.
- Vous avez besoin d'un compartiment Amazon S3 configuré pour bloquer tout accès public, avec activation de la gestion des versions des compartiments.
- Vous avez besoin d'un compte AWS d'une [autorisation pour utiliser Amazon MWWA](#) et d'une autorisation in Gestion des identités et des accès AWS (IAM) pour créer des rôles IAM. Si vous choisissez le mode d'accès au réseau privé pour le serveur Web Apache Airflow, qui limite l'accès à Apache Airflow au sein de votre Amazon VPC, vous aurez besoin d'une autorisation dans IAM pour créer des points de terminaison Amazon VPC.
- Pour activer les mises à jour de l'environnement Graceful, vous devez passer à la version 2.4.3 ou supérieure d'Apache Airflow. Pour mettre à niveau la version Airflow, reportez-vous à [Modification de la version d'Apache Airflow](#).

Stratégie de remplacement des travailleurs

Vous pouvez choisir une stratégie de remplacement des travailleurs pour contrôler la manière dont Amazon MWWA gère les travailleurs actifs lors d'une mise à jour de l'environnement. Vous pouvez sélectionner l'une des stratégies suivantes :

Mises à jour forcées

La mise à jour forcée est la stratégie de remplacement des travailleurs par défaut. Les mises à jour forcées arrêtent immédiatement tous les travailleurs actifs, ce qui entraîne l'échec des tâches en cours d'exécution pendant la mise à jour.

Mises à jour gracieuses

Les mises à jour gracieuses permettent aux travailleurs de continuer à exécuter leurs tâches jusqu'à 12 heures avant de s'arrêter. Il empêche les tâches d'échouer en raison d'interruptions de mise à jour, à condition qu'elles se terminent en moins de 12 heures. Les nouvelles tâches sont transmises aux travailleurs mis à jour.

Pour activer les mises à jour Graceful sur un environnement existant, vous devez effectuer une mise à jour forcée et vous assurer que l'environnement fonctionne sur Apache Airflow version 2.4.3 ou supérieure.

 Note

Si vous effectuez une mise à jour alors que l'ÉTAT de votre environnement est en cours, la stratégie de remplacement du personnel pour toute mise à jour continue de l'environnement passe de GRACEFUL à FORCED. Votre mise à jour est effectuée une fois la maintenance terminée.

Mettre à jour les ressources environnementales

Les mises à jour de l'environnement Amazon MWAA utilisent la configuration d'environnement existante par défaut. Pour mettre à jour l'environnement sans modifier votre configuration actuelle :

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Dans la liste Environnements, choisissez l'environnement que vous souhaitez mettre à jour.
3. Sur la page de l'environnement, choisissez Modifier pour modifier l'environnement.
4. Choisissez Suivant jusqu'à ce que vous soyez sur la page Réviser et enregistrer.
5. Sur la page Réviser et enregistrer, passez en revue vos modifications, puis choisissez Enregistrer.

Mise à jour d'un environnement

La section suivante décrit les étapes de mise à jour d'un environnement Amazon MWAA.

Première étape : Spécifier les détails

Pour spécifier les détails de l'environnement

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Dans la liste Environnements, choisissez l'environnement que vous souhaitez mettre à jour.
3. Sur la page de l'environnement, choisissez Modifier pour modifier l'environnement.
4. Dans la section Détails de l'environnement, pour la version Airflow, choisissez le nouveau numéro de version d'Apache Airflow vers lequel vous souhaitez mettre à niveau l'environnement dans la liste déroulante.

 Note

Avant de procéder à la mise à niveau, assurez-vous que vos ressources de flux de travail DAGs et les autres sont compatibles avec la nouvelle version d'Apache Airflow. Pour plus d'informations, consultez [Modification de la version d'Apache Airflow](#).

5. Sous le code DAG dans Amazon S3, spécifiez les éléments suivants :
 - a. Seau S3. Choisissez Browse S3 et sélectionnez votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - b. DAGs folder. Choisissez Browse S3 et sélectionnez le dags dossier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - c. Fichier de plugins : facultatif. Choisissez Browse S3 et sélectionnez le `plugins.zip` fichier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - d. Fichier des exigences : facultatif. Choisissez Browse S3 et sélectionnez le `requirements.txt` fichier dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
 - e. Fichier de script de démarrage : facultatif. Choisissez Browse S3 et sélectionnez le fichier de script dans votre compartiment Amazon S3, ou entrez l'URI Amazon S3.
6. Choisissez Suivant.

Deuxième étape : configurer les paramètres avancés

Pour configurer les paramètres avancés

1. Sous Accès au serveur Web, sélectionnez votre mode d'accès [Apache Airflow](#) préféré :
 - a. Réseau privé. Cela limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement. Pour cette étape, vous devez disposer d'une autorisation pour créer des points de terminaison Amazon VPC.

 Note

Choisissez l'option Réseau privé si l'interface utilisateur d'Apache Airflow n'est accessible qu'au sein d'un réseau d'entreprise et si vous n'avez pas besoin

d'accéder aux référentiels publics pour l'installation des exigences du serveur Web. Si vous choisissez cette option de mode d'accès, vous devez créer un mécanisme pour accéder à votre serveur Web Apache Airflow dans votre Amazon VPC. Pour plus d'informations, consultez [Accès au point de terminaison VPC de votre serveur Web Apache Airflow \(accès réseau privé\)](#).

- b. Réseau public. Cela permet aux utilisateurs autorisés à accéder à l'interface utilisateur d'Apache Airflow via Internet à la [politique IAM de](#) votre environnement.
2. Sous Groupes de sécurité, choisissez le groupe de sécurité utilisé pour sécuriser votre [Amazon VPC](#) :
 - a. Par défaut, Amazon MWAA crée un groupe de sécurité dans votre Amazon VPC avec des règles d'entrée et de sortie spécifiques dans Créer un nouveau groupe de sécurité.
 - b. Facultatif. Décochez la case dans Créer un nouveau groupe de sécurité pour sélectionner jusqu'à 5 groupes de sécurité.

 Note

Un groupe de sécurité Amazon VPC existant doit être configuré avec des règles entrantes et sortantes spécifiques pour autoriser le trafic réseau. Pour en savoir plus, reportez-vous à [Sécurité de votre VPC sur Amazon MWAA](#).

3. Sous Classe d'environnement, choisissez une [classe d'environnement](#).

Nous vous recommandons de choisir la plus petite taille nécessaire pour supporter votre charge de travail. Vous pouvez modifier la classe d'environnement à tout moment.

4. Pour le nombre maximal de travailleurs, spécifiez le nombre maximum de travailleurs Apache Airflow à exécuter dans l'environnement.

Pour plus d'informations, consultez [Exemple de cas d'utilisation à hautes performances](#).

5. Spécifiez le nombre maximal de serveurs Web et le nombre minimal de serveurs Web pour configurer la manière dont Amazon MWAA adapte les serveurs Web Apache Airflow dans votre environnement.

Pour plus d'informations sur le dimensionnement automatique du serveur Web, reportez-vous à [the section called "Configuration de la mise à l'échelle automatique du serveur Web"](#).

6. Sous Chiffrement, choisissez une option de chiffrement des données :

- a. Par défaut, Amazon MWAA utilise une clé AWS appartenant à Amazon pour chiffrer vos données.
- b. Facultatif. Choisissez Personnaliser les paramètres de chiffrement (avancés) pour choisir une autre AWS KMS clé. Si vous choisissez de spécifier une [clé gérée par le client](#) à cette étape, vous devez spécifier un ID de AWS KMS clé ou un ARN. [AWS KMS les alias et les clés multirégionales ne sont pas pris en charge par Amazon MWAA](#). Si vous avez spécifié une clé Amazon S3 pour le chiffrement côté serveur sur votre compartiment Amazon S3, vous devez spécifier la même clé pour votre environnement Amazon MWAA.

 Note

Vous devez être autorisé à accéder à la clé pour la sélectionner sur la console Amazon MWAA. Vous devez également autoriser Amazon MWAA à utiliser la clé en joignant la politique décrite dans [Joindre une politique clé](#).

7. Recommandé Sous Surveillance, choisissez une ou plusieurs catégories de journaux pour la configuration de journalisation Airflow afin d'envoyer les journaux Apache Airflow à Logs CloudWatch :
- a. Journaux des tâches Airflow. Choisissez le type de journaux de tâches Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- b. Journaux du serveur Web Airflow. Choisissez le type de journaux du serveur Web Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- c. Journaux du planificateur de débit d'air. Choisissez le type de journaux du planificateur Apache Airflow à envoyer au niveau Logs in CloudWatch Log.
- d. Journaux des travailleurs de Airflow. Choisissez le type de journaux de travail d'Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
- e. Journaux de traitement Airflow DAG. Choisissez le type de journaux de traitement du DAG Apache Airflow à envoyer au niveau CloudWatch Logs in Log.
8. Facultatif. Pour les options de configuration Airflow, choisissez l'option Ajouter une configuration personnalisée.

Vous pouvez choisir parmi la liste déroulante suggérée des options de [configuration d'Apache Airflow](#) pour votre version d'Apache Airflow, ou spécifier des options de configuration personnalisées. Par exemple, `core.default_task_retries : 3`.

9. Sous Autorisations, choisissez un rôle d'exécution :

- a. Par défaut, Amazon MWAA crée un [rôle d'exécution](#) dans Create a new role. Vous devez être autorisé à créer des rôles IAM pour utiliser cette option.
 - b. Facultatif. Choisissez Enter role ARN pour saisir le Amazon Resource Name (ARN) d'un rôle d'exécution existant.
10. Sous Spécifications de mise à jour, choisissez un [Stratégie de remplacement des travailleurs](#) pour contrôler la manière dont les travailleurs actifs sont traités lors d'une mise à jour.
 11. Choisissez Suivant.

Troisième étape : révision et mise à jour

Pour consulter un résumé de l'environnement

- Consultez le résumé de l'environnement, puis choisissez Enregistrer.

Note

Il faut environ vingt à trente minutes pour mettre à jour un environnement à l'aide de mises à jour forcées. Les mises à jour progressives de l'environnement peuvent prendre jusqu'à douze heures, car elles attendent la fin de vos tâches en cours.

Modification de la version d'Apache Airflow

Amazon MWAA prend en charge les mises à niveau et les rétrogradations de versions mineures. Cela signifie que vous pouvez mettre à jour votre environnement de version `x.4.z` à `x.5.z` ou de version `x.5.z` à `x.4.z`. Pour effectuer une mise à niveau de version majeure, par exemple de la version `1.y.z` vers `2.y.z`, vous devez créer un nouvel environnement et migrer vos ressources. Pour plus d'informations sur la mise à niveau vers une nouvelle version majeure d'Apache Airflow, reportez-vous à la section [Migration vers un nouvel environnement Amazon MWAA dans le guide](#) de migration Amazon MWAA.

Au cours du processus de mise à niveau ou de rétrogradation, Amazon MWAA capture un instantané des métadonnées de votre environnement, met à niveau ou rétrograde les travailleurs, les planificateurs et le serveur Web vers la nouvelle version d'Apache Airflow, puis restaure la base de données de métadonnées à l'aide de l'instantané.

Avant de procéder à une mise à niveau ou à une rétrogradation, assurez-vous que vos ressources de flux de travail DAGs et les autres ressources sont compatibles avec la nouvelle version d'Apache Airflow vers laquelle vous effectuez la mise à niveau. Si vous utilisez `requirements.txt` pour gérer les dépendances, vous devez également vous assurer que les dépendances que vous spécifiez dans vos exigences sont compatibles avec la nouvelle version.

Rubriques

- [Améliorez ou rétrogradez les ressources de votre flux de travail](#)
- [Spécifiez la nouvelle version](#)

Améliorez ou rétrogradez les ressources de votre flux de travail

Chaque fois que vous modifiez la version d'Apache Airflow, assurez-vous de [référer l'--constraintURL correcte](#) dans votre `requirements.txt`

Warning

La spécification d'exigences incompatibles avec votre version cible d'Apache Airflow lors d'une mise à niveau ou d'une rétrogradation peut entraîner un long processus de restauration vers la version précédente d'Apache Airflow avec la version précédente des exigences.

Migrez les ressources de votre flux de travail

1. Créez un fork du [aws-mwaa-docker-images](#) référentiel et clonez une copie du runner local Amazon MWAA.
2. Accédez à la branche du `aws-mwaa-docker-images` référentiel correspondant à la version vers laquelle vous effectuez une mise à niveau ou une rétrogradation.
3. Pour mettre à jour votre `requirements.txt` compte, suivez les meilleures pratiques que nous recommandons dans [la section Gestion des dépendances Python](#), dans le guide de l'utilisateur Amazon MWAA.
4. (Facultatif) Pour accélérer le processus de mise à niveau ou de rétrogradation, [nettoyez la base de données de métadonnées de l'environnement](#). La mise à niveau des environnements contenant une grande quantité de métadonnées peut prendre beaucoup plus de temps.

5. Une fois que vous avez testé avec succès les ressources de votre flux de travail `DAGsrequirements.txt`, copiez vos plugins et vos plugins dans le compartiment Amazon S3 de votre environnement.

Vous êtes maintenant prêt à modifier l'environnement, à spécifier une nouvelle version d'Apache Airflow et à démarrer la procédure de mise à jour.

Spécifiez la nouvelle version

Après avoir mis à jour les ressources de votre flux de travail pour garantir la compatibilité avec la nouvelle version d'Apache Airflow, procédez comme suit pour modifier les détails de l'environnement et spécifier la version d'Apache Airflow vers laquelle vous souhaitez effectuer la mise à niveau.

Note

Lorsque vous effectuez une mise à niveau ou une rétrogradation, toutes les tâches en cours d'exécution sur l'environnement sont interrompues au cours de la procédure. La procédure de mise à jour peut prendre jusqu'à deux heures, période pendant laquelle votre environnement ne sera pas disponible.

Spécifiez une nouvelle version à l'aide de la console

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Dans la liste des environnements, choisissez l'environnement que vous souhaitez mettre à niveau ou rétrograder.
3. Sur la page de l'environnement, choisissez Modifier pour modifier l'environnement.
4. Dans la section Détails de l'environnement, pour la version Airflow, choisissez le numéro de version d'Apache Airflow vers lequel vous souhaitez mettre à niveau ou rétrograder l'environnement dans la liste déroulante.
5. Choisissez Suivant jusqu'à ce que vous soyez sur la page Réviser et enregistrer.
6. Sur la page Réviser et enregistrer, passez en revue vos modifications, puis choisissez Enregistrer.

Lorsque vous appliquez des modifications, votre environnement lance la procédure de mise à niveau ou de rétrogradation. Au cours de cette période, l'[état](#) de votre environnement indique les actions entreprises par Amazon MWAA et indique si la procédure est réussie.

Dans un scénario de mise à niveau ou de rétrogradation réussi, le statut sera UPDATING alors CREATING_SNAPSHOT tel qu'Amazon MWAA capture une sauvegarde de vos métadonnées. Enfin, le statut reviendra d'abord àUPDATING, puis à une AVAILABLE fois la procédure terminée.

Si l'environnement ne parvient pas à passer à une mise à niveau ou à une rétrogradation, l'état de votre environnement sera le même. ROLLING_BACK Si la restauration est réussie, le statut sera d'abord affichéUPDATE_FAILED, indiquant que la mise à jour a échoué mais que l'environnement est disponible. Si la restauration échoue, le statut sera UNAVAILABLE le suivant : vous ne pouvez pas accéder à l'environnement.

Utilisation d'un script de démarrage avec Amazon MWAA

Un script de démarrage est un script shell (.sh) que vous hébergez dans le compartiment Amazon S3 de votre environnement DAGs, similaire à vos exigences et à vos plugins. Amazon MWAA exécute ce script au démarrage sur chaque composant Apache Airflow individuel (programme de travail, planificateur et serveur Web) avant d'installer les exigences et d'initialiser le processus Apache Airflow. Utilisez un script de démarrage pour effectuer les opérations suivantes :

- Installez les environnements d'exécution : installez les environnements d'exécution Linux requis par vos flux de travail et vos connexions.
- Configurer les variables d'environnement : définissez les variables d'environnement pour chaque composant Apache Airflow. Remplacez les variables courantes telles que PATHPYTHONPATH, etLD_LIBRARY_PATH.
- Gestion des clés et des jetons : transmettez des jetons d'accès aux référentiels personnalisés aux clés de sécurité requirements.txt et configurez-les.

Les rubriques suivantes décrivent comment configurer un script de démarrage pour installer des environnements d'exécution Linux, définir des variables d'environnement et résoudre les problèmes connexes à l'aide CloudWatch des journaux.

Rubriques

- [Configuration d'un script de démarrage](#)
- [Installation des environnements d'exécution Linux à l'aide d'un script de démarrage](#)

- [Définissez des variables d'environnement à l'aide d'un script de démarrage](#)

Configuration d'un script de démarrage

Pour utiliser un script de démarrage avec votre environnement Amazon MWAA existant, chargez un `.sh` fichier dans le compartiment Amazon S3 de votre environnement. Ensuite, pour associer le script à l'environnement, spécifiez ce qui suit dans les détails de votre environnement :

- Le chemin de l'URL Amazon S3 vers le script : le chemin relatif vers le script hébergé dans votre compartiment, par exemple, `s3://mwaa-environment/startup.sh`
- L'ID de version Amazon S3 du script : version du script shell de démarrage dans votre compartiment Amazon S3. Vous devez spécifier l'[ID de version](#) qu'Amazon S3 attribue au fichier chaque fois que vous mettez à jour le script. Les versions sont des chaînes opaques Unicode, codées en UTF-8, prêtes pour les URL, d'une longueur maximale de 1 024 octets, par exemple. `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`

Pour effectuer les étapes décrites dans cette section, utilisez l'exemple de script suivant. Le script affiche la valeur attribuée à `MWAA_AIRFLOW_COMPONENT`. Cette variable d'environnement identifie chaque composant Apache Airflow sur lequel le script s'exécute.

Copiez le code et enregistrez-le localement sous le nom `startup.sh`.

```
#!/bin/sh

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

Ensuite, téléchargez le script dans votre compartiment Amazon S3.

AWS Management Console

Pour télécharger un script shell (console)

1. Connectez-vous à la console Amazon S3 AWS Management Console et ouvrez-la à l'adresse <https://console.aws.amazon.com/s3/>.
2. Dans la liste des compartiments, choisissez le nom du compartiment associé à votre environnement.
3. Dans l'onglet Objets, choisissez Upload (Charger).

4. Sur la page de téléchargement, glissez et déposez le script shell que vous avez créé.
5. Choisissez Charger.

Le script est répertorié dans la liste des objets. Amazon S3 crée un nouvel ID de version pour le fichier. Si vous mettez à jour le script et que vous le chargez à nouveau en utilisant le même nom de fichier, un nouvel ID de version est attribué au fichier.

AWS CLI

Pour créer et télécharger un script shell (CLI)

1. Ouvrez une nouvelle invite de commande et exécutez la `ls` commande Amazon S3 pour répertorier et identifier le compartiment associé à votre environnement.

```
aws s3 ls
```

2. Accédez au dossier dans lequel vous avez enregistré le script shell. Utilisez-le dans une nouvelle fenêtre d'invite pour télécharger le script dans votre compartiment. Remplacez *amzn-s3-demo-bucket* par vos informations.

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

En cas de succès, Amazon S3 affiche le chemin URL de l'objet :

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

3. Utilisez la commande suivante pour récupérer le dernier ID de version du script.

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

```
BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Vous spécifiez cet ID de version lorsque vous associez le script à un environnement.

Associez maintenant le script à votre environnement.

AWS Management Console

Pour associer le script à un environnement (console)

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Sélectionnez la ligne correspondant à l'environnement que vous souhaitez mettre à jour, puis choisissez Modifier.
3. Sur la page Spécifier les détails, pour Fichier de script de démarrage - facultatif, entrez l'URL Amazon S3 du script, par exemple :s3://*amzn-s3-demo-bucket*/startup-sh..
4. Choisissez la dernière version dans la liste déroulante ou naviguez S3 pour trouver le script.
5. Choisissez Suivant, puis passez à la page Réviser et enregistrer.
6. Passez en revue les modifications, puis choisissez Enregistrer.

Les mises à jour de l'environnement peuvent prendre entre 10 et 30 minutes. Amazon MWAA exécute le script de démarrage lorsque chaque composant de votre environnement redémarre.

AWS CLI

Pour associer le script à un environnement (CLI)

- Ouvrez une invite de commande et `update-environment` utilisez-la pour spécifier l'URL Amazon S3 et l'ID de version du script.

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

En cas de succès, Amazon MWAA renvoie le nom de ressource Amazon (ARN) pour l'environnement :

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

La mise à jour de l'environnement peut prendre entre 10 et 30 minutes. Amazon MWAA exécute le script de démarrage lorsque chaque composant de votre environnement redémarre.

Enfin, récupérez les événements du journal pour vérifier que le script fonctionne comme prévu. Lorsque vous activez la journalisation pour chaque composant Apache Airflow, Amazon MWAA crée un nouveau groupe de journaux et un nouveau flux de journaux. Pour plus d'informations, reportez-vous à la section Types de [journaux Apache Airflow](#).

AWS Management Console

Pour obtenir le flux de log d'Apache Airflow (console)

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez votre environnement.
3. Dans le volet Surveillance, choisissez le groupe de journaux pour lequel vous souhaitez accéder aux journaux, par exemple le groupe de journaux du planificateur Airflow.
4. Dans la CloudWatch console, dans la liste Log streams, choisissez un flux avec le préfixe suivant :`startup_script_execution_ip`.
5. Dans le volet Journaliser les événements, vous allez faire référence à la sortie de la commande qui affiche la valeur pour `MWAA_AIRFLOW_COMPONENT`. Par exemple, pour les journaux du planificateur, vous obtiendrez ce qui suit :

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

Vous pouvez répéter les étapes précédentes pour accéder aux journaux des utilisateurs et du serveur Web.

Installation des environnements d'exécution Linux à l'aide d'un script de démarrage

Utilisez un script de démarrage pour mettre à jour le système d'exploitation d'un composant Apache Airflow et installez des bibliothèques d'exécution supplémentaires à utiliser avec vos flux de travail. Par exemple, le script suivant s'exécute `yum update` pour mettre à jour le système d'exploitation.

Lorsque vous exécutez `yum update` un script de démarrage, vous devez exclure Python --`exclude=python*` comme indiqué dans l'exemple. Pour que votre environnement fonctionne,

Amazon MWAA installe une version spécifique de Python compatible avec votre environnement. Par conséquent, vous ne pouvez pas mettre à jour la version Python de l'environnement à l'aide d'un script de démarrage.

```
#!/bin/sh

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

Pour installer des environnements d'exécution sur un composant Apache Airflow spécifique, MWAA_AIRFLOW_COMPONENT utilisez les instructions `if fi` et conditionnelles. Cet exemple exécute une seule commande pour installer la `libaio` bibliothèque sur le planificateur et le programme de travail, mais pas sur le serveur Web.

 Important

- Si vous avez configuré un [serveur Web privé](#), vous devez soit utiliser la condition suivante, soit fournir tous les fichiers d'installation localement pour éviter les délais d'installation.
- `sudo` À utiliser pour exécuter des opérations nécessitant des privilèges administratifs.

```
#!/bin/sh

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

Vous pouvez utiliser un script de démarrage pour obtenir la version de Python.

```
#!/bin/sh

export PYTHON_VERSION_CHECK=`python -c 'import sys; version=sys.version_info[:3];
print("{0}.{1}.{2}".format(*version))'`
echo "Python version is $PYTHON_VERSION_CHECK"
```

Amazon MWAA ne prend pas en charge le remplacement de la version par défaut de Python, car cela pourrait entraîner des incompatibilités avec les bibliothèques Apache Airflow installées.

Définissez des variables d'environnement à l'aide d'un script de démarrage

Utilisez des scripts de démarrage pour définir des variables d'environnement et modifier les configurations d'Apache Airflow. Ce qui suit définit une nouvelle variable, `ENVIRONMENT_STAGE`. Vous pouvez référencer cette variable dans un DAG ou dans vos modules personnalisés.

```
#!/bin/sh

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

Utilisez des scripts de démarrage pour remplacer les variables système ou Apache Airflow courantes. Par exemple, vous devez indiquer `LD_LIBRARY_PATH` à Python de rechercher des fichiers binaires dans le chemin que vous spécifiez. Cela vous permet de fournir des fichiers binaires personnalisés pour vos flux de travail à l'aide de [plugins](#) :

```
#!/bin/sh

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

Variables d'environnement réservées

Amazon MWAA réserve un ensemble de variables d'environnement critiques. Si vous remplacez une variable réservée, Amazon MWAA rétablit sa valeur par défaut. La liste suivante répertorie les variables réservées :

- `MWAA__AIRFLOW__COMPONENT`— Utilisé pour identifier le composant Apache Airflow avec l'une des valeurs suivantes : `schedulerworker`, ou `webserver`
- `AIRFLOW__WEBSERVER__SECRET_KEY`— La clé secrète utilisée pour signer en toute sécurité les cookies de session sur le serveur Web Apache Airflow.
- `AIRFLOW__CORE__FERNET_KEY`— La clé utilisée pour le chiffrement et le déchiffrement des données sensibles stockées dans la base de métadonnées, par exemple les mots de passe de connexion.
- `AIRFLOW_HOME`— Le chemin d'accès au répertoire de base d'Apache Airflow où les fichiers de configuration et les fichiers DAG sont stockés localement.
- `AIRFLOW__CELERY__BROKER_URL`— URL du courtier de messages utilisé pour la communication entre le planificateur Apache Airflow et les nœuds de travail Celery.

- `AIRFLOW__CELERY__RESULT_BACKEND`— URL de la base de données utilisée pour stocker les résultats des tâches Celery.
- `AIRFLOW__CORE__EXECUTOR`— La classe d'exécuteur qu'Apache Airflow doit utiliser. Dans Amazon MWAA, il s'agit d'un `CeleryExecutor`.
- `AIRFLOW__CORE__LOAD_EXAMPLES`— Utilisé pour activer ou désactiver le chargement d'un exemple DAGs.
- `AIRFLOW__METRICS__METRICS_BLOCK_LIST`— Utilisé pour gérer les métriques Apache Airflow émises et capturées par Amazon MWAA dans CloudWatch.
- `SQL_ALCHEMY_CONN`— Chaîne de connexion pour la base de données RDS pour PostgreSQL utilisée pour stocker les métadonnées Apache Airflow dans Amazon MWAA.
- `AIRFLOW__CORE__SQL_ALCHEMY_CONN`— Utilisé dans le même but que `SQL_ALCHEMY_CONN`, mais conformément à la nouvelle convention de dénomination d'Apache Airflow.
- `AIRFLOW__CELERY__DEFAULT_QUEUE`— La file d'attente par défaut pour les tâches Celery dans Apache Airflow.
- `AIRFLOW__OPERATORS__DEFAULT_QUEUE`— La file d'attente par défaut pour les tâches utilisant des opérateurs Apache Airflow spécifiques.
- `AIRFLOW_VERSION`— La version d'Apache Airflow installée dans l'environnement Amazon MWAA.
- `AIRFLOW_CONN_AWS_DEFAULT`— Les AWS informations d'identification par défaut utilisées pour l'intégration à d'autres AWS services dans.
- `AWS_DEFAULT_REGION`— Définit la valeur par défaut Région AWS utilisée avec les informations d'identification par défaut pour l'intégration à d'autres AWS services.
- `AWS_REGION`— Si elle est définie, cette variable d'environnement remplace les valeurs de la variable d'environnement `AWS_DEFAULT_REGION` et de la région de définition du profil.
- `PYTHONUNBUFFERED`— Utilisé pour envoyer `stdout` et diffuser `stderr` des journaux de conteneurs.
- `AIRFLOW__METRICS__STATSD_ALLOW_LIST`— Utilisé pour configurer une liste d'autorisation de préfixes séparés par des virgules afin d'envoyer les métriques commençant par les éléments de la liste.
- `AIRFLOW__METRICS__STATSD_ON`— Active l'envoi de métriques à `StatsD`.
- `AIRFLOW__METRICS__STATSD_HOST`— Utilisé pour se connecter au `StatsD` daemon.
- `AIRFLOW__METRICS__STATSD_PORT`— Utilisé pour se connecter au `StatsD` daemon.
- `AIRFLOW__METRICS__STATSD_PREFIX`— Utilisé pour se connecter au `StatsD` daemon.
- `AIRFLOW__CELERY__WORKER_AUTOSCALE`— Définit le maximum et le minimum de simultanéité.

- `AIRFLOW__CORE__DAG_CONCURRENCY`— Définit le nombre d'instances de tâches pouvant être exécutées simultanément par le planificateur dans un DAG.
- `AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG`— Définit le nombre maximum de tâches actives par DAG.
- `AIRFLOW__CORE__PARALLELISM`— Définit le nombre maximum d'instances de tâches pouvant être exécutées simultanément.
- `AIRFLOW__SCHEDULER__PARSING_PROCESSES`— Définit le nombre maximum de processus analysés par le planificateur à planifier. DAGs
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT`— Définit le nombre de secondes pendant lesquelles un collaborateur attend pour accuser réception de la tâche avant que le message ne soit remis à un autre collaborateur.
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION`— Définit Région AWS le transport Celery sous-jacent.
- `AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES`— Définit la file d'attente pour le transport Celery sous-jacent.
- `AIRFLOW__SCHEDULER__ALLOWED_RUN_ID_PATTERN`— Utilisé pour vérifier la validité de votre entrée pour le `run_id` paramètre lors du déclenchement d'un DAG.
- `AIRFLOW__WEBSERVER__BASE_URL`— URL du serveur Web utilisé pour héberger l'interface utilisateur d'Apache Airflow.
- `PYTHONPATH`(uniquement pour Apache Airflow v2.9 et versions ultérieures) — Réserve par Amazon MWAA pour garantir le bon fonctionnement de toutes les fonctionnalités de base de l'environnement.

Note

Pour les versions d'Apache Airflow antérieures à la version 2.9, il `PYTHONPATH` s'agit d'une variable d'environnement non réservée.

Variables d'environnement non réservées

Vous pouvez utiliser un script de démarrage pour remplacer les variables d'environnement non réservées. Voici une liste de certaines de ces variables courantes :

- `PATH`— Spécifie une liste de répertoires dans lesquels le système d'exploitation recherche les fichiers exécutables et les scripts. Lorsqu'une commande est exécutée dans la ligne de

commande, le système consulte les répertoires PATH pour rechercher et exécuter la commande. Lorsque vous créez des opérateurs ou des tâches personnalisés dans Apache Airflow, vous devrez peut-être vous appuyer sur des scripts ou des exécutables externes. Si les répertoires contenant ces fichiers ne se trouvent pas dans les répertoires spécifiés dans la PATH variable, les tâches ne s'exécutent pas lorsque le système ne parvient pas à les localiser. En ajoutant les répertoires appropriés PATH, les tâches Apache Airflow peuvent trouver et exécuter les exécutables requis.

- **PYTHONPATH**— Utilisé par l'interpréteur Python pour déterminer dans quels répertoires rechercher les modules et packages importés. Il s'agit d'une liste de répertoires que vous pouvez ajouter au chemin de recherche par défaut. Cela permet à l'interpréteur de rechercher et de charger des bibliothèques Python qui ne sont pas incluses dans la bibliothèque standard ou qui ne sont pas installées dans les répertoires du système. Utilisez cette variable pour ajouter vos modules et packages Python personnalisés et les utiliser avec votre DAGs.

 Note

Pour Apache Airflow v2.9 et versions ultérieures, il PYTHONPATH s'agit d'une variable d'environnement réservée.

- **LD_LIBRARY_PATH**— Variable d'environnement utilisée par l'éditeur de liens et le chargeur dynamiques sous Linux pour rechercher et charger des bibliothèques partagées. Il spécifie une liste de répertoires contenant des bibliothèques partagées, qui sont recherchés avant les répertoires des bibliothèques système par défaut. Utilisez cette variable pour spécifier vos fichiers binaires personnalisés.
- **CLASSPATH**— Utilisé par l'environnement d'exécution Java (JRE) et le kit de développement Java (JDK) pour localiser et charger des classes, des bibliothèques et des ressources Java lors de l'exécution. Il s'agit d'une liste de répertoires, de fichiers JAR et d'archives ZIP contenant du code Java compilé.

Travailler avec DAGs sur Amazon MWAA

Pour exécuter des graphes acycliques dirigés (DAGs) dans un environnement Amazon Managed Workflows for Apache Airflow, vous copiez vos fichiers dans le compartiment de stockage Amazon S3 attaché à votre environnement, puis vous indiquez à Amazon MWAA où se trouvent DAGs vos fichiers et les fichiers annexes sur la console Amazon MWAA. Amazon MWAA se charge de la synchronisation DAGs entre les travailleurs, les planificateurs et le serveur Web. Ce guide explique comment ajouter ou mettre à jour vos plugins personnalisés DAGs et vos dépendances Python et comment les installer dans un environnement Amazon MWAA.

Rubriques

- [Présentation du bucket Amazon S3](#)
- [Ajouter ou mettre à jour DAGs](#)
- [Installation de plugins personnalisés](#)
- [Installation des dépendances Python](#)
- [Suppression de fichiers sur Amazon S3](#)

Présentation du bucket Amazon S3

L'accès public d'un compartiment Amazon S3 pour un environnement Amazon MWAA doit être bloqué. Par défaut, toutes les ressources Amazon S3 (compartiments, objets et sous-ressources associées (par exemple, configuration du cycle de vie) sont privées.

- Seul le propriétaire de la ressource, Compte AWS celui qui a créé le compartiment, peut accéder à la ressource. Le propriétaire de la ressource (par exemple, votre administrateur) peut accorder des autorisations d'accès à d'autres personnes en rédigeant une politique de contrôle d'accès.
- La politique d'accès que vous configurez doit être autorisée à ajouter DAGs des plug-ins personnalisés et des dépendances Python `requirements.txt` à votre compartiment Amazon S3. `plugins.zip` Pour un exemple de politique contenant les autorisations requises, consultez [Amazon MWAA Full Console Access](#).

Le contrôle de version d'un compartiment Amazon S3 pour un environnement Amazon MWAA doit être activé. Lorsque le versionnement des compartiments Amazon S3 est activé, chaque fois qu'une nouvelle version est créée, une nouvelle copie est créée.

- La gestion des versions est activée pour les plugins personnalisés dans un `compartimentplugins.zip`, et les dépendances Python dans un `requirements.txt` compartiment Amazon S3.
- Vous devez spécifier la version d'un `plugins.zip` et `requirements.txt` sur la console Amazon MWAA chaque fois que ces fichiers sont mis à jour sur votre compartiment Amazon S3.

Ajouter ou mettre à jour DAGs

Les graphes acycliques dirigés (DAGs) sont définis dans un fichier Python qui définit la structure du DAG sous forme de code. Vous pouvez utiliser la console Amazon S3 AWS CLI ou la console Amazon S3 pour effectuer le téléchargement DAGs dans votre environnement. Cette rubrique décrit les étapes pour ajouter ou mettre à jour Apache Airflow DAGs dans votre environnement Amazon Managed Workflows for Apache Airflow à l'aide `dags` du dossier de votre compartiment Amazon S3.

Sections

- [Prérequis](#)
- [Comment ça marche](#)
- [Qu'est-ce qui a changé ?](#)
- [Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI](#)
- [Téléchargement du code DAG sur Amazon S3](#)
- [Spécifier le chemin d'accès à votre DAGs dossier sur la console Amazon MWAA \(pour la première fois\)](#)
- [Accès aux modifications sur votre interface utilisateur Apache Airflow](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Vous aurez besoin des éléments suivants avant de pouvoir effectuer les étapes indiquées sur cette page.

- Autorisations — Votre administrateur Compte AWS doit vous avoir accordé l'accès à la politique de contrôle MWAAFull ConsoleAccess d'accès [Amazon](#) pour votre environnement. En outre, votre environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder aux AWS ressources utilisées par votre environnement.

- Accès : si vous avez besoin d'accéder à des référentiels publics pour installer des dépendances directement sur le serveur Web, votre environnement doit être configuré avec un accès au serveur Web du réseau public. Pour plus d'informations, consultez [the section called “Modes d'accès Apache Airflow”](#).
- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAGs plugins personnalisés et vos dépendances Python `requirements.txt` doit être configuré avec l'accès public bloqué et le versionnage activé. `plugins.zip`

Comment ça marche

Un graphe acyclique dirigé (DAG) est défini dans un seul fichier Python qui définit la structure du DAG sous forme de code. Il comprend les éléments suivants :

- Une définition du [DAG](#).
- [Opérateurs](#) qui décrivent comment exécuter le DAG et les [tâches](#) à exécuter.
- [Relations entre les opérateurs](#) qui décrivent l'ordre dans lequel les tâches doivent être exécutées.

Pour exécuter une plateforme Apache Airflow dans un environnement Amazon MWAA, vous devez copier la définition de votre DAG dans le dags dossier de votre bucket de stockage. Par exemple, le dossier DAG de votre bucket de stockage doit ressembler à :

Exemple dossier DAG

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

Amazon MWAA synchronise automatiquement les objets nouveaux et modifiés de votre compartiment Amazon S3 avec le planificateur Amazon MWAA et le `/usr/local/airflow/dags` dossier des conteneurs de travail toutes les 30 secondes, préservant ainsi la hiérarchie des fichiers de la source Amazon S3, quel que soit le type de fichier. L'heure à laquelle la nouvelle DAGs prise doit être répertoriée dans votre interface utilisateur d'Apache Airflow est contrôlée par [scheduler.dag_dir_list_interval](#). Les modifications apportées à l'existant DAGs seront prises en compte lors de la prochaine [boucle de traitement DAG](#).

Note

Il n'est pas nécessaire d'inclure le fichier `airflow.cfg` de configuration dans votre dossier DAG. Vous pouvez remplacer les configurations par défaut d'Apache Airflow depuis la console Amazon MWAA. Pour plus d'informations, consultez [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).

Qu'est-ce qui a changé ?

Pour consulter les modifications apportées à une version spécifique d'Apache Airflow, reportez-vous à la page des [notes de version](#).

- [Configurations d'Apache Airflow v3 : référence de configuration](#)
- Informations sur l'interface publique d'Apache Airflow v2 : [Interface publique](#) d'Airflow

Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI

- L'utilitaire d'interface de ligne de commande (CLI) reproduit localement un environnement Amazon Managed Workflows pour Apache Airflow.
- La CLI crée localement une image de conteneur Docker similaire à une image de production Amazon MWAA. Vous pouvez l'utiliser pour exécuter un environnement Apache Airflow local afin de développer et de tester DAGs des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-docker-images](#) sur GitHub.

Téléchargement du code DAG sur Amazon S3

Vous pouvez utiliser la console Amazon S3 ou le AWS Command Line Interface (AWS CLI) pour télécharger le code DAG dans votre compartiment Amazon S3. Les étapes suivantes supposent que vous téléchargez du code (`.py`) dans un dossier nommé `dags` dans votre compartiment Amazon S3.

En utilisant le AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Pour effectuer un téléchargement à l'aide du AWS CLI

1. Utilisez la commande suivante pour répertorier tous vos compartiments Amazon S3.

```
aws s3 ls
```

2. Utilisez la commande suivante pour répertorier les fichiers et les dossiers du compartiment Amazon S3 de votre environnement.

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

3. La commande suivante télécharge un `dag_def.py` fichier dags dans un dossier.

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

Si aucun dossier nommé dags n'existe déjà dans votre compartiment Amazon S3, cette commande crée le dags dossier et télécharge le fichier nommé `dag_def.py` dans le nouveau dossier.

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur Web que vous pouvez utiliser pour créer et gérer les ressources de votre compartiment Amazon S3. Les étapes suivantes supposent que vous avez un DAGs dossier nommé dags.

Pour charger à l'aide de la console Amazon S3

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sélectionnez le lien du compartiment S3 dans le code DAG du volet S3 pour ouvrir votre compartiment de stockage dans la console.
4. Choisissez le dossier dags.
5. Choisissez Charger.
6. Choisissez Ajouter un fichier.

7. Sélectionnez la copie locale de votre fichier `dag_def.py`, puis choisissez Upload.

Spécifier le chemin d'accès à votre DAGs dossier sur la console Amazon MWAA (pour la première fois)

Les étapes suivantes supposent que vous spécifiez le chemin d'accès à un dossier nommé dans votre compartiment Amazon S3 `dag`.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez l'environnement dans lequel vous souhaitez exécuter DAGs.
3. Choisissez Modifier.
4. Dans le code DAG dans le volet Amazon S3, choisissez Browse S3 à côté du champ du dossier DAG.
5. Sélectionnez votre `dag` dossier.
6. Choisissez Choisir.
7. Choisissez Suivant, Mettre à jour l'environnement.

Accès aux modifications sur votre interface utilisateur Apache Airflow

Vous avez besoin d'[Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAA Web Server Access](#) autorisations pour que votre Compte AWS identifiant Gestion des identités et des accès AWS (IAM) puisse accéder à votre interface utilisateur Apache Airflow.

Pour accéder à votre interface utilisateur Apache Airflow

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Open Airflow UI.

Quelle est la prochaine étape ?

Testez vos DAGs plugins personnalisés et vos dépendances Python localement en utilisant [aws-mwaa-docker-images](#) sur GitHub.

Installation de plugins personnalisés

Amazon Managed Workflows pour Apache Airflow prend en charge le gestionnaire de plugins intégré d'Apache Airflow, qui vous permet d'utiliser des opérateurs, des hooks, des capteurs ou des interfaces Apache Airflow personnalisés. Cette page décrit les étapes d'installation des [plugins personnalisés Apache Airflow](#) sur votre environnement Amazon MWAA à l'aide d'un fichier `plugins.zip`

Table des matières

- [Prérequis](#)
- [Comment ça marche](#)
- [Quand utiliser les plugins](#)
- [Vue d'ensemble des plugins personnalisés](#)
 - [Répertoire des plugins personnalisés et limites de taille](#)
- [Exemples de plugins personnalisés](#)
 - [Exemple d'utilisation d'une structure de répertoire plate dans `plugins.zip`](#)
 - [Exemple d'utilisation d'une structure de répertoire imbriquée dans `plugins.zip`](#)
- [Création d'un fichier `plugins.zip`](#)
 - [Première étape : tester les plugins personnalisés à l'aide de l'utilitaire Amazon MWAA CLI](#)
 - [Deuxième étape : créer le fichier `plugins.zip`](#)
- [Téléchargement `plugins.zip` vers Amazon S3](#)
 - [À l'aide du AWS CLI](#)
 - [Utilisation de la console Amazon S3](#)
- [Installation de plugins personnalisés sur votre environnement](#)
 - [Spécifier le chemin d'accès `plugins.zip` sur la console Amazon MWAA \(pour la première fois\)](#)
 - [Spécification de la `plugins.zip` version sur la console Amazon MWAA](#)
- [Exemples de cas d'utilisation pour `plugins.zip`](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Vous aurez besoin des éléments suivants avant de pouvoir effectuer les étapes indiquées sur cette page.

- **Autorisations** — Votre administrateur Compte AWS doit vous avoir accordé l'accès à la politique de contrôle MWAAFull ConsoleAccess d'accès [Amazon](#) pour votre environnement. En outre, votre environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder aux AWS ressources utilisées par votre environnement.
- **Accès** : si vous avez besoin d'accéder à des référentiels publics pour installer des dépendances directement sur le serveur Web, votre environnement doit être configuré avec un accès au serveur Web du réseau public. Pour plus d'informations, consultez [the section called "Modes d'accès Apache Airflow"](#).
- **Configuration Amazon S3** — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAGs plugins personnalisés et vos dépendances Python `requirements.txt` doit être configuré avec l'accès public bloqué et le versionnage activé. `plugins.zip`

Comment ça marche

Pour exécuter des plugins personnalisés sur votre environnement, vous devez effectuer trois opérations :

1. Créez un `plugins.zip` fichier localement.
2. Téléchargez le `plugins.zip` fichier local dans votre compartiment Amazon S3.
3. Spécifiez la version de ce fichier dans le champ Fichier de plugins de la console Amazon MWAA.

Note

Si c'est la première fois que vous chargez un `plugins.zip` fichier dans votre compartiment Amazon S3, vous devez également spécifier le chemin d'accès au fichier sur la console Amazon MWAA. Vous ne devez effectuer cette étape qu'une seule fois.

Quand utiliser les plugins

Les plug-ins ne sont nécessaires que pour étendre l'interface utilisateur d'Apache Airflow, comme indiqué dans la documentation d'[Apache Airflow](#). Les opérateurs personnalisés peuvent être placés directement dans le `/dags` dossier à côté de votre DAG code.

Si vous devez créer vos propres intégrations avec des systèmes externes, placez-les dans le `dags` dossier/ou dans l'un de ses sous-dossiers, mais pas dans le `plugins.zip` dossier. Dans Apache Airflow 2.x, les plugins sont principalement utilisés pour étendre l'interface utilisateur.

De même, aucune autre dépendance ne peut être insérée dans `plugins.zip`. Ils peuvent plutôt être stockés dans un emplacement du `/dags` dossier Amazon S3, où ils seront synchronisés avec chaque conteneur Amazon MWAA avant le démarrage d'Apache Airflow.

Note

Tout fichier du `/dags` dossier ou `plugins.zip` qui ne définit pas explicitement un objet DAG d'Apache Airflow doit être répertorié dans un `.airflowignore` fichier.

Vue d'ensemble des plugins personnalisés

Le gestionnaire de plugins intégré à Apache Airflow peut intégrer des fonctionnalités externes à son cœur en déposant simplement des fichiers dans un `$AIRFLOW_HOME/plugins` dossier. Vous pouvez l'utiliser pour utiliser des opérateurs, des hooks, des capteurs ou des interfaces Apache Airflow personnalisés. La section suivante fournit un exemple de structures de répertoires plates et imbriquées dans un environnement de développement local et les instructions d'importation qui en résultent, qui déterminent la structure de répertoire dans un fichier `plugins.zip`.

Répertoire des plugins personnalisés et limites de taille

Le planificateur Apache Airflow et les opérateurs recherchent des plugins personnalisés lors du démarrage sur le conteneur AWS Fargate géré pour votre environnement à l'adresse. `/usr/local/airflow/plugins/*`

- **Structure du répertoire.** La structure du répertoire (at/`*`) est basée sur le contenu de votre `plugins.zip` fichier. Par exemple, si vous `plugins.zip` conservez le `operators` répertoire en tant que répertoire de niveau principal, le répertoire sera extrait dans votre `/usr/local/airflow/plugins/operators` environnement.
- **Limite de taille.** Nous recommandons un `plugins.zip` fichier de moins de 1 Go. Plus la taille d'un `plugins.zip` fichier est importante, plus le temps de démarrage d'un environnement est long. Bien qu'Amazon MWAA ne limite pas explicitement la taille d'un `plugins.zip` fichier, si les dépendances ne peuvent pas être installées dans les dix minutes, le service Fargate expirera et tentera de rétablir la stabilité de l'environnement.

Note

Pour les environnements utilisant Apache Airflow v2.0.2, Amazon MWAA limite le trafic sortant sur le serveur Web Apache Airflow et n'autorise pas l'installation de plugins ni de dépendances Python directement sur le serveur Web. À partir de la version 2.2.2 d'Apache Airflow, Amazon MWAA peut installer des plugins et des dépendances directement sur le serveur Web.

Exemples de plugins personnalisés

La section suivante utilise un exemple de code du guide de référence Apache Airflow pour expliquer comment structurer votre environnement de développement local.

Exemple d'utilisation d'une structure de répertoire plate dans plugins.zip

Apache Airflow v3

L'exemple suivant présente un `plugins.zip` fichier avec une structure de répertoire plate pour Apache Airflow v3.

Exemple répertoire plat avec **PythonVirtualenvOperator** plugins.zip

L'exemple suivant affiche l'arborescence de niveau principal d'un fichier plugins.zip pour le plugin PythonVirtualenvOperator personnalisé dans. [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#)

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

Exemple plugins/virtual_python_plugin.py

L'exemple suivant affiche le plugin PythonVirtualenvOperator personnalisé.

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

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

```

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

from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

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

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'

```

Apache Airflow v2

L'exemple suivant présente un `plugins.zip` fichier avec une structure de répertoire plate pour Apache Airflow v2.

Exemple répertoire plat avec **PythonVirtualenvOperator** `plugins.zip`

L'exemple suivant affiche l'arborescence de niveau principal d'un fichier `plugins.zip` pour le plugin `PythonVirtualenvOperator` personnalisé dans. [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#)

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

Exemple `plugins/virtual_python_plugin.py`

L'exemple suivant affiche le plugin `PythonVirtualenvOperator` personnalisé.

```
"""
```

```
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'
```

Exemple d'utilisation d'une structure de répertoire imbriquée dans plugins.zip

Apache Airflow v3

L'exemple suivant présente un `plugins.zip` fichier avec des répertoires distincts pour `hooks`, `operators`, et un `sensors` répertoire.

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
```

L'exemple suivant affiche les instructions d'importation dans le DAG ([DAGs dossier](#)) qui utilise les plugins personnalisés.

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]
```

Les exemples suivants présentent chacune des instructions d'importation nécessaires dans les fichiers de plug-in personnalisés.

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
```

Suivez les étapes décrites dans [Tester des plugins personnalisés à l'aide de l'utilitaire Amazon MWAA CLI](#), puis [créez un fichier plugins.zip](#) pour compresser le contenu dans votre plugins répertoire. Par exemple, `cd plugins`.

Apache Airflow v2

L'exemple suivant présente un `plugins.zip` fichier avec des répertoires distincts pour `hooks`, `operators`, et un `sensors` répertoire.

Exemple 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
```

L'exemple suivant affiche les instructions d'importation dans le DAG ([DAGs dossier](#)) qui utilise les plugins personnalisés.

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]
```

Les exemples suivants présentent chacune des instructions d'importation nécessaires dans les fichiers de plug-in personnalisés.

Exemple hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook

class MyHook(BaseHook):

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

Exemple 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
```

Suivez les étapes décrites dans [Tester des plugins personnalisés à l'aide de l'utilitaire Amazon MWAA CLI](#), puis [créez un fichier plugins.zip](#) pour compresser le contenu dans votre plugins répertoire. Par exemple, `cd plugins`.

Création d'un fichier plugins.zip

Les étapes suivantes décrivent les étapes que nous recommandons pour créer un fichier plugins.zip localement.

Première étape : tester les plugins personnalisés à l'aide de l'utilitaire Amazon MWAA CLI

- L'utilitaire d'interface de ligne de commande (CLI) reproduit localement un environnement Amazon Managed Workflows pour Apache Airflow.
- La CLI crée localement une image de conteneur Docker similaire à une image de production Amazon MWAA. Vous pouvez l'utiliser pour exécuter un environnement Apache Airflow local afin de développer et de tester DAGs des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-docker-images](#) sur GitHub.

Deuxième étape : créer le fichier plugins.zip

Vous pouvez utiliser un utilitaire d'archivage ZIP intégré ou tout autre utilitaire ZIP (tel que [7zip](#)) pour créer un fichier .zip.

Note

L'utilitaire zip intégré pour le système d'exploitation Windows peut ajouter des sous-dossiers lorsque vous créez un fichier .zip. Nous vous recommandons de vérifier le contenu du fichier plugins.zip avant de le télécharger dans votre compartiment Amazon S3 afin de vous assurer qu'aucun répertoire supplémentaire n'a été ajouté.

1. Remplacez les répertoires par votre répertoire de plugins Airflow local. Par exemple :

```
myproject$ cd plugins
```


2. Exécutez la commande suivante pour vous assurer que le contenu dispose d'autorisations exécutables (macOS et Linux uniquement).

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

3. Compressez le contenu de votre plugins dossier.

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

Téléchargement **plugins.zip** vers Amazon S3

Vous pouvez utiliser la console Amazon S3 ou le AWS Command Line Interface (AWS CLI) pour charger un `plugins.zip` fichier dans votre compartiment Amazon S3.

À l'aide du AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Pour effectuer un téléchargement à l'aide du AWS CLI

1. Dans votre invite de commande, accédez au répertoire dans lequel votre `plugins.zip` fichier est stocké. Par exemple :

```
cd plugins
```

2. Utilisez la commande suivante pour répertorier tous vos compartiments Amazon S3.

```
aws s3 ls
```

3. Utilisez la commande suivante pour répertorier les fichiers et les dossiers du compartiment Amazon S3 de votre environnement.

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

4. Utilisez la commande suivante pour charger le `plugins.zip` fichier dans le compartiment Amazon S3 de votre environnement.

```
aws s3 cp plugins.zip s3://amzn-s3-demo-bucket/plugins.zip
```

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur Web que vous pouvez utiliser pour créer et gérer les ressources de votre compartiment Amazon S3.

Pour charger à l'aide de la console Amazon S3

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sélectionnez le lien du compartiment S3 dans le code DAG du volet S3 pour ouvrir votre compartiment de stockage dans la console.
4. Choisissez Charger.
5. Choisissez Ajouter un fichier.
6. Sélectionnez la copie locale de votre fichier `plugins.zip`, puis choisissez Upload.

Installation de plugins personnalisés sur votre environnement

Cette section décrit comment installer les plugins personnalisés que vous avez chargés dans votre compartiment Amazon S3 en spécifiant le chemin d'accès au fichier `plugins.zip` et en spécifiant la version du fichier `plugins.zip` chaque fois que le fichier zip est mis à jour.

Spécifier le chemin d'accès **plugins.zip** sur la console Amazon MWAA (pour la première fois)

Si c'est la première fois que vous chargez un `plugins.zip` fichier dans votre compartiment Amazon S3, vous devez également spécifier le chemin d'accès au fichier sur la console Amazon MWAA. Vous ne devez effectuer cette étape qu'une seule fois.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.

4. Dans le code DAG du volet Amazon S3, choisissez Browse S3 adjacent au fichier Plugins - champ facultatif.
5. Sélectionnez le `plugins.zip` fichier dans votre compartiment Amazon S3.
6. Choisissez Choisir.
7. Choisissez Suivant, Mettre à jour l'environnement.

Spécification de la **plugins.zip** version sur la console Amazon MWAA

Vous devez spécifier la version de votre `plugins.zip` fichier sur la console Amazon MWAA chaque fois que vous chargez une nouvelle version de votre fichier `plugins.zip` dans votre compartiment Amazon S3.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Sur le code DAG dans le volet Amazon S3, choisissez une `plugins.zip` version dans la liste déroulante.
5. Choisissez Suivant.

Exemples de cas d'utilisation pour plugins.zip

- Découvrez comment créer un plugin personnalisé dans [Plugin personnalisé avec Apache Hive et Hadoop](#).
- Découvrez comment créer un plugin personnalisé dans [Plugin personnalisé à patcher PythonVirtualenvOperator](#).
- Découvrez comment créer un plugin personnalisé dans [Plug-in personnalisé avec Oracle](#).
- Découvrez comment créer un plugin personnalisé dans [the section called “Modifier le fuseau horaire d'un DAG”](#).

Quelle est la prochaine étape ?

Testez vos DAGs plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-docker-images](#) on GitHub.

Installation des dépendances Python

Une dépendance Python est un package ou une distribution non inclus dans l'installation de base d'Apache Airflow pour votre version d'Apache Airflow sur votre environnement Amazon Managed Workflows for Apache Airflow. Cette rubrique décrit les étapes à suivre pour installer les dépendances Python d'Apache Airflow sur votre environnement Amazon MWAA à l'aide d'un `requirements.txt` fichier de votre compartiment Amazon S3.

Table des matières

- [Conditions préalables](#)
- [Comment ça marche](#)
- [Vue d'ensemble des dépendances Python](#)
 - [Emplacement et limites de taille des dépendances Python](#)
- [Création d'un fichier requirements.txt](#)
 - [Étape 1 : tester les dépendances Python à l'aide de l'utilitaire Amazon MWAA CLI](#)
 - [Deuxième étape : créer le requirements.txt](#)
- [Téléchargement requirements.txt vers Amazon S3](#)
 - [En utilisant le AWS CLI](#)
 - [Utilisation de la console Amazon S3](#)
- [Installation de dépendances Python dans votre environnement](#)
 - [Spécifier le chemin d'accès requirements.txt sur la console Amazon MWAA \(pour la première fois\)](#)
 - [Spécification de la requirements.txt version sur la console Amazon MWAA](#)
- [Accès aux journaux de votre requirements.txt](#)
- [Quelle est la prochaine étape ?](#)

Conditions préalables

Vous aurez besoin des éléments suivants avant de pouvoir effectuer les étapes indiquées sur cette page.

- Autorisations — Votre administrateur Compte AWS doit vous avoir accordé l'accès à la politique de contrôle MWAAFull ConsoleAccess d'accès [Amazon](#) pour votre environnement. En outre, votre

environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder aux AWS ressources utilisées par votre environnement.

- Accès : si vous devez accéder à des référentiels publics pour installer des dépendances directement sur le serveur Web, votre environnement doit être configuré avec un accès au serveur Web du réseau public. Pour plus d'informations, consultez [the section called “Modes d'accès Apache Airflow”](#).
- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAGs plugins personnalisés et vos dépendances Python `requirements.txt` doit être configuré avec l'accès public bloqué et le versionnage activé. `plugins.zip`

Comment ça marche

Sur Amazon MWAA, vous installez toutes les dépendances Python en téléchargeant un `requirements.txt` fichier dans votre compartiment Amazon S3, puis en spécifiant la version du fichier sur la console Amazon MWAA à chaque mise à jour du fichier. Amazon MWAA s'exécute `pip3 install -r requirements.txt` pour installer les dépendances Python sur le planificateur Apache Airflow et sur chacun des travailleurs.

Pour exécuter des dépendances Python dans votre environnement, vous devez effectuer trois opérations :

1. Créez un `requirements.txt` fichier localement.
2. Téléchargez le fichier local `requirements.txt` dans votre compartiment Amazon S3.
3. Spécifiez la version de ce fichier dans le champ Fichier d'exigences de la console Amazon MWAA.

Note

Si c'est la première fois que vous créez et chargez un `requirements.txt` fichier dans votre compartiment Amazon S3, vous devez également spécifier le chemin d'accès au fichier sur la console Amazon MWAA. Vous ne devez effectuer cette étape qu'une seule fois.

Vue d'ensemble des dépendances Python

Vous pouvez installer des suppléments Apache Airflow et d'autres dépendances Python à partir du Python Package Index (PyPi.org), des roues Python (.whl) ou des dépendances Python hébergées sur un dépôt privé conforme à PyPi /PEP-503 sur votre environnement.

Emplacement et limites de taille des dépendances Python

Le planificateur Apache Airflow et les travailleurs recherchent les packages dans le `requirements.txt` fichier et les packages sont installés dans l'environnement à l'adresse. `/usr/local/airflow/.local/bin`

- Limite de taille. Nous recommandons un `requirements.txt` fichier qui fait référence à des bibliothèques dont la taille combinée est inférieure à 1 Go. Plus Amazon MWAA doit installer de bibliothèques, plus le temps de démarrage d'un environnement est long. Bien qu'Amazon MWAA ne limite pas explicitement la taille des bibliothèques installées, si les dépendances ne peuvent pas être installées dans les dix minutes, le service Fargate expirera et tentera de rétablir la stabilité de l'environnement.

Création d'un fichier requirements.txt

Les étapes suivantes décrivent les étapes que nous recommandons pour créer un fichier `requirements.txt` localement.

Étape 1 : tester les dépendances Python à l'aide de l'utilitaire Amazon MWAA CLI

- L'utilitaire d'interface de ligne de commande (CLI) reproduit localement un environnement Amazon Managed Workflows pour Apache Airflow.
- La CLI crée localement une image de conteneur Docker similaire à une image de production Amazon MWAA. Vous pouvez l'utiliser pour exécuter un environnement Apache Airflow local afin de développer et de tester DAGs des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-docker-images](#) sur GitHub.

Deuxième étape : créer le **requirements.txt**

La section suivante décrit comment spécifier les dépendances Python à partir de l'[index des packages Python](#) dans un `requirements.txt` fichier.

Apache Airflow v3

1. Testez localement. Ajoutez des bibliothèques supplémentaires de manière itérative pour trouver la bonne combinaison de packages et de leurs versions, avant de créer un `requirements.txt` fichier. Pour exécuter l'utilitaire Amazon MWAA CLI, reportez-vous à [aws-mwaa-docker-images](#) sur GitHub.
2. Consultez les suppléments du package Apache Airflow. Pour accéder à la liste des packages installés pour Apache Airflow v3 sur Amazon MWAA, consultez le site Web. [aws-mwaa-docker-images requirements.txt](#) GitHub
3. Ajoutez une déclaration de contraintes. Ajoutez le fichier de contraintes pour votre environnement Apache Airflow v3 en haut de votre fichier. `requirements.txt` Les fichiers de contraintes d'Apache Airflow spécifient les versions des fournisseurs disponibles au moment de la publication d'Apache Airflow.

Dans l'exemple suivant, remplacez `{environment-version}` par le numéro de version de votre environnement et `{Python-version}` par la version de Python compatible avec votre environnement.

Pour plus d'informations sur la version de Python compatible avec votre environnement Apache Airflow, reportez-vous à la section Versions d'[Apache Airflow](#).

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

Si le fichier de contraintes détermine que le `xyz==1.0` package n'est pas compatible avec les autres packages de votre environnement, `pip3 install` il n'empêchera pas l'installation de bibliothèques incompatibles dans votre environnement. Si l'installation d'un package échoue, vous pouvez accéder aux journaux d'erreurs de chaque composant Apache Airflow (le planificateur, le programme de travail et le serveur Web) dans le flux de journal correspondant sur Logs. CloudWatch Pour plus d'informations sur les types de journaux, reportez-vous à [the section called "Accès aux journaux Airflow"](#).

4. Paquets Apache Airflow. Ajoutez les [extras du package](#) et la version (`==`). Cela permet d'éviter que des packages portant le même nom, mais dont la version est différente, ne soient installés sur votre environnement.

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

5. Bibliothèques Python. Ajoutez le nom du package et la version (==) dans votre `requirements.txt` fichier. Cela permet d'éviter qu'une future mise à jour de rupture de [PyPi.org](https://pypi.org) ne soit automatiquement appliquée.

```
library == version
```

Exemple Boto3 et psycpg2-binary

Cet exemple est fourni à des fins de démonstration. Les bibliothèques boto et psycpg2-binary sont incluses dans l'installation de base d'Apache Airflow v3 et n'ont pas besoin d'être spécifiées dans un fichier `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycpg2-binary==2.8.6
```

Si un package est spécifié sans version, Amazon MWAA installe la dernière version du package depuis [PyPi.org](https://pypi.org). Cette version peut entrer en conflit avec les autres packages de votre `requirements.txt`.

Apache Airflow v2

1. Testez localement. Ajoutez des bibliothèques supplémentaires de manière itérative pour trouver la bonne combinaison de packages et de leurs versions, avant de créer un `requirements.txt` fichier. Pour exécuter l'utilitaire Amazon MWAA CLI, reportez-vous à [aws-mwaa-docker-images](https://github.com/aws/aws-mwaa-docker-images) sur GitHub.
2. Consultez les suppléments du package Apache Airflow. Pour accéder à la liste des packages installés pour Apache Airflow v2 sur Amazon MWAA, rendez-vous [aws-mwaa-docker-images requirements.txt](https://github.com/aws/aws-mwaa-docker-images/requirements.txt) sur le site Web. GitHub
3. Ajoutez une déclaration de contraintes. Ajoutez le fichier de contraintes pour votre environnement Apache Airflow v2 en haut de votre `requirements.txt` fichier. Les fichiers de contraintes d'Apache Airflow spécifient les versions des fournisseurs disponibles au moment de la publication d'Apache Airflow.

À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. `--constraint` Si vous ne fournissez aucune contrainte, Amazon MWAA vous

en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Dans l'exemple suivant, remplacez `{environment-version}` par le numéro de version de votre environnement et `{Python-version}` par la version de Python compatible avec votre environnement.

Pour plus d'informations sur la version de Python compatible avec votre environnement Apache Airflow, reportez-vous à la section Versions d'[Apache Airflow](#).

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

Si le fichier de contraintes détermine que le `xyz==1.0` package n'est pas compatible avec les autres packages de votre environnement, `pip3 install` il n'empêchera pas l'installation de bibliothèques incompatibles dans votre environnement. Si l'installation d'un package échoue, vous pouvez accéder aux journaux d'erreurs de chaque composant Apache Airflow (le planificateur, le programme de travail et le serveur Web) dans le flux de journal correspondant sur Logs. CloudWatch Pour plus d'informations sur les types de journaux, reportez-vous à [the section called "Accès aux journaux Airflow"](#).

4. Paquets Apache Airflow. Ajoutez les [extras du package](#) et la version (`==`). Cela permet d'éviter que des packages portant le même nom, mais dont la version est différente, ne soient installés sur votre environnement.

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

5. Bibliothèques Python. Ajoutez le nom du package et la version (`==`) dans votre `requirements.txt` fichier. Cela permet d'éviter qu'une future mise à jour de rupture de [PyPi.org](#) ne soit automatiquement appliquée.

```
library == version
```

Exemple Boto3 et psycopg2-binary

Cet exemple est fourni à des fins de démonstration. Les bibliothèques boto et psycopg2-binary sont incluses dans l'installation de base d'Apache Airflow v2 et n'ont pas besoin d'être spécifiées dans un fichier `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycpg2-binary==2.8.6
```

Si un package est spécifié sans version, Amazon MWAA installe la dernière version du package depuis PyPi .org. Cette version peut entrer en conflit avec les autres packages de votre `requirements.txt`.

Téléchargement **requirements.txt** vers Amazon S3

Vous pouvez utiliser la console Amazon S3 ou le AWS Command Line Interface (AWS CLI) pour charger un `requirements.txt` fichier dans votre compartiment Amazon S3.

En utilisant le AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source que vous pouvez utiliser pour interagir avec les AWS services à l'aide de commandes dans votre interface de ligne de commande. Pour effectuer les étapes indiquées sur cette page, vous avez besoin des éléments suivants :

- [AWS CLI — Installez la version 2.](#)
- [AWS CLI — Configuration rapide avec `aws configure`.](#)

Pour télécharger à l'aide du AWS CLI

1. Utilisez la commande suivante pour répertorier tous vos compartiments Amazon S3.

```
aws s3 ls
```

2. Utilisez la commande suivante pour répertorier les fichiers et les dossiers du compartiment Amazon S3 de votre environnement.

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

3. La commande suivante télécharge un `requirements.txt` fichier dans un compartiment Amazon S3.

```
aws s3 cp requirements.txt s3://amzn-s3-demo-bucket/requirements.txt
```

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur Web que vous pouvez utiliser pour créer et gérer les ressources de votre compartiment Amazon S3.

Pour charger à l'aide de la console Amazon S3

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sélectionnez le lien du compartiment S3 dans le code DAG du volet S3 pour ouvrir votre compartiment de stockage dans la console.
4. Choisissez Charger.
5. Choisissez Ajouter un fichier.
6. Sélectionnez la copie locale de votre fichier `requirements.txt`, puis choisissez Upload.

Installation de dépendances Python dans votre environnement

Cette section décrit comment installer les dépendances que vous avez téléchargées dans votre compartiment Amazon S3 en spécifiant le chemin d'accès au fichier `requirements.txt` et en spécifiant la version du fichier `requirements.txt` à chaque mise à jour.

Spécifier le chemin d'accès **`requirements.txt`** sur la console Amazon MWAA (pour la première fois)

Si c'est la première fois que vous créez et chargez un `requirements.txt` fichier dans votre compartiment Amazon S3, vous devez également spécifier le chemin d'accès au fichier sur la console Amazon MWAA. Vous ne devez effectuer cette étape qu'une seule fois.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Dans le code DAG du volet Amazon S3, choisissez Parcourir S3 à côté du fichier d'exigences - champ facultatif.

5. Sélectionnez le `requirements.txt` fichier dans votre compartiment Amazon S3.
6. Choisissez Choisir.
7. Choisissez Suivant, Mettre à jour l'environnement.

Vous pouvez commencer à utiliser les nouveaux packages immédiatement après la fin de la mise à jour de votre environnement.

Spécification de la **requirements.txt** version sur la console Amazon MWAA

Vous devez spécifier la version de votre `requirements.txt` fichier sur la console Amazon MWAA chaque fois que vous chargez une nouvelle version de votre fichier `requirements.txt` dans votre compartiment Amazon S3.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Sur le code DAG dans le volet Amazon S3, choisissez une `requirements.txt` version dans la liste déroulante.
5. Choisissez Suivant, Mettre à jour l'environnement.

Vous pouvez commencer à utiliser les nouveaux packages immédiatement après la fin de la mise à jour de votre environnement.

Accès aux journaux de votre **requirements.txt**

Vous pouvez consulter les journaux d'Apache Airflow pour le planificateur qui planifie vos flux de travail et analyse votre dossier. dags Les étapes suivantes décrivent comment ouvrir le groupe de journaux pour le planificateur sur la console Amazon MWAA et accéder aux journaux Apache Airflow sur la console Logs. CloudWatch

Pour accéder aux journaux d'un **requirements.txt**

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez le groupe de journaux du planificateur Airflow dans le volet de surveillance.
4. Choisissez le `requirements_install_ip` log in Log streams.

5. Reportez-vous à la liste des packages installés sur l'environnement à l'adresse `/usr/local/airflow/.local/bin`. Par exemple :

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Consultez la liste des packages et vérifiez si l'un d'entre eux a rencontré une erreur lors de l'installation. En cas de problème, vous obtenez un message d'erreur similaire à ce qui suit :

```
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))
```

Quelle est la prochaine étape ?

Testez vos DAGs plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-docker-images](#) sur GitHub.

Suppression de fichiers sur Amazon S3

Cette page décrit le fonctionnement du versionnement dans un compartiment Amazon S3 pour un environnement Amazon Managed Workflows for Apache Airflow, ainsi que les étapes à suivre pour supprimer un DAG ou un plugins.zip fichier. requirements.txt

Table des matières

- [Prérequis](#)
- [Vue d'ensemble du versionnement](#)
- [Fonctionnement](#)
- [Supprimer un DAG sur Amazon S3](#)
- [Suppression d'un fichier requirements.txt ou plugins.zip « actuel » d'un environnement](#)
- [Suppression d'une version « non actuelle » \(précédente\) de requirements.txt ou plugins.zip](#)

- [Utiliser les cycles de vie pour supprimer les versions « non actuelles » \(précédentes\) et supprimer automatiquement les marqueurs](#)
- [Exemple de politique de cycle de vie pour supprimer les versions « non actuelles » de requirements.txt et supprimer automatiquement les marqueurs](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Vous aurez besoin des éléments suivants avant de pouvoir effectuer les étapes indiquées sur cette page.

- Autorisations — Votre administrateur Compte AWS doit vous avoir accordé l'accès à la politique de contrôle MWAAFull ConsoleAccess d'accès [Amazon](#) pour votre environnement. En outre, votre environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder aux AWS ressources utilisées par votre environnement.
- Accès : si vous avez besoin d'accéder à des référentiels publics pour installer des dépendances directement sur le serveur Web, votre environnement doit être configuré avec un accès au serveur Web du réseau public. Pour plus d'informations, consultez [the section called “Modes d'accès Apache Airflow”](#).
- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAGs plugins personnalisés et vos dépendances Python requirements.txt doit être configuré avec l'accès public bloqué et le versionnage activé. plugins.zip

Vue d'ensemble du versionnement

Les informations requirements.txt et plugins.zip contenues dans votre compartiment Amazon S3 sont versionnées. Lorsque la gestion des versions des compartiments Amazon S3 est activée pour un objet et qu'un artefact (par exemple, plugins.zip) est supprimé d'un compartiment Amazon S3, le fichier n'est pas entièrement supprimé. Chaque fois qu'un artefact est supprimé sur Amazon S3, une nouvelle copie du fichier est créée. Il s'agit d'un fichier d'erreur 404 (Object not found) /0k portant le libellé suivant : I 'm not here Amazon S3 appelle cela un marqueur de suppression. Un marqueur de suppression est une version « nulle » du fichier avec un nom de clé (ou clé) et un identifiant de version, comme tout autre objet.

Nous vous recommandons de supprimer régulièrement les versions des fichiers et de supprimer les marqueurs afin de réduire les coûts de stockage de votre compartiment Amazon S3. Pour supprimer

complètement les versions de fichiers « non actuelles » (précédentes), vous devez supprimer les versions des fichiers, puis le marqueur de suppression correspondant à la version.

Fonctionnement

Amazon MWAA exécute une opération de synchronisation sur votre compartiment Amazon S3 toutes les trente secondes. Cela entraîne la synchronisation de toutes les suppressions de DAG dans un compartiment Amazon S3 avec l'image Airflow de votre conteneur Fargate.

Pour `requirements.txt` les fichiers `plugins.zip` et, les modifications ne se produisent qu'après une mise à jour de l'environnement, lorsqu'Amazon MWAA crée une nouvelle image Airflow de votre conteneur Fargate à l'aide des plugins personnalisés et des dépendances Python. Si vous supprimez la version actuelle d'un `plugins.zip` fichier `requirements.txt` ou, puis que vous mettez à jour votre environnement sans fournir de nouvelle version pour le fichier supprimé, la mise à jour échouera avec un message d'erreur tel que, `Unable to read version {version number} of file {file name}`.

Supprimer un DAG sur Amazon S3

Un fichier DAG (`.py`) n'est pas versionné et peut être supprimé directement sur la console Amazon S3. Les étapes suivantes décrivent comment supprimer un DAG sur votre compartiment Amazon S3.

Pour supprimer un DAG

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sélectionnez le lien du compartiment S3 dans le code DAG du volet S3 pour ouvrir votre compartiment de stockage dans la console.
4. Choisissez le dossier dags.
5. Sélectionnez le DAG, puis Supprimer.
6. Sous Supprimer des objets ? , tapez `delete`.
7. Choisissez Supprimer les objets.

Note

Apache Airflow préserve l'historique des exécutions DAG. Une fois qu'un DAG a été exécuté dans Apache Airflow, il reste dans la DAGs liste Airflow quel que soit le statut du fichier,

jusqu'à ce que vous le supprimiez dans Apache Airflow. Pour supprimer un DAG dans Apache Airflow, cliquez sur le bouton rouge « Supprimer » dans la colonne Liens.

Suppression d'un fichier requirements.txt ou plugins.zip « actuel » d'un environnement

Actuellement, il n'existe aucun moyen de supprimer un fichier plugins.zip ou requirements.txt d'un environnement après leur ajout, mais nous travaillons sur le problème. Dans l'intervalle, une solution consiste à pointer vers un fichier texte ou un fichier zip vide, respectivement.

Suppression d'une version « non actuelle » (précédente) de requirements.txt ou plugins.zip

Les plugins.zip fichiers requirements.txt et de votre compartiment Amazon S3 sont versionnés sur Amazon MWAA. Si vous souhaitez supprimer complètement ces fichiers de votre compartiment Amazon S3, vous devez récupérer la version actuelle (121212) de l'objet (par exemple, plugins.zip), supprimer la version, puis supprimer le marqueur de suppression pour les versions des fichiers.

Vous pouvez également supprimer les versions de fichiers « non actuelles » (précédentes) sur la console Amazon S3 ; toutefois, vous devrez tout de même supprimer le marqueur de suppression à l'aide de l'une des options suivantes.

- Pour récupérer la version de l'objet, reportez-vous à la section [Extraction des versions d'objets depuis un compartiment activé pour la gestion](#) des versions dans le guide Amazon S3.
- Pour supprimer la version de l'objet, reportez-vous à la section [Supprimer des versions d'objet d'un compartiment activé pour la gestion](#) des versions dans le guide Amazon S3.
- Pour supprimer un marqueur de suppression, reportez-vous à [la section Gestion des marqueurs de suppression](#) du guide Amazon S3.

Utiliser les cycles de vie pour supprimer les versions « non actuelles » (précédentes) et supprimer automatiquement les marqueurs

Vous pouvez configurer une politique de cycle de vie pour votre compartiment Amazon S3 afin de supprimer les versions « non actuelles » (précédentes) des fichiers plugins.zip et requirements.txt de

vosre compartiment Amazon S3 après un certain nombre de jours, ou de supprimer le marqueur de suppression d'un objet expiré.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sous le code DAG dans Amazon S3, choisissez votre compartiment Amazon S3.
4. Choisissez Créer une règle de cycle de vie.

Exemple de politique de cycle de vie pour supprimer les versions « non actuelles » de requirements.txt et supprimer automatiquement les marqueurs

Utilisez l'exemple suivant pour créer une règle de cycle de vie qui supprime définitivement les versions « non actuelles » d'un fichier requirements.txt et leurs marqueurs de suppression après trente jours.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Sous le code DAG dans Amazon S3, choisissez votre compartiment Amazon S3.
4. Choisissez Créer une règle de cycle de vie.
5. Dans Nom de la règle du cycle de vie, tapez `Delete previous requirements.txt versions and delete markers after thirty days`.
6. Dans Préfixe, exigez.
7. Dans Actions relatives aux règles du cycle de vie, choisissez Supprimer définitivement les versions précédentes des objets et Supprimer les marqueurs de suppression expirés ou les téléchargements partitionnés incomplets.
8. Dans Nombre de jours après que les objets sont devenus des versions précédentes, tapez `30`.
9. Dans Marqueurs de suppression d'objets expirés, choisissez Supprimer les marqueurs de suppression d'objets expirés, les objets sont définitivement supprimés au bout de 30 jours.

Quelle est la prochaine étape ?

- Pour en savoir plus sur les marqueurs de suppression Amazon S3, consultez la [section Gestion des marqueurs de suppression](#).

- Pour en savoir plus sur les cycles de vie d'Amazon S3, consultez [Expiring](#) objects.

Réseaux

Ce guide décrit la configuration du réseau Amazon VPC dont vous aurez besoin pour un environnement Amazon MWAA.

Sections

- [À propos de la mise en réseau sur Amazon MWAA](#)
- [Sécurité de votre VPC sur Amazon MWAA](#)
- [Gestion de l'accès aux points de terminaison Amazon VPC spécifiques à un service sur Amazon MWAA](#)
- [Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé](#)
- [Gestion de vos propres points de terminaison Amazon VPC sur Amazon MWAA](#)

À propos de la mise en réseau sur Amazon MWAA

Un Amazon VPC est un réseau virtuel connecté à votre. Compte AWS Il vous apporte la sécurité du cloud et la capacité d'évoluer de manière dynamique en fournissant un contrôle précis de votre infrastructure virtuelle et de la segmentation du trafic réseau. Cette page décrit l'infrastructure Amazon VPC avec le routage public ou privé nécessaire pour prendre en charge un environnement Amazon Managed Workflows for Apache Airflow.

Table des matières

- [Conditions](#)
- [Ce qui est pris en charge](#)
- [Présentation de l'infrastructure VPC](#)
 - [Routage public sur Internet](#)
 - [Routage privé sans accès à Internet](#)
- [Exemples de cas d'utilisation pour un Amazon VPC et le mode d'accès Apache Airflow](#)
 - [L'accès à Internet est autorisé - nouveau réseau Amazon VPC](#)
 - [L'accès à Internet n'est pas autorisé - nouveau réseau Amazon VPC](#)
 - [L'accès à Internet n'est pas autorisé - réseau Amazon VPC existant](#)

Conditions

Routage public

Un réseau Amazon VPC ayant accès à Internet.

Routage privé

Un réseau Amazon VPC sans accès à Internet.

Ce qui est pris en charge

Le tableau suivant décrit les types de produits pris en charge par VPCs Amazon MWAA.

Types de VPC Amazon	Pris en charge	
Un Amazon VPC appartenant au compte qui tente de créer l'environnement.	Oui	
Un Amazon VPC partagé où plusieurs Comptes AWS créent leurs ressources. AWS	Oui	

Présentation de l'infrastructure VPC

Lorsque vous créez un environnement Amazon MWAA, Amazon MWAA crée entre un et deux points de terminaison VPC pour votre environnement en fonction du mode d'accès Apache Airflow que vous avez choisi pour votre environnement. Ces points de terminaison sont présentés sous la forme d'interfaces réseau élastiques (ENIs) avec accès privé IPs dans votre Amazon VPC. Une fois ces points de terminaison créés, tout trafic qui leur est destiné IPs est acheminé de manière privée ou publique vers les AWS services correspondants utilisés par votre environnement.

La section suivante décrit l'infrastructure Amazon VPC requise pour acheminer le trafic en public sur Internet ou en privé au sein de votre Amazon VPC.

Routage public sur Internet

Cette section décrit l'infrastructure Amazon VPC d'un environnement doté d'un routage public. Vous aurez besoin de l'infrastructure VPC suivante :

- Un groupe de sécurité VPC. Un groupe de sécurité VPC agit comme un pare-feu virtuel pour contrôler le trafic réseau entrant (entrant) et sortant (sortant) sur une instance.
 - Jusqu'à 5 groupes de sécurité peuvent être spécifiés.
 - Le groupe de sécurité doit définir une règle d'autoréférencement entrant pour lui-même.
 - Le groupe de sécurité doit spécifier une règle de trafic sortant pour l'ensemble du trafic (0.0.0.0/0; pour IPv6, utiliser : :/0).
 - Le groupe de sécurité doit autoriser tout le trafic dans la règle d'autoréférencement. Par exemple, [\(Recommandé\) Exemple de groupe de sécurité autoréférencé à tous les accès](#).
 - Le groupe de sécurité peut éventuellement restreindre davantage le trafic en spécifiant la plage de ports pour la plage de ports HTTPS 443 et une plage de ports TCP. 5432 Par exemple : [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 5432](#) et [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 443](#).
- Deux sous-réseaux publics. Un sous-réseau public est un sous-réseau associé à une table de routage comportant une route vers une passerelle Internet.
 - Deux sous-réseaux publics sont requis. Cela permet à Amazon MWAA de créer une nouvelle image de conteneur pour votre environnement dans votre autre zone de disponibilité, en cas de défaillance d'un conteneur.
 - Les sous-réseaux doivent se trouver dans des zones de disponibilité différentes. Par exemple, us-east-1a, us-east-1b.
 - Les sous-réseaux doivent être acheminés vers une passerelle NAT (ou instance NAT) dotée d'une adresse IP élastique (EIP).
 - Les sous-réseaux doivent disposer d'une table de routage qui dirige le trafic Internet vers une passerelle Internet.
- Deux sous-réseaux privés. Un sous-réseau privé est un sous-réseau qui n'est pas associé à une table de routage comportant un itinéraire vers une passerelle Internet.
 - Deux sous-réseaux privés sont nécessaires. Cela permet à Amazon MWAA de créer une nouvelle image de conteneur pour votre environnement dans votre autre zone de disponibilité, en cas de défaillance d'un conteneur.

- Les sous-réseaux doivent se trouver dans des zones de disponibilité différentes. Par exemple, `us-east-1a`, `us-east-1b`.
- Les sous-réseaux doivent disposer d'une table de routage vers un périphérique NAT (passerelle ou instance).
- Les sous-réseaux ne doivent pas être acheminés vers une passerelle Internet.
- Défini `assignIPv6AddressOnCreation` sur `true` pour les IPv6 sous-réseaux.
- Pour les sous-réseaux IPv6 privés, vous devez disposer d'une connexion à une passerelle Internet de sortie uniquement (EIGW).
- Une liste de contrôle d'accès réseau (ACL). Une NACL gère (par des règles d'autorisation ou de refus) le trafic entrant et sortant au niveau du sous-réseau.
 - La NACL doit avoir une règle entrante qui autorise tout le trafic (`0.0.0.0/0`; pour IPv6, utiliser `::/0`).
 - La NACL doit avoir une règle sortante qui autorise tout le trafic (`0.0.0.0/0`; pour IPv6, utiliser `::/0`).
 - Par exemple, [Exemple \(recommandé\) ACLs](#).
- Deux passerelles NAT (ou instances NAT). Un périphérique NAT transmet le trafic des instances du sous-réseau privé vers Internet ou d'autres AWS services, puis redirige la réponse vers les instances.
 - Le périphérique NAT doit être attaché à un sous-réseau public. (Un périphérique NAT par sous-réseau public.)
 - Le périphérique NAT doit avoir une IPv4 adresse élastique (EIP) attachée à chaque sous-réseau public.
- Une passerelle Internet. Une passerelle Internet connecte un Amazon VPC à Internet et à d'autres AWS services.
 - Une passerelle Internet doit être attachée à l'Amazon VPC.

Routage privé sans accès à Internet

Cette section décrit l'infrastructure Amazon VPC d'un environnement doté d'un routage privé. Vous aurez besoin de l'infrastructure VPC suivante :

- Un groupe de sécurité VPC. Un groupe de sécurité VPC agit comme un pare-feu virtuel pour contrôler le trafic réseau entrant (entrant) et sortant (sortant) sur une instance.
 - Jusqu'à 5 groupes de sécurité peuvent être spécifiés.

- Le groupe de sécurité doit définir une règle d'autoréférencement entrant pour lui-même.
- Le groupe de sécurité doit spécifier une règle de trafic sortant pour l'ensemble du trafic (0.0.0.0/0; pour IPv6, utiliser : :/0).
- Le groupe de sécurité doit autoriser tout le trafic dans la règle d'autoréférencement. Par exemple, [\(Recommandé\) Exemple de groupe de sécurité autoréférencé à tous les accès](#).
- Le groupe de sécurité peut éventuellement restreindre davantage le trafic en spécifiant la plage de ports pour la plage de ports HTTPS 443 et une plage de ports TCP. 5432 Par exemple : [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 5432](#) et [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 443](#).
- Deux sous-réseaux privés. Un sous-réseau privé est un sous-réseau qui n'est pas associé à une table de routage comportant un itinéraire vers une passerelle Internet.
 - Deux sous-réseaux privés sont nécessaires. Cela permet à Amazon MWAA de créer une nouvelle image de conteneur pour votre environnement dans votre autre zone de disponibilité, en cas de défaillance d'un conteneur.
 - Les sous-réseaux doivent se trouver dans des zones de disponibilité différentes. Par exemple, us-east-1a, us-east-1b.
 - Les sous-réseaux doivent disposer d'une table de routage vers les points de terminaison de votre VPC.
 - Les sous-réseaux doivent disposer d'une table de routage vers un EIGW à télécharger depuis Internet dans le cadre d'un DAG.
 - Les sous-réseaux ne doivent pas avoir de table de routage vers un périphérique NAT (passerelle ou instance), ni de passerelle Internet.
- Une liste de contrôle d'accès réseau (ACL). Une NACL gère (par des règles d'autorisation ou de refus) le trafic entrant et sortant au niveau du sous-réseau.
 - La NACL doit avoir une règle entrante qui autorise tout le trafic (0.0.0.0/0; pour IPv6, utiliser : :/0).
 - La NACL doit avoir une règle de trafic sortant qui refuse tout trafic (0.0.0.0/0; pour IPv6, utilisation : :/0).
 - Par exemple, [Exemple \(recommandé\) ACLs](#).
- Une table de routage locale. Une table de routage locale est une route par défaut pour les communications au sein du VPC.
 - La table de routage locale doit être associée à vos sous-réseaux privés.

- La table de routage locale doit permettre aux instances de votre VPC de communiquer avec votre propre réseau. Par exemple, si vous utilisez un pour accéder AWS Client VPN au point de terminaison de l'interface VPC de votre serveur Web Apache Airflow, la table de routage doit être acheminée vers le point de terminaison VPC.
- Des points de terminaison VPC pour chaque AWS service utilisé par votre environnement et des points de terminaison VPC Apache Airflow dans le même environnement Amazon VPC que votre environnement Amazon Région AWS MWAA.
- Un point de terminaison VPC pour chaque AWS service utilisé par l'environnement et des points de terminaison VPC pour Apache Airflow. Par exemple, [Points de terminaison VPC \(obligatoires\)](#).
- Le DNS privé doit être activé sur les points de terminaison VPC.
- Les points de terminaison VPC doivent être associés aux deux sous-réseaux privés de votre environnement.
- Les points de terminaison VPC doivent être associés au groupe de sécurité de votre environnement.
- La politique de point de terminaison VPC pour chaque point de terminaison doit être configurée pour autoriser l'accès aux AWS services utilisés par l'environnement. Par exemple, [\(Recommandé\) Exemple de politique de point de terminaison VPC pour autoriser tous les accès](#).
- Une politique de point de terminaison VPC pour Amazon S3 doit être configurée pour autoriser l'accès au compartiment. Par exemple, [\(Recommandé\) Exemple de politique de point de terminaison de la passerelle Amazon S3 pour autoriser l'accès au bucket](#).

Exemples de cas d'utilisation pour un Amazon VPC et le mode d'accès Apache Airflow

Cette section décrit les différents cas d'utilisation de l'accès au réseau dans votre Amazon VPC et le mode d'accès au serveur Web Apache Airflow choisi sur la console Amazon MWAA.

L'accès à Internet est autorisé - nouveau réseau Amazon VPC

Si l'accès à Internet dans votre VPC est autorisé par votre organisation et que vous souhaitez que les utilisateurs accèdent à votre serveur Web Apache Airflow via Internet :

1. Créez un réseau Amazon VPC avec accès à Internet.

2. Créez un environnement avec le mode d'accès au réseau public pour votre serveur Web Apache Airflow.
3. Ce que nous recommandons : nous vous recommandons d'utiliser le modèle de CloudFormation démarrage rapide qui crée simultanément l'infrastructure Amazon VPC, un compartiment Amazon S3 et un environnement Amazon MWAA. Pour en savoir plus, reportez-vous à [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#).

Si l'accès à Internet dans votre VPC est autorisé par votre organisation et que vous souhaitez limiter l'accès au serveur Web Apache Airflow aux utilisateurs de votre VPC :

1. Créez un réseau Amazon VPC avec accès à Internet.
2. Créez un mécanisme pour accéder au point de terminaison de l'interface VPC de votre serveur Web Apache Airflow depuis votre ordinateur.
3. Créez un environnement avec le mode d'accès réseau privé pour votre serveur Web Apache Airflow.
4. Ce que nous recommandons :
 - a. Nous vous recommandons d'utiliser la console Amazon MWAA dans [Première option : créer le réseau VPC sur la console Amazon MWAA](#) ou le CloudFormation modèle dans [Deuxième option : créer un réseau Amazon VPC avec accès à Internet](#).
 - b. Nous vous recommandons de configurer l'accès à l'aide d'un AWS Client VPN à votre serveur Web Apache Airflow dans. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#)

L'accès à Internet n'est pas autorisé - nouveau réseau Amazon VPC

Si l'accès à Internet dans votre VPC n'est pas autorisé par votre organisation :

1. Créez un réseau Amazon VPC sans accès à Internet.
2. Créez un mécanisme pour accéder au point de terminaison de l'interface VPC de votre serveur Web Apache Airflow depuis votre ordinateur.
3. Créez des points de terminaison VPC pour chaque AWS service utilisé par votre environnement.
4. Créez un environnement avec le mode d'accès réseau privé pour votre serveur Web Apache Airflow.
5. Ce que nous recommandons :

- a. Nous vous recommandons d'utiliser le CloudFormation modèle pour créer un Amazon VPC sans accès à Internet et les points de terminaison VPC pour chaque service AWS utilisé par Amazon MWAA dans. [Troisième option : créer un réseau Amazon VPC sans accès à Internet](#)
- b. Nous vous recommandons de configurer l'accès à l'aide d'un AWS Client VPN à votre serveur Web Apache Airflow dans. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#)

L'accès à Internet n'est pas autorisé - réseau Amazon VPC existant

Si l'accès à Internet dans votre VPC n'est pas autorisé par votre organisation et que vous disposez déjà du réseau Amazon VPC requis sans accès à Internet :

1. Créez des points de terminaison VPC pour chaque AWS service utilisé par votre environnement.
2. Créez des points de terminaison VPC pour Apache Airflow.
3. Créez un mécanisme pour accéder au point de terminaison de l'interface VPC de votre serveur Web Apache Airflow depuis votre ordinateur.
4. Créez un environnement avec le mode d'accès réseau privé pour votre serveur Web Apache Airflow.
5. Ce que nous recommandons :
 - a. Nous vous recommandons de créer et d'attacher les points de terminaison VPC nécessaires à chaque AWS service utilisé par Amazon MWAA, ainsi que les points de terminaison VPC nécessaires pour Apache Airflow in. [Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé](#)
 - b. Nous vous recommandons de configurer l'accès à l'aide d'un AWS Client VPN à votre serveur Web Apache Airflow dans. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#)

Sécurité de votre VPC sur Amazon MWAA

Cette page décrit les composants Amazon VPC utilisés pour sécuriser votre environnement Amazon Managed Workflows for Apache Airflow et les configurations nécessaires pour ces composants.

Table des matières

- [Conditions](#)
- [Aperçu de la sécurité](#)
- [Listes de contrôle d'accès au réseau \(ACLs\)](#)
 - [Exemple \(recommandé\) ACLs](#)
- [Groupes de sécurité VPC](#)
 - [\(Recommandé\) Exemple de groupe de sécurité autoréférencé à tous les accès](#)
 - [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 5432](#)
 - [\(Facultatif\) Exemple de groupe de sécurité qui restreint l'accès entrant au port 443](#)
- [Politiques de point de terminaison VPC \(routage privé uniquement\)](#)
 - [\(Recommandé\) Exemple de politique de point de terminaison VPC pour autoriser tous les accès](#)
 - [\(Recommandé\) Exemple de politique de point de terminaison de la passerelle Amazon S3 pour autoriser l'accès au bucket](#)

Conditions

Routage public

Un réseau Amazon VPC ayant accès à Internet.

Routage privé

Un réseau Amazon VPC sans accès à Internet.

Aperçu de la sécurité

Les groupes de sécurité et les listes de contrôle d'accès (ACLs) permettent de contrôler le trafic réseau sur les sous-réseaux et les instances de votre Amazon VPC à l'aide de règles que vous spécifiez.

- Le trafic réseau à destination et en provenance d'un sous-réseau peut être contrôlé par des listes de contrôle d'accès (ACLs). Vous n'avez besoin que d'une seule ACL, et la même ACL peut être utilisée sur plusieurs environnements.
- Le trafic réseau à destination et en provenance d'une instance peut être contrôlé par un groupe de sécurité Amazon VPC. Vous pouvez utiliser entre un et cinq groupes de sécurité par environnement.

- Le trafic réseau à destination et en provenance d'une instance peut également être contrôlé par des politiques de point de terminaison VPC. [Si l'accès à Internet au sein de votre Amazon VPC n'est pas autorisé par votre organisation et que vous utilisez un réseau Amazon VPC avec un routage privé, une politique de point de terminaison VPC est requise pour les points de terminaison VPC et les points de terminaison VPC Apache AWS Airflow.](#)

Listes de contrôle d'accès au réseau (ACLs)

Une [liste de contrôle d'accès réseau \(ACL\)](#) peut gérer (selon des règles d'autorisation ou de refus) le trafic entrant et sortant au niveau du sous-réseau. Une ACL est apatride, ce qui signifie que les règles entrantes et sortantes doivent être spécifiées séparément et explicitement. Il est utilisé pour spécifier les types de trafic réseau autorisés à entrer ou à sortir des instances d'un réseau VPC.

Chaque Amazon VPC possède une ACL par défaut qui autorise tout le trafic entrant et sortant. Vous pouvez modifier les règles ACL par défaut ou créer une ACL personnalisée et l'associer à vos sous-réseaux. Un sous-réseau ne peut être associé qu'à une seule ACL à la fois, mais une ACL peut être attachée à plusieurs sous-réseaux.

Exemple (recommandé) ACLs

L'exemple suivant affiche les règles ACL entrantes et sortantes qui peuvent être utilisées pour un Amazon VPC avec un routage public ou un routage privé.

Numéro de règle	Type	Protocole	Plage de ports	Source	Autoriser/ Refuser
100	Tout IPv4 le trafic	Tous	Tous	0.0.0.0/0	Allow
*	Tout IPv4 le trafic	Tous	Tous	0.0.0.0/0	Refuser

Groupes de sécurité VPC

Un [groupe de sécurité VPC](#) agit comme un pare-feu virtuel qui contrôle le trafic réseau au niveau de l'instance. Un groupe de sécurité est dynamique, ce qui signifie que lorsqu'une connexion entrante

est autorisée, il est autorisé à répondre. Il est utilisé pour spécifier les types de trafic réseau autorisés à entrer depuis les instances d'un réseau VPC.

Chaque Amazon VPC possède un groupe de sécurité par défaut. Par défaut, il n'a aucune règle d'entrée. Il dispose d'une règle de trafic sortant qui autorise tout le trafic sortant. Vous pouvez modifier les règles du groupe de sécurité par défaut ou créer un groupe de sécurité personnalisé et l'associer à votre Amazon VPC. Sur Amazon MWAA, vous devez configurer des règles entrantes et sortantes pour diriger le trafic vers vos passerelles NAT.

(Recommandé) Exemple de groupe de sécurité autoréférencé à tous les accès

L'exemple suivant affiche les règles du groupe de sécurité entrant qui autorisent tout le trafic pour un Amazon VPC avec un routage public ou un routage privé. Dans cet exemple, le groupe de sécurité est une règle d'autoréférencement à lui-même.

Type	Protocole	Type de source	Source		
Tout le trafic	Tous	Tous	sg-0909e8e81919/-groupe my-mwaa-vpc-security		

L'exemple suivant montre les règles du groupe de sécurité sortant.

Type	Protocole	Type de source	Source		
Tout le trafic	Tous	Tous	0.0.0.0/0		

(Facultatif) Exemple de groupe de sécurité qui restreint l'accès entrant au port 5432

L'exemple suivant affiche les règles du groupe de sécurité entrant qui autorisent tout le trafic HTTPS sur le port 5432 pour la base de données de métadonnées Amazon Aurora PostgreSQL (détenue par Amazon MWAA) pour votre environnement.

Note

Si vous choisissez de restreindre le trafic à l'aide de cette règle, vous devrez en ajouter une autre pour autoriser le trafic TCP sur le port 443.

Type	Protocole	Plage de ports	Source type (Type de source)	Source	
TCP personnalisé	TCP	5432	Personnalisé	sg-0909e8e81919/-groupe my-mwaa-vpc-security	

(Facultatif) Exemple de groupe de sécurité qui restreint l'accès entrant au port 443

L'exemple suivant affiche les règles du groupe de sécurité entrant qui autorisent tout le trafic TCP sur le port 443 pour le serveur Web Apache Airflow.

Type	Protocole	Plage de ports	Source type (Type de source)	Source	
HTTPS	TCP	443	Personnalisé	sg-0909e8e81919/-groupe my-mwaa-vpc-security	

Politiques de point de terminaison VPC (routage privé uniquement)

Une politique de point de [terminaison VPC \(AWS PrivateLink\)](#) contrôle l'accès aux AWS services depuis votre sous-réseau privé. Une politique de point de terminaison VPC est une politique de ressources IAM que vous attachez à votre passerelle VPC ou à votre point de terminaison d'interface. Cette section décrit les autorisations nécessaires pour les politiques de point de terminaison VPC pour chaque point de terminaison VPC.

Nous vous recommandons d'utiliser une politique de point de terminaison d'interface VPC pour chacun des points de terminaison VPC que vous avez créés, qui autorise un accès complet à tous les AWS services, et d'utiliser votre rôle d'exécution exclusivement pour les autorisations. AWS

(Recommandé) Exemple de politique de point de terminaison VPC pour autoriser tous les accès

L'exemple suivant présente une politique de point de terminaison d'interface VPC pour un Amazon VPC doté d'un routage privé.

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

(Recommandé) Exemple de politique de point de terminaison de la passerelle Amazon S3 pour autoriser l'accès au bucket

L'exemple suivant présente une politique de point de terminaison de passerelle VPC qui fournit l'accès aux compartiments Amazon S3 requis pour les opérations Amazon ECR pour un Amazon VPC avec routage privé. Cela est nécessaire pour que votre image Amazon ECR soit récupérée, en plus du compartiment dans lequel vos fichiers DAGs et ceux de support sont stockés.

```
{
  "Statement": [
```

```
{
  "Sid": "Access-to-specific-bucket-only",
  "Principal": "*",
  "Action": [
    "s3:GetObject"
  ],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::prod-us-east-1-starport-layer-bucket/*"]
}
```

Gestion de l'accès aux points de terminaison Amazon VPC spécifiques à un service sur Amazon MWAA

Un point de terminaison VPC (AWS PrivateLink) que vous pouvez utiliser pour connecter votre VPC en privé à des services hébergés AWS sans avoir besoin d'une passerelle Internet, d'un périphérique NAT, d'un VPN ou de proxys de pare-feu. Ces points de terminaison sont des périphériques virtuels évolutifs horizontalement et hautement disponibles qui permettent la communication entre les instances de votre VPC AWS et les services. Cette page décrit les points de terminaison VPC créés par Amazon MWAA et explique comment accéder au point de terminaison VPC pour votre serveur Web Apache Airflow si vous avez choisi le mode d'accès au réseau privé sur Amazon Managed Workflows for Apache Airflow.

Table des matières

- [Tarification](#)
- [Vue d'ensemble des points de terminaison VPC](#)
 - [Mode d'accès au réseau public](#)
 - [Mode d'accès au réseau privé](#)
- [Autorisation d'utiliser d'autres AWS services](#)
- [Accès aux points de terminaison VPC](#)
 - [Accès aux points de terminaison VPC sur la console Amazon VPC](#)
 - [Identification des adresses IP privées de votre serveur Web Apache Airflow et de son point de terminaison VPC](#)
- [Accès au point de terminaison VPC de votre serveur Web Apache Airflow \(accès réseau privé\)](#)
 - [À l'aide d'un AWS Client VPN](#)

- [Utilisation d'un hôte Linux Bastion](#)
- [Utilisation d'un Load Balancer \(avancé\)](#)

Tarification

- [AWS PrivateLink Tarification](#)

Vue d'ensemble des points de terminaison VPC

Lorsque vous créez un environnement Amazon MWAA, Amazon MWAA crée entre un et deux points de terminaison VPC pour votre environnement. Ces points de terminaison sont présentés sous la forme d'interfaces réseau élastiques (ENIs) avec accès privé IPs dans votre Amazon VPC. Une fois ces points de terminaison créés, tout trafic qui leur est destiné IPs est acheminé de manière privée ou publique vers les AWS services correspondants utilisés par votre environnement.

Mode d'accès au réseau public

Si vous avez choisi le mode d'accès au réseau public pour votre serveur Web Apache Airflow, le trafic réseau est routé publiquement sur Internet.

- Amazon MWAA crée un point de terminaison d'interface VPC pour votre base de données de métadonnées Amazon Aurora PostgreSQL. Le point de terminaison est créé dans les zones de disponibilité mappées à vos sous-réseaux privés et est indépendant des autres. Comptes AWS
- Amazon MWAA lie ensuite une adresse IP de vos sous-réseaux privés aux points de terminaison de l'interface. Ceci est conçu pour soutenir la meilleure pratique qui consiste à lier une adresse IP unique à chaque zone de disponibilité de l'Amazon VPC.

Mode d'accès au réseau privé

Si vous avez choisi le mode d'accès réseau privé pour votre serveur Web Apache Airflow, le trafic réseau est acheminé de manière privée au sein de votre Amazon VPC.

- Amazon MWAA crée un point de terminaison d'interface VPC pour votre serveur Web Apache Airflow et un point de terminaison d'interface pour votre base de données de métadonnées Amazon Aurora PostgreSQL. Les points de terminaison sont créés dans les zones de disponibilité mappées à vos sous-réseaux privés et sont indépendants des autres. Comptes AWS

- Amazon MWAA lie ensuite une adresse IP de vos sous-réseaux privés aux points de terminaison de l'interface. Ceci est conçu pour soutenir la meilleure pratique qui consiste à lier une adresse IP unique à chaque zone de disponibilité de l'Amazon VPC.

Autorisation d'utiliser d'autres AWS services

Les points de terminaison de l'interface utilisent le rôle d'exécution de votre environnement dans Gestion des identités et des accès AWS (IAM) pour gérer les autorisations d'accès aux AWS ressources utilisées par votre environnement. Au fur et à mesure que de nouveaux AWS services sont activés pour un environnement, chaque service nécessite que vous configuriez l'autorisation en utilisant le rôle d'exécution de votre environnement. Pour ajouter des autorisations, reportez-vous à [Rôle d'exécution Amazon MWAA](#).

Si vous avez choisi le mode d'accès au réseau privé pour votre serveur Web Apache Airflow, vous devez également autoriser l'autorisation dans la politique de point de terminaison du VPC pour chaque point de terminaison. Pour en savoir plus, reportez-vous à [the section called “Politiques de point de terminaison VPC \(routage privé uniquement\)”](#).

Accès aux points de terminaison VPC

Cette section décrit comment accéder aux points de terminaison VPC créés par Amazon MWAA et comment identifier les adresses IP privées de votre point de terminaison VPC Apache Airflow.

Accès aux points de terminaison VPC sur la console Amazon VPC

La section suivante présente les étapes pour accéder aux points de terminaison VPC créés par Amazon MWAA, ainsi qu'à tous les points de terminaison VPC que vous pourriez avoir créés si vous utilisez le routage privé pour votre Amazon VPC.

Pour accéder aux points de terminaison VPC

1. Ouvrez la [page Endpoints](#) sur la console Amazon VPC.
2. Sélectionnez votre Région AWS.
3. Reportez-vous aux points de terminaison de l'interface VPC créés par Amazon MWAA et à tous les points de terminaison VPC que vous pourriez avoir créés si vous utilisez le routage privé dans votre Amazon VPC.

Pour en savoir plus sur les points de terminaison du service VPC requis pour un Amazon VPC avec routage privé, reportez-vous à [Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé](#)

Identification des adresses IP privées de votre serveur Web Apache Airflow et de son point de terminaison VPC

Les étapes suivantes décrivent comment récupérer le nom d'hôte de votre serveur Web Apache Airflow et de son point de terminaison d'interface VPC, ainsi que leurs adresses IP privées.

1. Utilisez la commande suivante AWS Command Line Interface (AWS CLI) pour récupérer le nom d'hôte de votre serveur Web Apache Airflow.

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

Vous obtenez une réponse similaire à la suivante :

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

2. Exécutez une commande dig sur le nom d'hôte renvoyé dans la réponse à la commande précédente. Par exemple :

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

Vous obtenez une réponse similaire à la suivante :

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

3. Utilisez la commande suivante AWS Command Line Interface (AWS CLI) pour récupérer le nom DNS du point de terminaison VPC renvoyé dans la réponse à la commande précédente. Par exemple :

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

Vous obtenez une réponse similaire à la suivante :

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

4. Exécutez une commande nslookup ou dig sur le nom de votre hôte Apache Airflow et le nom DNS de son point de terminaison VPC pour récupérer les adresses IP. Par exemple :

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

Vous obtenez une réponse similaire à la suivante :

```
192.0.5.1  
192.0.6.1
```

Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)

Si vous avez choisi le mode d'accès réseau privé pour votre serveur Web Apache Airflow, vous devez créer un mécanisme pour accéder au point de terminaison de l'interface VPC de votre serveur Web Apache Airflow. Vous devez utiliser le même Amazon VPC, le même groupe de sécurité VPC et les mêmes sous-réseaux privés que votre environnement Amazon MWAA pour ces ressources.

À l'aide d'un AWS Client VPN

AWS Client VPN est un service VPN géré basé sur le client que vous pouvez utiliser pour accéder en toute sécurité à vos AWS ressources et à celles de votre réseau sur site. Il fournit une connexion TLS sécurisée depuis n'importe quel endroit à l'aide du client OpenVPN.

Nous vous recommandons de suivre le didacticiel Amazon MWAA pour configurer un Client VPN : [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#).

Utilisation d'un hôte Linux Bastion

Un hôte bastion est un serveur dont le but est de fournir un accès à un réseau privé à partir d'un réseau externe, par exemple via Internet à partir de votre ordinateur. Les instances Linux se trouvent dans un sous-réseau public et sont configurées avec un groupe de sécurité qui autorise l'accès SSH depuis le groupe de sécurité attaché à l'EC2 instance Amazon sous-jacente exécutant l'hôte bastion.

Nous vous recommandons de suivre le didacticiel Amazon MWAA pour configurer un hôte Linux Bastion : [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#)

Utilisation d'un Load Balancer (avancé)

La section suivante présente les configurations que vous devez appliquer à un [Application Load Balancer](#).

1. **Groupes cibles** Vous devez utiliser des groupes cibles qui pointent vers les adresses IP privées de votre serveur Web Apache Airflow et de son point de terminaison d'interface VPC. Nous vous recommandons de spécifier les deux adresses IP privées comme cibles enregistrées, car l'utilisation d'une seule adresse peut réduire la disponibilité. Pour plus d'informations sur la façon d'identifier les adresses IP privées, reportez-vous à [the section called "Identification des adresses IP privées de votre serveur Web Apache Airflow et de son point de terminaison VPC"](#).
2. **Codes de statut.** Nous vous recommandons d'utiliser 200 des codes 302 d'état dans les paramètres de votre groupe cible. Dans le cas contraire, les cibles risquent d'être signalées comme étant défectueuses si le point de terminaison VPC du serveur Web Apache Airflow répond par une erreur. 302 Redirect
3. **Écouteur HTTPS.** Vous devez spécifier le port cible pour le serveur Web Apache Airflow. Par exemple :

Protocole	Port
HTTPS	443

4. **Nouveau domaine ACM.** Si vous souhaitez associer un SSL/TLS certificat à AWS Certificate Manager, vous devez créer un nouveau domaine pour l'écouteur HTTPS de votre équilibreur de charge.
5. **Région du certificat ACM.** Si vous souhaitez associer un SSL/TLS certificat à AWS Certificate Manager, vous devez le télécharger dans le même environnement Région AWS que celui de votre environnement. Par exemple :
 - Exemple région pour télécharger le certificat

```
aws acm import-certificate --certificate fileb://Certificate.pem --certificate-chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --  
region us-west-2
```

Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé

Un réseau Amazon VPC existant sans accès à Internet a besoin de points de terminaison de service VPC supplémentaires () pour AWS PrivateLink utiliser Apache Airflow sur Amazon Managed Workflows pour Apache Airflow. Cette page décrit les points de terminaison VPC requis pour les AWS services utilisés par Amazon MWAA, les points de terminaison VPC requis pour Apache Airflow et comment créer et associer les points de terminaison VPC à un Amazon VPC existant avec un routage privé.

Table des matières

- [Tarification](#)
- [Réseau privé et routage privé](#)
- [Points de terminaison VPC \(obligatoires\)](#)
- [Connexion des points de terminaison VPC requis](#)
 - [Points de terminaison VPC requis pour les services AWS](#)
 - [Points de terminaison VPC requis pour Apache Airflow](#)
- [\(Facultatif\) Activez les adresses IP privées pour le point de terminaison de votre interface Amazon S3 VPC](#)
 - [Utilisation de la Route 53](#)
 - [VPCs avec DNS personnalisé](#)

Tarification

- [AWS PrivateLink Tarification](#)

Réseau privé et routage privé

Le mode d'accès au réseau privé limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement.

Lorsque vous créez un environnement avec accès privé à un serveur Web, vous devez empaqueter toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans

votrerequirements.txt. Pour obtenir des instructions sur l'emballage et l'installation de vos dépendances à l'aide de wheel, reportez-vous à [la section Gestion des dépendances à l'aide de Python Wheel](#).

L'image suivante indique où trouver l'option Réseau privé sur la console Amazon MWAA.

- Routage privé. Un [Amazon VPC sans accès à Internet](#) limite le trafic réseau au sein du VPC. Cette page part du principe que votre Amazon VPC n'a pas accès à Internet et nécessite des points de terminaison VPC pour chaque AWS service utilisé par votre environnement, ainsi que des points de terminaison VPC pour Apache Airflow dans le même environnement et Amazon VPC que votre environnement Région AWS Amazon MWAA.

Points de terminaison VPC (obligatoires)

La section suivante affiche les points de terminaison VPC requis pour un Amazon VPC sans accès à Internet. Il répertorie les points de terminaison VPC pour chaque AWS service utilisé par Amazon MWAA, y compris les points de terminaison VPC nécessaires à Apache Airflow.

```
com.amazonaws.us-east-1.s3
com.amazonaws.us-east-1.monitoring
com.amazonaws.us-east-1.logs
com.amazonaws.us-east-1.sqs
com.amazonaws.us-east-1.kms
```

Note

Lorsque vous utilisez Transit Gateway ou tout autre routage qui ne mène pas directement aux points de terminaison de l' AWS API, nous vous recommandons d'ajouter AWS PrivateLink à vos sous-réseaux privés Amazon MWAA les services suivants :

- Amazon S3
- Amazon SQS
- CloudWatch Journaux
- CloudWatch métriques
- AWS KMS (le cas échéant)

Cela garantit que votre environnement Amazon MWAA peut communiquer de manière sécurisée et efficace avec ces services sans acheminer le trafic via l'Internet public, améliorant ainsi la sécurité et les performances.

Connexion des points de terminaison VPC requis

Cette section décrit les étapes à suivre pour associer les points de terminaison VPC requis pour un Amazon VPC doté d'un routage privé.

Points de terminaison VPC requis pour les services AWS

La section suivante présente les étapes permettant d'associer les points de terminaison VPC pour les AWS services utilisés par un environnement à un Amazon VPC existant.

Pour associer des points de terminaison VPC à vos sous-réseaux privés

1. Ouvrez la [page Endpoints](#) sur la console Amazon VPC.
2. Sélectionnez votre Région AWS.
3. Créez le point de terminaison pour Amazon S3 :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.s3**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Nous vous recommandons de choisir le point de terminaison de service répertorié pour le type de passerelle.

Par exemple, **com.amazonaws.us-west-2.s3 amazon Gateway**

- d. Choisissez Amazon VPC dans VPC de votre environnement.
- e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que le DNS privé est activé en sélectionnant Activer le nom DNS.
- f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
- g. Choisissez Accès complet dans la politique.
- h. Choisissez Créer un point de terminaison.

4. Créez le point de terminaison pour CloudWatch les journaux :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.logs**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.
 - f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
5. Créez le point de terminaison pour CloudWatch la surveillance :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.monitoring**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.
 - f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
6. Créez le point de terminaison pour Amazon SQS :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.sqs**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.

- f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
7. Créez le point de terminaison pour AWS KMS :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.kms**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.
 - f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.

Points de terminaison VPC requis pour Apache Airflow

La section suivante présente les étapes à suivre pour associer les points de terminaison VPC pour Apache Airflow à un Amazon VPC existant.

Pour associer des points de terminaison VPC à vos sous-réseaux privés

1. Ouvrez la [page Endpoints](#) sur la console Amazon VPC.
2. Sélectionnez votre Région AWS.
3. Créez le point de terminaison pour l'API Apache Airflow :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.airflow.api**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.

- f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
4. Créez le premier point de terminaison pour l'environnement Apache Airflow :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.airflow.env**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.
 - f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
5. Créez le deuxième point de terminaison pour les opérations Apache Airflow :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.airflow.ops**, puis appuyez sur la touche Entrée de votre clavier.
 - c. Sélectionnez le point de terminaison du service.
 - d. Choisissez Amazon VPC dans VPC de votre environnement.
 - e. Assurez-vous que vos deux sous-réseaux privés situés dans des zones de disponibilité différentes sont sélectionnés et que l'option Activer le nom DNS est activée.
 - f. Choisissez les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.

(Facultatif) Activez les adresses IP privées pour le point de terminaison de votre interface Amazon S3 VPC

Les points de terminaison de l'interface Amazon S3 ne prennent pas en charge le DNS privé. Les demandes du point de terminaison S3 sont toujours résolues vers une adresse IP publique. Pour

transformer l'adresse S3 en adresse IP privée, vous devez ajouter une [zone hébergée privée dans Route 53](#) pour le point de terminaison régional S3.

Utilisation de la Route 53

Cette section décrit les étapes à suivre pour activer les adresses IP privées pour un point de terminaison de l'interface S3 à l'aide de la Route 53.

1. Créez une zone hébergée privée pour le point de terminaison de votre interface Amazon S3 VPC (tel que `s3.eu-west-1.amazonaws.com`) et associez-la à votre Amazon VPC.
2. Créez un enregistrement ALIAS A pour votre point de terminaison d'interface VPC Amazon S3 (tel que `s3.eu-west-1.amazonaws.com`) qui correspond au nom DNS de votre point de terminaison d'interface VPC.
3. Créez un enregistrement générique ALIAS A pour le point de terminaison de votre interface Amazon S3 (tel que, `*.s3.eu-west-1.amazonaws.com`) qui correspond au nom DNS du point de terminaison de l'interface VPC.

VPCs avec DNS personnalisé

Si votre Amazon VPC utilise un routage DNS personnalisé, vous devez apporter les modifications dans votre résolveur DNS (et non dans Route 53, généralement une EC2 instance exécutant un serveur DNS) en créant un enregistrement CNAME. Exemples :

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value: *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

Gestion de vos propres points de terminaison Amazon VPC sur Amazon MWAA

Amazon MWAA utilise les points de terminaison Amazon VPC pour s'intégrer aux AWS différents services nécessaires à la configuration d'un environnement Apache Airflow. La gestion de vos propres points de terminaison comprend deux cas d'utilisation principaux :

1. Cela signifie que vous pouvez créer des environnements Apache Airflow dans un Amazon VPC partagé lorsque vous utilisez un [AWS Organizations](#) pour gérer plusieurs Comptes AWS ressources et partager des ressources.

2. Il vous permet d'utiliser des politiques d'accès plus restrictives en limitant vos autorisations aux ressources spécifiques qui utilisent vos points de terminaison.

Si vous choisissez de gérer vos propres points de terminaison VPC, vous êtes responsable de créer vos propres points de terminaison pour la base de données RDS pour PostgreSQL de l'environnement et pour le serveur Web de l'environnement.

Pour plus d'informations sur la manière dont Amazon MWAA déploie Apache Airflow dans le cloud, consultez le schéma d'architecture [Amazon MWAA](#).

 Important

Amazon MWAA ne valide pas la sélection du type d'adresse IP (`AddressType`) pour les points de terminaison gérés par le client. Assurez-vous donc de le spécifier correctement `AddressType` (les options valides sont `IPv4` `IPv6`).

Création d'un environnement dans un Amazon VPC partagé

Si vous gérez plusieurs Comptes AWS personnes [AWS Organizations](#) partageant des ressources, vous pouvez utiliser des points de terminaison VPC gérés par le client avec Amazon MWAA pour partager les ressources de l'environnement avec un autre compte de votre organisation.

Lorsque vous configurez l'accès VPC partagé, le compte propriétaire du VPC Amazon principal (propriétaire) partage les deux sous-réseaux privés requis par Amazon MWAA avec d'autres comptes (participants) appartenant à la même organisation. Les comptes participants qui partagent ces sous-réseaux peuvent afficher, créer, modifier et supprimer des environnements dans le Amazon VPC partagé.

Supposons que vous ayez un compte `Owner`, qui fait office de Root compte dans l'organisation et possède les ressources Amazon VPC, et un compte de participant `Participant`, membre de la même organisation. Lors de la `Participant` création d'un nouvel Amazon MWAA dans Amazon VPC avec lequel il est partagé `Owner`, Amazon MWAA crée d'abord les ressources VPC du service, puis entre dans [PENDING](#) un état pendant 72 heures au maximum.

Lorsque l'état de l'environnement passe de `CREATING` à `PENDING`, un directeur agissant pour le compte de `Owner` crée les points de terminaison requis. Pour ce faire, Amazon MWAA répertorie la base de données et le point de terminaison du serveur Web dans la console Amazon MWAA. Vous

pouvez également appeler l'action [GetEnvironment](#) API pour obtenir les points de terminaison du service.

Note

Si le VPC Amazon que vous utilisez pour partager des ressources est un Amazon VPC privé, vous devez tout de même suivre les étapes décrites dans [the section called “Gestion de l'accès aux points de terminaison VPC”](#). Cette rubrique couvre la configuration d'un ensemble différent de points de terminaison Amazon VPC liés à d'autres AWS services intégrés AWS, tels qu'Amazon ECR, Amazon ECS et Amazon SQS. Ces services sont essentiels pour exploiter et gérer votre environnement Apache Airflow dans le cloud.

Conditions préalables

Avant de créer un environnement Amazon MWAA dans un VPC partagé, vous avez besoin des ressources suivantes :

- Et Compte AWS, Owner à utiliser comme compte propriétaire de l'Amazon VPC.
- Une unité [AWS Organizations](#) organisationnelle, MyOrganization créée en tant que racine.
- Un second Compte AWS, Participant, en dessous MyOrganization pour servir le compte du participant qui crée le nouvel environnement.

En outre, nous vous recommandons de vous familiariser avec les [responsabilités et les autorisations des propriétaires et des participants](#) lorsque vous partagez des ressources dans Amazon VPC.

Création de l'Amazon VPC

Tout d'abord, créez un nouvel Amazon VPC que les comptes propriétaire et participant partageront :

1. Connectez-vous à la console en utilisant Owner, puis CloudFormation ouvrez-la. Utilisez le modèle suivant pour créer une pile. Cette pile fournit un certain nombre de ressources réseau, notamment un Amazon VPC et les sous-réseaux que les deux comptes partageront dans ce scénario.

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

```
Description: >-
```

```
This template deploys a VPC, with a pair of public and private subnets spread  
across two Availability Zones. It deploys an internet gateway, with a default
```

route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.

Parameters:

EnvironmentName:

Description: An environment name that is prefixed to resource names

Type: String

Default: mwaa-

VpcCIDR:

Description: Please enter the IP range (CIDR notation) for this VPC

Type: String

Default: 10.192.0.0/16

PublicSubnet1CIDR:

Description: >-

Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone

Type: String

Default: 10.192.10.0/24

PublicSubnet2CIDR:

Description: >-

Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone

Type: String

Default: 10.192.11.0/24

PrivateSubnet1CIDR:

Description: >-

Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone

Type: String

Default: 10.192.20.0/24

PrivateSubnet2CIDR:

Description: >-

Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone

Type: String

Default: 10.192.21.0/24

Resources:

VPC:

Type: 'AWS::EC2::VPC'

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

```

    Value: !Ref EnvironmentName
InternetGateway:
  Type: 'AWS::EC2::InternetGateway'
  Properties:
    Tags:
      - Key: Name
        Value: !Ref EnvironmentName
InternetGatewayAttachment:
  Type: 'AWS::EC2::VPCGatewayAttachment'
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC
PublicSubnet1:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ1)'
PublicSubnet2:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select
      - 1
      - !GetAZs ''
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ2)'
PrivateSubnet1:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
    CidrBlock: !Ref PrivateSubnet1CIDR

```



```

    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub '${EnvironmentName} Private Subnet (AZ1)'
PrivateSubnet2:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select
      - 1
      - !GetAZs ''
    CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub '${EnvironmentName} Private Subnet (AZ2)'
NatGateway1EIP:
  Type: 'AWS::EC2::EIP'
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
NatGateway2EIP:
  Type: 'AWS::EC2::EIP'
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
NatGateway1:
  Type: 'AWS::EC2::NatGateway'
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1
NatGateway2:
  Type: 'AWS::EC2::NatGateway'
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2
PublicRouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${EnvironmentName} Public Routes'
DefaultPublicRoute:

```

```
Type: 'AWS::EC2::Route'
DependsOn: InternetGatewayAttachment
Properties:
  RouteTableId: !Ref PublicRouteTable
  DestinationCidrBlock: 0.0.0.0/0
  GatewayId: !Ref InternetGateway
PublicSubnet1RouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1
PublicSubnet2RouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2
PrivateRouteTable1:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
  Tags:
    - Key: Name
      Value: !Sub '${EnvironmentName} Private Routes (AZ1)'
DefaultPrivateRoute1:
  Type: 'AWS::EC2::Route'
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1
PrivateSubnet1RouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1
PrivateRouteTable2:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
  Tags:
    - Key: Name
      Value: !Sub '${EnvironmentName} Private Routes (AZ2)'
DefaultPrivateRoute2:
  Type: 'AWS::EC2::Route'
  Properties:
```

```

    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2
PrivateSubnet2RouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2
SecurityGroup:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupName: 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
  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

```

```

Zone
    PrivateSubnet1:
        Description: A reference to the private subnet in the 1st Availability
        Value: !Ref PrivateSubnet1
    PrivateSubnet2:
        Description: A reference to the private subnet in the 2nd Availability
        Value: !Ref PrivateSubnet2
    SecurityGroupIngress:
        Description: Security group with self-referencing inbound rule
        Value: !Ref SecurityGroupIngress

```

2. Une fois les nouvelles ressources Amazon VPC provisionnées, accédez à la AWS Resource Access Manager console, puis choisissez Create resource share.
3. Choisissez les sous-réseaux que vous avez créés lors de la première étape dans la liste des sous-réseaux disponibles avec lesquels vous pouvez partager. Participant

Création de l'environnement

Procédez comme suit pour créer un environnement Amazon MWAA avec des points de terminaison Amazon VPC gérés par le client.

1. Connectez-vous à l'aide Participant de la console Amazon MWAA et ouvrez-la. Terminez la première étape : spécifiez les détails pour spécifier un compartiment Amazon S3, un dossier DAG et les dépendances pour votre nouvel environnement. Pour plus d'informations, reportez-vous à la section [Mise en route](#).
2. Sur la page Configurer les paramètres avancés, sous Mise en réseau, choisissez les sous-réseaux du Amazon VPC partagé.
3. Sous Gestion des terminaux, choisissez CLIENT dans la liste déroulante.
4. Conservez la valeur par défaut pour les autres options de la page, puis choisissez Créer un environnement sur la page Réviser et créer.

L'environnement commence dans un CREATING état, puis passe àPENDING. Lorsque l'environnement l'estPENDING, notez le nom du service de point de terminaison de base de données et le nom du service de point de terminaison du serveur Web (si vous configurez un serveur Web privé) à l'aide de la console.

Lorsque vous créez un nouvel environnement à l'aide de la console Amazon MWAA, Amazon MWAA crée un nouveau groupe de sécurité avec les règles d'entrée et de sortie requises. Notez l'ID du groupe de sécurité.

Dans la section suivante, Owner nous utiliserons les points de terminaison du service et l'ID du groupe de sécurité pour créer de nouveaux points de terminaison Amazon VPC dans le Amazon VPC partagé.

Création des points de terminaison Amazon VPC

Procédez comme suit pour créer les points de terminaison Amazon VPC requis pour votre environnement.

1. Connectez-vous à l' AWS Management Console utilisateurOwner, puis ouvrez <https://console.aws.amazon.com/vpc/>.
2. Choisissez Groupes de sécurité dans le panneau de navigation de gauche, puis créez un nouveau groupe de sécurité dans l'Amazon VPC partagé en appliquant les règles entrantes et sortantes suivantes :

	Type	Protocole	Source type (Type de source)	Source
Entrant	Tout le trafic	Tous	Tous	Votre groupe de sécurité environnementale
Sortant	Tout le trafic	Tous	Tous	0.0.0.0/0

Warning

Le Owner compte doit configurer un groupe de sécurité dans le Owner compte pour autoriser le trafic du nouvel environnement vers le Amazon VPC partagé. Vous pouvez le faire en créant un nouveau groupe de sécurité dans Owner ou en modifiant un groupe existant.

3. Choisissez Endpoints, puis créez de nouveaux points de terminaison pour la base de données d'environnement et le serveur Web (en mode privé) en utilisant les noms de service des points de terminaison indiqués dans les étapes précédentes. Choisissez le VPC Amazon partagé, les sous-réseaux que vous avez utilisés pour l'environnement et le groupe de sécurité de l'environnement.

En cas de succès, l'environnement passera de PENDING retour à CREATING, puis enfin à AVAILABLE. Lorsque c'est le cas AVAILABLE, vous pouvez vous connecter à la console Apache Airflow.

Résolution des problèmes liés au partage d'Amazon VPC

Utilisez la référence suivante pour résoudre les problèmes que vous rencontrez lors de la création d'environnements dans un Amazon VPC partagé.

PENDING État de l'environnement **CREATE_FAILED** dans le passé

- Vérifiez Owner que les sous-réseaux sont partagés en Participant utilisant [AWS Resource Access Manager](#).
- Vérifiez que les points de terminaison Amazon VPC pour la base de données et le serveur Web sont créés dans les mêmes sous-réseaux associés à l'environnement.
- Vérifiez que le groupe de sécurité utilisé avec vos points de terminaison autorise le trafic provenant des groupes de sécurité utilisés pour l'environnement. Le Owner compte crée des règles qui font référence au groupe de sécurité Participant sous la forme *123456789012/security-group-id* :

Type	Protocole	Source type (Type de source)	Source
Tout le trafic	Tous	Tous	<i>123456789012 /sg-0909e8e81919</i>

Pour plus d'informations, reportez-vous à la section [Responsabilités et autorisations des propriétaires et des participants](#)

PENDING État bloqué de l'environnement

Vérifiez l'état de chaque point de terminaison VPC pour vous assurer qu'il l'est. `Available` Si vous configurez un environnement avec un serveur Web privé, vous devez également créer un point de terminaison pour le serveur Web. Si l'environnement est bloqué `PENDING`, cela peut indiquer que le point de terminaison du serveur Web privé est manquant.

The Vpc Endpoint Service '*vpce-service-name*' does not exist

Erreur reçue

Si vous faites référence à l'erreur suivante, vérifiez que le compte qui crée les points de terminaison se trouve dans le `Owner` compte propriétaire du VPC partagé :

```
ClientError: An error occurred (InvalidServiceName) when calling the
CreateVpcEndpoint operation:
```

```
The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Tutoriels pour Amazon Managed Workflows pour Apache Airflow

Ce guide inclut des step-by-step didacticiels sur l'utilisation et la configuration d'un environnement Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#)
- [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#)
- [Tutoriel : Restreindre l'accès d'un utilisateur Amazon MWAA à un sous-ensemble de DAGs](#)
- [Tutoriel : Automatisez la gestion des points de terminaison de votre propre environnement sur Amazon MWAA](#)

Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN

Ce didacticiel explique les étapes à suivre pour créer un tunnel VPN entre votre ordinateur et le serveur Web Apache Airflow pour votre environnement Amazon Managed Workflows for Apache Airflow. Pour vous connecter à Internet via un tunnel VPN, vous devez d'abord créer un AWS Client VPN point de terminaison. Une fois configuré, un point de terminaison VPN Client agit comme un serveur VPN permettant une connexion sécurisée entre votre ordinateur et les ressources de votre VPC. Vous vous connecterez ensuite au Client VPN depuis votre ordinateur à l'aide de [AWS Client VPN for Desktop](#).

Sections

- [Réseau privé](#)
- [Cas d'utilisation](#)
- [Avant de commencer](#)
- [Objectifs](#)
- [\(Facultatif\) Première étape : identifier votre VPC, vos règles CIDR et la sécurité de votre VPC](#)
- [Deuxième étape : créer les certificats du serveur et du client](#)
- [Troisième étape : enregistrer le CloudFormation modèle localement](#)
- [Quatrième étape : créer la CloudFormation pile VPN du Client](#)

- [Étape 5 : associer des sous-réseaux à votre Client VPN](#)
- [Sixième étape : ajouter une règle d'entrée d'autorisation à votre Client VPN](#)
- [Étape 7 : Téléchargez le fichier de configuration du point de terminaison du Client VPN](#)
- [Huitième étape : Connectez-vous au AWS Client VPN](#)
- [Quelle est la prochaine étape ?](#)

Réseau privé

Ce didacticiel part du principe que vous avez choisi le mode d'accès au réseau privé pour votre serveur Web Apache Airflow.

Le mode d'accès au réseau privé limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement.

Lorsque vous créez un environnement avec accès privé à un serveur Web, vous devez empaqueter toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans `requirements.txt`. Pour obtenir des instructions sur l'empaquetage et l'installation de vos dépendances à l'aide de wheel, reportez-vous à [la section Gestion des dépendances à l'aide de Python Wheel](#).

L'image suivante indique où trouver l'option Réseau privé sur la console Amazon MWAA.

Cas d'utilisation

Vous pouvez utiliser ce didacticiel avant ou après avoir créé un environnement Amazon MWAA. Vous devez utiliser les mêmes Amazon VPC, les mêmes groupes de sécurité VPC et les mêmes sous-réseaux privés que votre environnement. Si vous utilisez ce didacticiel après avoir créé un environnement Amazon MWAA, une fois les étapes terminées, vous pouvez revenir à la console Amazon MWAA et changer le mode d'accès de votre serveur Web Apache Airflow en mode réseau privé.

Avant de commencer

1. Vérifiez les autorisations des utilisateurs. Assurez-vous que votre compte dans Gestion des identités et des accès AWS (IAM) dispose des autorisations suffisantes pour créer et gérer les ressources VPC.

2. Utilisez votre VPC Amazon MWAA. Ce didacticiel part du principe que vous associez le Client VPN à un VPC existant. Le VPC Amazon doit être Région AWS identique à un environnement Amazon MWAA et disposer de deux sous-réseaux privés. Si vous n'avez pas créé de VPC Amazon, utilisez le CloudFormation modèle dans. [Troisième option : créer un réseau Amazon VPC sans accès à Internet](#)

Objectifs

Dans le cadre de ce didacticiel, vous effectuerez les tâches suivantes :

1. Créez un AWS Client VPN point de terminaison à l'aide d'un CloudFormation modèle pour un Amazon VPC existant.
2. Générez des certificats et des clés de serveur et de client, puis téléchargez le certificat de serveur et la clé AWS Certificate Manager dans un environnement Amazon MWAA Région AWS comme dans un environnement Amazon MWAA.
3. Téléchargez et modifiez un fichier de configuration du point de terminaison Client VPN pour votre Client VPN, puis utilisez-le pour créer un profil VPN afin de vous connecter à l'aide du Client VPN pour ordinateur de bureau.

(Facultatif) Première étape : identifier votre VPC, vos règles CIDR et la sécurité de votre VPC

La section suivante explique comment IDs rechercher votre Amazon VPC, votre groupe de sécurité VPC et comment identifier les règles CIDR dont vous aurez besoin pour créer votre Client VPN lors des étapes suivantes.

Identifiez vos règles CIDR

La section suivante explique comment identifier les règles CIDR dont vous aurez besoin pour créer votre Client VPN.

Pour identifier le CIDR de votre Client VPN

1. Ouvrez la [VPCs page Your Amazon](#) sur la console Amazon VPC.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir un environnement Amazon MWAA Région AWS identique.

3. Choisissez votre Amazon VPC.
4. En supposant que CIDRs les sous-réseaux privés sont les suivants :
 - Sous-réseau privé 1 : 10.192.10.0 /24
 - Sous-réseau privé 2 : 10.192.11.0 /24

Si le CIDR de votre Amazon VPC est /16 10.192.0.0, le CIDR IPv4 client que vous devez spécifier pour votre VPN client sera 10.192.0.0. /22

5. Enregistrez cette valeur CIDR et la valeur de votre ID VPC pour les étapes suivantes.

Identifiez votre VPC et vos groupes de sécurité

La section suivante explique comment trouver l'ID de votre Amazon VPC et des groupes de sécurité, dont vous aurez besoin pour créer votre Client VPN.

Note

Vous utilisez peut-être plusieurs groupes de sécurité. Vous devrez spécifier tous les groupes de sécurité de votre VPC dans les étapes suivantes.

Pour identifier les groupes de sécurité

1. Ouvrez la [page Security Groups](#) sur la console Amazon VPC.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir le Région AWS.
3. Recherchez l'Amazon VPC dans l'ID du VPC et identifiez les groupes de sécurité associés au VPC.
4. Enregistrez l'ID de vos groupes de sécurité et de votre VPC pour les étapes suivantes.

Deuxième étape : créer les certificats du serveur et du client

Un point de terminaison VPN Client prend en charge uniquement les tailles de clés RSA 1024-bits et 2048-bits. La section suivante explique comment utiliser OpenVPN easy-rsa pour générer les certificats et clés du serveur et du client, puis télécharger les certificats sur ACM à l'aide de (). AWS Command Line Interface AWS CLI

Pour créer les certificats clients

1. Suivez ces étapes rapides pour créer et télécharger les certificats vers ACM via la AWS CLI section [Authentification et autorisation du client : authentification mutuelle](#).
2. Au cours de ces étapes, vous devez spécifier la même chose Région AWS qu'un environnement Amazon MWAA dans la AWS CLI commande lors du téléchargement de vos certificats de serveur et de client. Voici quelques exemples de la manière de spécifier la région dans ces commandes :

- a. Exemple région pour le certificat de serveur

```
aws acm import-certificate --certificate fileb://server.crt --private-key  
fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
```

- b. Exemple région pour le certificat client

```
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. Après ces étapes, enregistrez la valeur renvoyée dans la AWS CLI réponse pour le certificat de serveur et le certificat client ARNs. Vous allez les spécifier ARNs dans votre CloudFormation modèle pour créer le Client VPN.
3. Au cours de ces étapes, un certificat client et une clé privée sont enregistrés sur votre ordinateur. Voici un exemple où trouver ces informations d'identification :
- a. Exemple sur macOS

Sur macOS, le contenu est enregistré dans `/Users/your-user/custom_folder`. Si vous listez tout (`ls -a`) le contenu de ce répertoire, vous obtenez quelque chose de similaire à ce qui suit :

```
.  
..  
ca.crt  
client1.domain.tld.crt  
client1.domain.tld.key  
server.crt  
server.key
```

- b. Après ces étapes, enregistrez le contenu ou notez l'emplacement du certificat client dans `client1.domain.tld.crt` et de la clé privée `client1.domain.tld.key`. Vous allez ajouter ces valeurs au fichier de configuration de votre Client VPN.

Troisième étape : enregistrer le CloudFormation modèle localement

La section suivante contient le CloudFormation modèle pour créer le Client VPN. Vous devez spécifier les mêmes Amazon VPC, les mêmes groupes de sécurité VPC et les mêmes sous-réseaux privés que votre environnement Amazon MWAA.

- Copiez le contenu du modèle suivant et enregistrez-le localement `sousmwaa_vpn_client.yaml`. Vous pouvez également [télécharger le modèle](#).

Remplacez les valeurs suivantes :

- YOUR_CLIENT_ROOT_CERTIFICATE_ARN**— L'ARN de votre certificat `client1.domain.tld` dans `ClientRootCertificateChainArn`
- YOUR_SERVER_CERTIFICATE_ARN**— L'ARN de votre certificat de serveur dans `ServerCertificateArn`.
- La règle IPv4 CIDR du client dans `ClientCidrBlock` Une règle CIDR de `10.192.0.0/22` est fournie.
- Votre identifiant Amazon VPC dans `VpcId` Un VPC de `vpc-010101010101` est fourni.
- Votre groupe IDs de sécurité VPC dans `SecurityGroupIds` Un groupe de sécurité de `sg-0101010101` est fourni.

```
AWSTemplateFormatVersion: 2010-09-09
Description: This template deploys a VPN Client Endpoint.
Resources:
  ClientVpnEndpoint:
    Type: 'AWS::EC2::ClientVpnEndpoint'
    Properties:
      AuthenticationOptions:
        - Type: "certificate-authentication"
      MutualAuthentication:
        ClientRootCertificateChainArn: "YOUR_CLIENT_ROOT_CERTIFICATE_ARN"
      ClientCidrBlock: 10.192.0.0/22
      ClientConnectOptions:
        Enabled: false
```

```
ConnectionLogOptions:
  Enabled: false
Description: "MWA Client VPN"
DnsServers: []
SecurityGroupIds:
  - sg-0101010101
SelfServicePortal: ''
ServerCertificateArn: "YOUR_SERVER_CERTIFICATE_ARN"
SplitTunnel: true
TagSpecifications:
  - ResourceType: "client-vpn-endpoint"
    Tags:
      - Key: Name
        Value: MWA-Client-VPN
TransportProtocol: udp
VpcId: vpc-010101010101
VpnPort: 443
```

Note

Si vous utilisez plusieurs groupes de sécurité pour votre environnement, vous pouvez spécifier plusieurs groupes de sécurité au format suivant :

```
SecurityGroupIds:
  - sg-0112233445566778b
  - sg-0223344556677889f
```

Quatrième étape : créer la CloudFormation pile VPN du Client

Pour créer le AWS Client VPN

1. Ouvrez la [AWS CloudFormation console](#).
2. Choisissez Le modèle est prêt, puis chargez un fichier modèle.
3. Choisissez Choisir un fichier, puis sélectionnez votre `mwa_vpn_client.yaml` fichier.
4. Appuyez sur Suivant, Suivant.
5. Sélectionnez l'accusé de réception, puis choisissez Create stack.

Étape 5 : associer des sous-réseaux à votre Client VPN

Pour associer des sous-réseaux privés au AWS Client VPN

1. Ouvrez la [console VPC Amazon](#).
2. Choisissez la page Points de terminaison du Client VPN.
3. Sélectionnez votre Client VPN, puis cliquez sur l'onglet Associations, Associer.
4. Choisissez ce qui suit dans la liste déroulante :
 - Votre Amazon VPC en VPC.
 - L'un de vos sous-réseaux privés dans Choisissez un sous-réseau à associer.
5. Choisissez Associer.

Note

Plusieurs minutes sont nécessaires pour que le VPC et le sous-réseau soient associés au Client VPN.

Sixième étape : ajouter une règle d'entrée d'autorisation à votre Client VPN

Vous devez ajouter une règle d'entrée d'autorisation à l'aide de la règle CIDR de votre VPC à votre Client VPN. Si vous souhaitez autoriser des utilisateurs ou des groupes spécifiques à partir de votre groupe Active Directory ou de votre fournisseur d'identité (IdP) basé sur SAML, reportez-vous aux [règles d'autorisation du guide](#) Client VPN.

Pour ajouter le CIDR au AWS Client VPN

1. Ouvrez la [console VPC Amazon](#).
2. Choisissez la page Points de terminaison du Client VPN.
3. Sélectionnez votre Client VPN, puis cliquez sur l'onglet Autorisation, Autoriser l'entrée.
4. Spécifiez les paramètres suivants :
 - La règle CIDR de votre Amazon VPC dans le réseau de destination doit être activée.Exemples :

```
10.192.0.0/16
```

- Choisissez Autoriser l'accès à tous les utilisateurs dans Accorder l'accès à.
- Entrez un nom descriptif dans Description.

5. Choisissez Ajouter une règle d'autorisation.

 Note

En fonction des composants réseau de votre Amazon VPC, vous pouvez également avoir besoin de cette règle d'entrée d'autorisation dans votre liste de contrôle d'accès réseau (NACL).

Étape 7 : Téléchargez le fichier de configuration du point de terminaison du Client VPN

Pour télécharger le fichier de configuration :

1. Suivez ces étapes rapides pour télécharger le fichier de configuration du VPN client sur [Télécharger le fichier de configuration du point de terminaison du VPN client](#).
 2. Au cours de ces étapes, il vous est demandé d'ajouter une chaîne au nom DNS de votre point de terminaison VPN Client. Voici un exemple :
- Exemple nom DNS du point de terminaison

Si le nom DNS du point de terminaison de votre Client VPN est :

```
remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
```

Vous pouvez ajouter une chaîne pour identifier le point de terminaison de votre Client VPN comme suit :

```
remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
```


3. Au cours de ces étapes, il vous est demandé d'ajouter le contenu du certificat client entre un nouvel ensemble de `<cert></cert>` balises et le contenu de la clé privée entre un nouvel ensemble de `<key></key>` balises. Voici un exemple :
 - a. Ouvrez une invite de commande et remplacez les répertoires par l'emplacement de votre certificat client et de votre clé privée.
 - b. Exemple macOS `client1.domain.tld.crt`

Pour afficher le contenu du `client1.domain.tld.crt` fichier sur macOS, vous pouvez utiliser `cat client1.domain.tld.crt`.

Copiez la valeur depuis le terminal et collez-la `downloaded-client-config.ovpn` comme ceci :

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

- c. Exemple macOS `client1.domain.tld.key`

Pour afficher le contenu du `client1.domain.tld.key`, vous pouvez utiliser `cat client1.domain.tld.key`.

Copiez la valeur depuis le terminal et collez-la `downloaded-client-config.ovpn` comme ceci :

```
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>
```

Huitième étape : Connectez-vous au AWS Client VPN

Le formulaire client AWS Client VPN est fourni gratuitement. Vous pouvez connecter directement votre ordinateur AWS Client VPN pour bénéficier d'une expérience end-to-end VPN.

Pour vous connecter au Client VPN

1. Téléchargez et installez le [AWS Client VPN pour ordinateur de bureau](#).
2. Ouvrez le AWS Client VPN.
3. Choisissez Fichier, Profils gérés dans le menu du client VPN.
4. Choisissez Ajouter un profil, puis sélectionnez le `downloaded-client-config.ovpn`.
5. Entrez un nom descriptif dans Nom d'affichage.
6. Choisissez Ajouter un profil, OK.
7. Choisissez Se connecter.

Une fois connecté au Client VPN, vous devez vous déconnecter des autres VPNs pour accéder aux ressources de votre Amazon VPC.

Note

Il se peut que vous deviez quitter le client et recommencer avant de pouvoir vous connecter.

Quelle est la prochaine étape ?

- Découvrez comment créer un environnement Amazon MWAA dans [Commencez avec Amazon Managed Workflows for Apache Airflow](#). Vous devez créer un environnement Région AWS identique à celui du VPN client et utiliser le même VPC, les mêmes sous-réseaux privés et le même groupe de sécurité que le VPN client.

Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion

Ce didacticiel explique les étapes à suivre pour créer un tunnel SSH entre votre ordinateur et le serveur Web Apache Airflow pour votre environnement Amazon Managed Workflows for Apache Airflow. Cela suppose que vous avez déjà créé un environnement Amazon MWAA. Une fois configuré, un hôte Linux Bastion agit comme un serveur de saut permettant une connexion sécurisée entre votre ordinateur et les ressources de votre VPC. Vous utiliserez ensuite un module complémentaire de gestion de proxy SOCKS pour contrôler les paramètres du proxy dans votre navigateur afin d'accéder à votre interface utilisateur Apache Airflow.

Sections

- [Réseau privé](#)
- [Cas d'utilisation](#)
- [Avant de commencer](#)
- [Objectifs](#)
- [Première étape : créer l'instance de bastion](#)
- [Deuxième étape : créer le tunnel SSH](#)
- [Troisième étape : configurer le groupe de sécurité Bastion en tant que règle entrante](#)
- [Étape 4 : Copier l'URL d'Apache Airflow](#)
- [Étape 5 : configurer les paramètres du proxy](#)
- [Sixième étape : ouvrir l'interface utilisateur d'Apache Airflow](#)
- [Quelle est la prochaine étape ?](#)

Réseau privé

Ce didacticiel part du principe que vous avez choisi le mode d'accès au réseau privé pour votre serveur Web Apache Airflow.

Le mode d'accès au réseau privé limite l'accès à l'interface utilisateur d'Apache Airflow aux utilisateurs de votre Amazon VPC qui ont obtenu l'accès à la politique [IAM](#) de votre environnement.

Lorsque vous créez un environnement avec accès privé à un serveur Web, vous devez empaqueter toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans

votrerequirements.txt. Pour obtenir des instructions sur l'emballage et l'installation de vos dépendances à l'aide de wheel, reportez-vous à [la section Gestion des dépendances à l'aide de Python Wheel](#).

L'image suivante indique où trouver l'option Réseau privé sur la console Amazon MWAA.

Cas d'utilisation

Vous pouvez utiliser ce didacticiel après avoir créé un environnement Amazon MWAA. Vous devez utiliser les mêmes Amazon VPC, les mêmes groupes de sécurité VPC et les mêmes sous-réseaux publics que votre environnement.

Avant de commencer

1. Vérifiez les autorisations des utilisateurs. Assurez-vous que votre compte dans Gestion des identités et des accès AWS (IAM) dispose des autorisations suffisantes pour créer et gérer les ressources VPC.
2. Utilisez votre VPC Amazon MWAA. Ce didacticiel part du principe que vous associez l'hôte bastion à un VPC existant. Le VPC Amazon doit se trouver dans la même région que votre environnement Amazon MWAA et disposer de deux sous-réseaux privés, comme défini dans. [Création du réseau VPC](#)
3. Créez une clé SSH. Vous devez créer une clé Amazon EC2 SSH (.pem) dans la même région que votre environnement Amazon MWAA pour vous connecter aux serveurs virtuels. Si vous n'avez pas de clé SSH, reportez-vous à la section [Créer ou importer une paire de clés](#) dans le guide de l'EC2 utilisateur Amazon.

Objectifs

Dans le cadre de ce didacticiel, vous effectuerez les tâches suivantes :

1. Créez une instance d'hôte Linux Bastion à l'aide d'un [CloudFormation modèle pour un VPC existant](#).
2. Autorisez le trafic entrant vers le groupe de sécurité de l'instance Bastion à l'aide d'une règle d'entrée sur le port. 22
3. Autorisez le trafic entrant depuis le groupe de sécurité d'un environnement Amazon MWAA vers le groupe de sécurité de l'instance Bastion.

4. Créez un tunnel SSH vers l'instance de bastion.
5. Installez et configurez le FoxyProxy module complémentaire pour le navigateur Firefox afin d'accéder à l'interface utilisateur d'Apache Airflow.

Première étape : créer l'instance de bastion

La section suivante décrit les étapes de création de l'instance Linux Bastion à l'aide d'un [CloudFormation modèle pour un VPC existant](#) sur la CloudFormation console.

Pour créer l'hôte Linux Bastion

1. Ouvrez la page de [démarrage rapide du déploiement](#) sur la CloudFormation console.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir celui qui correspond à votre Région AWS environnement Amazon MWAA.
3. Choisissez Suivant.
4. Entrez un nom dans le champ de texte Stack name, tel `quemwaa-linux-bastion`.
5. Dans le volet Paramètres, Configuration réseau, choisissez les options suivantes :
 - a. Choisissez l'ID VPC de votre environnement Amazon MWAA.
 - b. Choisissez l'ID de sous-réseau public 1 de votre environnement Amazon MWAA.
 - c. Choisissez l'ID de sous-réseau public 2 de votre environnement Amazon MWAA.
 - d. Entrez la plage d'adresses la plus étroite possible (par exemple, une plage d'adresses CIDR interne) dans Allowed Bastion external access CIDR.

Note

Le moyen le plus simple d'identifier une plage est d'utiliser la même plage CIDR que celle de vos sous-réseaux publics. Par exemple, les sous-réseaux publics du CloudFormation modèle de la [Création du réseau VPC](#) page sont `10.192.10.0/24` et `10.192.11.0/24`.

6. Dans le volet EC2 de configuration d'Amazon, choisissez ce qui suit :
 - a. Choisissez votre clé SSH dans la liste déroulante dans Nom de la paire de clés.
 - b. Entrez un nom dans le champ Nom d'hôte de Bastion.
 - c. Choisissez true pour le transfert TCP.

 Warning

Le transfert TCP doit être défini sur true à cette étape. Dans le cas contraire, vous ne pourrez pas créer de tunnel SSH à l'étape suivante.

7. Appuyez sur Suivant, Suivant.
8. Sélectionnez l'accusé de réception, puis choisissez Create stack.

Pour en savoir plus sur l'architecture de votre hôte Linux Bastion, reportez-vous à [Linux Bastion Hosts on the AWS Cloud : Architecture](#).

Deuxième étape : créer le tunnel SSH

Les étapes suivantes décrivent comment créer le tunnel SSH vers votre bastion Linux. Un tunnel SSH reçoit la demande de votre adresse IP locale vers le bastion Linux, c'est pourquoi le transfert TCP pour le bastion Linux a été défini dans les étapes précédentes. true

macOS/Linux

Pour créer un tunnel à l'aide de la ligne de commande

1. Ouvrez la page [Instances](#) sur la EC2 console Amazon.
2. Choisissez un type d'instance.
3. Copiez l'adresse dans le IPv4 DNS public. Par exemple, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Dans votre invite de commande, accédez au répertoire dans lequel votre clé SSH est stockée.
5. Exécutez la commande suivante pour vous connecter à l'instance de bastion à l'aide de ssh. Remplacez la valeur de l'échantillon par le nom de votre clé SSH. `mykeypair.pem`

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

Pour créer un tunnel à l'aide de PuTTY

1. Ouvrez la page [Instances](#) sur la EC2 console Amazon.
2. Choisissez un type d'instance.
3. Copiez l'adresse dans le IPv4 DNS public. Par exemple, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Ouvrez [PuTTY](#), sélectionnez Session.
5. Entrez le nom d'hôte dans Nom d'hôte sous la forme `ec2-user@ YOUR_PUBLIC_IPV4_DNS` et le port sous la forme. 22
6. Développez l'onglet SSH, sélectionnez Auth. Dans le fichier de clé privée pour l'authentification, choisissez votre fichier « ppk » local.
7. Sous SSH, choisissez l'onglet Tunnels, puis sélectionnez les options Dynamique et Auto.
8. Dans Port source, ajoutez le 8157 port (ou tout autre port non utilisé), puis laissez le champ Port de destination vide. Choisissez Ajouter.
9. Choisissez l'onglet Session et entrez un nom de session. Par exemple SSH Tunnel1.
10. Choisissez Enregistrer, Ouvrir.

Note

Il se peut que vous deviez saisir une phrase secrète pour votre clé publique.

Note

Si vous recevez un `Permission denied (publickey)` message d'erreur, nous vous recommandons d'utiliser l'[AWS Support Troubleshoot SSH](#) outil et de choisir Run this Automation (console) pour résoudre les problèmes liés à votre configuration SSH.

Troisième étape : configurer le groupe de sécurité Bastion en tant que règle entrante

L'accès aux serveurs et l'accès régulier à Internet depuis les serveurs sont autorisés grâce à un groupe de sécurité de maintenance spécial attaché à ces serveurs. Les étapes suivantes décrivent

comment configurer le groupe de sécurité Bastion en tant que source de trafic entrant vers le groupe de sécurité VPC d'un environnement.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Dans le volet Mise en réseau, choisissez le groupe de sécurité VPC.
4. Choisissez Modifier les règles entrantes.
5. Choisissez Ajouter une règle.
6. Choisissez l'ID de votre groupe de sécurité VPC dans la liste déroulante Source.
7. Laissez les options restantes vides ou définissez leurs valeurs par défaut.
8. Sélectionnez Enregistrer les règles.

Étape 4 : Copier l'URL d'Apache Airflow

Les étapes suivantes décrivent comment ouvrir la console Amazon MWAA et copier l'URL dans l'interface utilisateur d'Apache Airflow.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Copiez l'URL dans l'interface utilisateur d'Airflow pour les étapes suivantes.

Étape 5 : configurer les paramètres du proxy

Si vous utilisez un tunnel SSH avec réacheminement de port dynamique, vous devez utiliser un module complémentaire de gestion de proxy SOCKS pour contrôler les paramètres de proxy dans votre navigateur. Par exemple, vous pouvez utiliser la `--proxy-server` fonctionnalité de Chromium pour démarrer une session de navigateur ou utiliser l' FoxyProxy extension dans le Firefox navigateur Mozilla.

Première option : configurer un tunnel SSH à l'aide de la redirection de port locale

Si vous ne souhaitez pas utiliser de proxy SOCKS, vous pouvez configurer un tunnel SSH en utilisant la redirection de port locale. L'exemple de commande suivant accède à l'interface EC2 ResourceManagerWeb Amazon en transférant le trafic sur le port local 8157.

1. Ouvrez une nouvelle fenêtre d'invite de commandes.

2. Entrez la commande suivante pour ouvrir un tunnel SSH.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-vpce.us-east-1.airflow.amazonaws.com:443 ubuntu@YOUR_PUBLIC_IPV4_DNS.us-east-1.compute.amazonaws.com
```

-L signifie l'utilisation de la redirection de port local que vous pouvez utiliser pour spécifier un port local utilisé pour transférer les données vers le port distant identifié sur le serveur Web local du nœud.

3. Entrez `http://localhost:8157/` dans votre navigateur.

Note

Vous devrez peut-être utiliser `https://localhost:8157/`.

Deuxième option : Proxies utilisant la ligne de commande

Vous pouvez utiliser la plupart des navigateurs Web pour configurer des proxys à l'aide d'une ligne de commande ou d'un paramètre de configuration. Par exemple, avec Chromium, vous pouvez démarrer le navigateur avec la commande suivante :

```
chromium --proxy-server="socks5://localhost:8157"
```

Cela démarre une session de navigateur qui utilise le tunnel SSH que vous avez créé lors des étapes précédentes pour transmettre ses demandes par proxy. Vous pouvez ouvrir l'URL de votre environnement Amazon MWAA privé (avec `https ://`) comme suit :

```
https://YOUR_VPC_ENDPOINT_ID-vpce.us-east-1.airflow.amazonaws.com/home.
```

Troisième option : utilisation de proxys FoxyProxy pour Mozilla Firefox

L'exemple suivant illustre une configuration FoxyProxy Standard (version 7.5.1) pour Mozilla Firefox. FoxyProxy fournit un ensemble d'outils de gestion de proxy. Il vous permet d'utiliser un serveur proxy pour faire correspondre URLs les modèles correspondant aux domaines utilisés par l'interface utilisateur d'Apache Airflow.

1. Dans Firefox, ouvrez la page de l'extension [FoxyProxy Standard](#).

2. Choisissez Ajouter à Firefox.
3. Choisissez Ajouter.
4. Choisissez l' FoxyProxy icône dans la barre d'outils de votre navigateur, puis sélectionnez Options.
5. Copiez le code suivant et enregistrez-le localement sous `mwa-proxy.json`. Remplacez la valeur de l'échantillon `YOUR_HOST_NAME` par votre URL Apache Airflow.

```
{
  "e0b7kh1606694837384": {
    "type": 3,
    "color": "#66cc66",
    "title": "airflow",
    "active": true,
    "address": "localhost",
    "port": 8157,
    "proxyDNS": false,
    "username": "",
    "password": "",
    "whitePatterns": [
      {
        "title": "airflow-ui",
        "pattern": "YOUR_HOST_NAME",
        "type": 1,
        "protocols": 1,
        "active": true
      }
    ],
    "blackPatterns": [],
    "pacURL": "",
    "index": -1
  },
  "k20d21508277536715": {
    "active": true,
    "title": "Default",
    "notes": "These are the settings that are used when no patterns match a URL.",
    "color": "#0055E5",
    "type": 5,
    "whitePatterns": [
      {
        "title": "all URLs",
        "active": true,
        "pattern": "*"
      }
    ]
  }
}
```

```
        "type": 1,
        "protocols": 1
      },
    ],
    "blackPatterns": [],
    "index": 9007199254740991
  },
  "logging": {
    "active": true,
    "maxSize": 500
  },
  "mode": "patterns",
  "browserVersion": "82.0.3",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

6. Dans le volet Paramètres d'importation à partir de la FoxyProxy version 6.0, choisissez Paramètres d'importation et sélectionnez le `mwaa-proxy.json` fichier.
7. Choisissez OK.

Sixième étape : ouvrir l'interface utilisateur d'Apache Airflow

Les étapes suivantes décrivent comment ouvrir l'interface utilisateur d'Apache Airflow.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez Open Airflow UI.

Quelle est la prochaine étape ?

- Découvrez comment exécuter les commandes de la CLI Airflow sur un tunnel SSH vers un hôte bastion dans. [Référence des commandes de la CLI Apache Airflow](#)
- Découvrez comment télécharger le code DAG dans votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Tutoriel : Restreindre l'accès d'un utilisateur Amazon MWAA à un sous-ensemble de DAGs

[Amazon MWAA gère l'accès à votre environnement en mappant vos principaux IAM à un ou plusieurs rôles par défaut d'Apache Airflow.](#) Utilisez le didacticiel suivant pour empêcher les utilisateurs individuels d'Amazon MWAA d'accéder et d'interagir uniquement avec un DAG spécifique ou un ensemble de DAGs.

Note

Les étapes de ce didacticiel peuvent être effectuées à l'aide d'un accès fédéré, à condition que les rôles IAM puissent être assumés.

Rubriques

- [Conditions préalables](#)
- [Étape 1 : fournissez un accès au serveur Web Amazon MWAA à votre principal IAM avec le rôle Apache Airflow Public par défaut.](#)
- [Deuxième étape : créer un nouveau rôle personnalisé pour Apache Airflow](#)
- [Troisième étape : attribuer le rôle que vous avez créé à votre utilisateur Amazon MWAA](#)
- [Étapes suivantes](#)
- [Ressources connexes](#)

Conditions préalables

Pour suivre les étapes de ce didacticiel, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA avec plusieurs DAGs](#)
- Un principal IAM, Admin avec des [AdministratorAccess](#) autorisations, et un utilisateur IAMMWAAUser, en tant que principal pour lequel vous pouvez limiter l'accès au DAG. Pour plus d'informations sur les rôles d'administrateur, reportez-vous à la section [Fonction du poste d'administrateur](#) dans le guide de l'utilisateur IAM

Note

N'associez pas de politiques d'autorisation directement à vos utilisateurs IAM. Nous vous recommandons de configurer des rôles IAM que les utilisateurs peuvent assumer pour obtenir un accès temporaire à vos ressources Amazon MWAA.

- [AWS Command Line Interface version 2](#) installée.

Étape 1 : fournissez un accès au serveur Web Amazon MWAA à votre principal IAM avec le rôle Apache Airflow **Public** par défaut.

Pour accorder une autorisation à l'aide du AWS Management Console

1. Connectez-vous à votre compte Compte AWS avec un Admin rôle et ouvrez la [console IAM](#).
2. Dans le volet de navigation de gauche, choisissez Utilisateurs, puis choisissez votre utilisateur Amazon MWAA IAM dans le tableau des utilisateurs.
3. Sur la page des détails de l'utilisateur, sous Résumé, choisissez l'onglet Autorisations, puis choisissez Politiques d'autorisations pour développer la carte et choisissez Ajouter des autorisations.
4. Dans la section Accorder des autorisations, choisissez Joindre directement les politiques existantes, puis choisissez Créer une politique pour créer et joindre votre propre politique d'autorisations personnalisée.
5. Sur la page Créer une politique, choisissez JSON, puis copiez et collez la politique d'autorisation JSON suivante dans l'éditeur de stratégie. Cette politique accorde l'accès au serveur Web à l'utilisateur doté du rôle `Public` Apache Airflow par défaut.

JSON

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

```

        "arn:aws:airflow:us-
east-1:111122223333:role/YOUR_ENVIRONMENT_NAME/Public"
    ]
}
]
}

```

Deuxième étape : créer un nouveau rôle personnalisé pour Apache Airflow

Pour créer un nouveau rôle à l'aide de l'interface utilisateur d'Apache Airflow

1. À l'aide de votre rôle d'administrateur IAM, ouvrez la [console Amazon MWAA](#) et lancez l'interface utilisateur Apache Airflow de votre environnement.
2. Dans le volet de navigation en haut, passez le curseur sur Security pour ouvrir la liste déroulante, puis choisissez List Roles pour accéder aux rôles Apache Airflow par défaut.
3. Dans la liste des rôles, sélectionnez Utilisateur, puis au début de la page, sélectionnez Actions pour ouvrir le menu déroulant. Choisissez Copier le rôle, puis confirmez OK

Note

Copiez les rôles Ops ou Viewer pour accorder plus ou moins d'accès, respectivement.

4. Repérez le nouveau rôle que vous avez créé dans le tableau et choisissez Modifier l'enregistrement.
5. Sur la page Modifier le rôle, procédez comme suit :
 - Pour Nom, entrez un nouveau nom pour le rôle dans le champ de texte. Par exemple, **Restricted**.
 - Pour la liste des autorisations, supprimez can read on DAGs puis can edit on DAGs ajoutez des autorisations de lecture et d'écriture pour l'ensemble auquel DAGs vous souhaitez donner accès. Par exemple, pour un DAG `example_dag.py`, ajoutez **can read on DAG: *example_dag*** et **can edit on DAG: *example_dag***.

Choisissez Enregistrer. Vous avez désormais un nouveau rôle qui limite l'accès à un sous-ensemble des éléments DAGs disponibles dans votre environnement Amazon MWAA. Vous pouvez attribuer ce rôle à n'importe quel utilisateur d'Apache Airflow existant.

Troisième étape : attribuer le rôle que vous avez créé à votre utilisateur Amazon MWAA

Pour attribuer le nouveau rôle

1. À l'aide des informations d'accès pour `MWAAUser`, exécutez la commande CLI suivante pour récupérer l'URL du serveur Web de votre environnement.

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

En cas de succès, vous allez vous référer au résultat suivant :

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. Une `MWAAUser` fois connecté au AWS Management Console, ouvrez une nouvelle fenêtre de navigateur et accédez aux éléments suivants URI. Remplacez `Webserver-URL` par vos informations.

```
https://<Webserver-URL>/home
```

En cas de succès, vous obtiendrez une page Forbidden d'erreur car vous `MWAAUser` n'avez pas encore obtenu l'autorisation d'accéder à l'interface utilisateur d'Apache Airflow.

3. Une fois Admin connecté au AWS Management Console, ouvrez à nouveau la console Amazon MWAA et lancez l'interface utilisateur Apache Airflow de votre environnement.
4. Dans le tableau de bord de l'interface utilisateur, élargissez le menu déroulant Sécurité et, cette fois, choisissez Lister les utilisateurs.
5. Dans le tableau des utilisateurs, recherchez le nouvel utilisateur d'Apache Airflow et choisissez Modifier l'enregistrement. Le prénom de l'utilisateur correspondra à votre nom d'utilisateur IAM selon le schéma suivant : `user/mwaa-user`.
6. Sur la page Modifier un utilisateur, dans la section Rôle, ajoutez le nouveau rôle personnalisé que vous avez créé, puis choisissez Enregistrer.

Note

Le champ Nom de famille est obligatoire, mais un espace répond à cette exigence.

Le `Public` principal IAM accorde l'`MWAAUser` autorisation d'accéder à l'interface utilisateur d'Apache Airflow, tandis que le nouveau rôle fournit les autorisations supplémentaires nécessaires pour obtenir les leurs. DAGs

 Important

Tous les 5 rôles par défaut (tels que `Admin`) non autorisés par IAM ajoutés à l'aide de l'interface utilisateur d'Apache Airflow seront supprimés lors de la prochaine connexion de l'utilisateur.

Étapes suivantes

- Pour en savoir plus sur la gestion de l'accès à votre environnement Amazon MWAA et pour obtenir des exemples de politiques IAM JSON que vous pouvez utiliser pour les utilisateurs de votre environnement, reportez-vous à [the section called “Accès à un environnement Amazon MWAA”](#)

Ressources connexes

- [Contrôle d'accès](#) (documentation Apache Airflow) — Pour en savoir plus sur les rôles par défaut d'Apache Airflow, consultez le site Web de documentation d'Apache Airflow.

Tutoriel : Automatisez la gestion des points de terminaison de votre propre environnement sur Amazon MWAA

Si vous en gérez plusieurs Comptes AWS qui partagent des ressources, Amazon MWAA vous permet de créer et de gérer vos propres points de terminaison Amazon VPC. [AWS Organizations](#) Cela signifie que vous pouvez utiliser des politiques de sécurité plus strictes qui n'autorisent l'accès qu'aux ressources requises par votre environnement.

Lorsque vous créez un environnement dans un Amazon VPC partagé, le compte propriétaire du VPC Amazon principal (propriétaire) partage les deux sous-réseaux privés requis par Amazon MWAA avec d'autres comptes (participants) appartenant à la même organisation. Les comptes

participants qui partagent ces sous-réseaux peuvent ensuite afficher, créer, modifier et supprimer des environnements dans le VPC partagé.

Lorsque vous créez un environnement dans un Amazon VPC partagé ou soumis à des règles, Amazon MWAA crée d'abord les ressources VPC du service, puis entre dans [PENDING](#) un état pendant 72 heures maximum.

Lorsque le statut de l'environnement passe de CREATING à PENDING, Amazon MWAA envoie une EventBridge notification Amazon concernant le changement d'état. Cela permet au compte propriétaire de créer les points de terminaison requis pour le compte des participants en fonction des informations de service des points de terminaison provenant de la console ou de l'API Amazon MWAA, ou par programmation. Dans ce qui suit, nous créons de nouveaux points de terminaison Amazon VPC à l'aide d'une fonction Lambda et d'une EventBridge règle qui écoute les notifications de changement d'état d'Amazon MWAA.

Ici, nous créons les nouveaux points de terminaison dans le même Amazon VPC que l'environnement. Pour configurer un Amazon VPC partagé, créez la EventBridge règle et la fonction Lambda dans le compte du propriétaire, ainsi que l'environnement Amazon MWAA dans le compte du participant.

Rubriques

- [Prérequis](#)
- [Création de l'Amazon VPC](#)
- [Créer la fonction Lambda](#)
- [Créez la EventBridge règle](#)
- [Création de l'environnement Amazon MWAA](#)

Prérequis

Pour effectuer les étapes de ce didacticiel, vous aurez besoin des éléments suivants :

- ...

Création de l'Amazon VPC

Utilisez le CloudFormation modèle et la AWS CLI commande suivants pour créer un nouvel Amazon VPC. Le modèle configure les ressources Amazon VPC et modifie la politique du point de terminaison afin de restreindre l'accès à une file d'attente spécifique.

1. Téléchargez le CloudFormation [modèle](#), puis décompressez le .yaml fichier.
2. Dans une nouvelle fenêtre d'invite de commande, accédez au dossier dans lequel vous avez enregistré le modèle, puis [create-stack](#)utilisez-le pour créer la pile. L'`--template-body`indicateur indique le chemin d'accès au modèle.

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

Dans la section suivante, vous allez créer la fonction Lambda.

Créer la fonction Lambda

Utilisez le code Python et la politique IAM JSON suivants pour créer une nouvelle fonction Lambda et un nouveau rôle d'exécution. Cette fonction crée des points de terminaison Amazon VPC pour un serveur Web Apache Airflow privé et une file d'attente Amazon SQS. Amazon MWAA utilise Amazon SQS pour mettre en file d'attente les tâches avec Celery entre plusieurs travailleurs lors de la mise à l'échelle de votre environnement.

1. Téléchargez le [code de la fonction](#) Python.
2. Téléchargez la [politique d'autorisation](#) IAM, puis décompressez le fichier.
3. Ouvrez une invite de commande, puis accédez au dossier dans lequel vous avez enregistré la politique d'autorisation JSON. Utilisez la [create-role](#)commande IAM pour créer le nouveau rôle.

```
aws iam create-role --role-name function-role \
    --assume-role-policy-document file://lambda-mwaa-vpce-policy.json
```

Notez l'ARN du rôle indiqué dans la AWS CLI réponse. À l'étape suivante, nous spécifierons ce nouveau rôle comme rôle d'exécution de la fonction à l'aide de son ARN.

4. Accédez au dossier dans lequel vous avez enregistré le code de fonction, puis utilisez la [create-function](#)commande pour créer une nouvelle fonction.

```
aws lambda create-function --function-name mwaa-vpce-lambda \  
--zip-file file://mwaa-lambda-shared-vpc.zip --runtime python3.8 --role  
arn:aws:iam::123456789012:role/function-role --handler lambda_handler
```

Notez la fonction ARN figurant dans la AWS CLI réponse. À l'étape suivante, nous spécifierons l'ARN pour configurer la fonction en tant que cible pour une nouvelle EventBridge règle.

Dans la section suivante, vous allez créer la EventBridge règle qui invoque cette fonction lorsque l'environnement entre dans un PENDING état.

Créez la EventBridge règle

Procédez comme suit pour créer une nouvelle règle qui écoute les notifications Amazon MWAA et cible votre nouvelle fonction Lambda.

1. Utilisez la EventBridge `put-rule` commande pour créer une nouvelle EventBridge règle.

```
aws events put-rule --name "mwaa-lambda-rule" \  
--event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment  
Status Change\"]}"
```

Le modèle d'événement écoute les notifications qu'Amazon MWAA envoie chaque fois que l'état de l'environnement change.

```
{  
  "source": ["aws.airflow"],  
  "detail-type": ["MWAA Environment Status Change"]  
}
```

2. Utilisez la `put-targets` commande pour ajouter la fonction Lambda comme cible pour la nouvelle règle.

```
aws events put-targets --rule "mwaa-lambda-rule" \  
--targets "Id"="1", "Arn"="arn:aws:lambda:us-east-1:123456789012:function:mwaa-vpce-  
lambda"
```

Vous êtes prêt à créer un nouvel environnement Amazon MWAA avec des points de terminaison Amazon VPC gérés par le client.

Création de l'environnement Amazon MWAA

Utilisez la console Amazon MWAA pour créer un nouvel environnement avec des points de terminaison Amazon VPC gérés par le client.

1. Ouvrez la console [Amazon MWAA](#) et choisissez Create an environment.
2. Pour Nom, entrez un nom unique.
3. Pour la version Airflow, choisissez la dernière version.
4. Choisissez un compartiment et un DAGs dossier Amazon S3, par exemple dags/ à utiliser avec l'environnement, puis choisissez Next.
5. Sur la page Configurer les paramètres avancés, procédez comme suit :
 - a. Pour Virtual Private Cloud, choisissez le VPC Amazon que vous avez créé à l'étape [précédente](#).
 - b. Pour accéder au serveur Web, choisissez Réseau public (accessible à Internet).
 - c. Pour les groupes de sécurité, choisissez le groupe de sécurité que vous avez créé avec CloudFormation. Étant donné que les groupes de sécurité pour les AWS PrivateLink points de terminaison de l'étape précédente sont autoréférencés, vous devez choisir le même groupe de sécurité pour votre environnement.
 - d. Pour la gestion des terminaux, choisissez les points de terminaison gérés par le client.
6. Conservez les paramètres par défaut restants, puis choisissez Next.
7. Passez en revue vos sélections, puis choisissez Créer un environnement.

Tip

Pour plus d'informations sur la configuration d'un nouvel environnement, consultez [Getting started with Amazon MWAA](#).

Lorsque l'environnement est PENDING, Amazon MWAA envoie une notification correspondant au modèle d'événement que vous avez défini pour votre règle. La règle invoque votre fonction Lambda. La fonction analyse l'événement de notification et obtient les informations de point de

terminaison requises pour le serveur Web et la file d'attente Amazon SQS. Il crée ensuite les points de terminaison dans votre Amazon VPC.

Lorsque les points de terminaison sont disponibles, Amazon MWAA reprend la création de votre environnement. Lorsque vous êtes prêt, l'état de l'environnement devient AVAILABLE et vous pouvez accéder au serveur Web Apache Airflow à l'aide de la console Amazon MWAA.

Exemples de code pour Amazon Managed Workflows pour Apache Airflow

Ce guide contient des exemples de code, y compris DAGs des plugins personnalisés, que vous pouvez utiliser dans un environnement Amazon Managed Workflows for Apache Airflow. Pour d'autres exemples d'utilisation d'Apache Airflow avec AWS des services, reportez-vous au [dags](#) répertoire du référentiel Apache GitHub Airflow.

Exemples

- [Utilisation d'un DAG pour importer des variables dans la CLI](#)
- [Création d'une connexion SSH à l'aide du SSHOperator](#)
- [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow Snowflake](#)
- [Utilisation d'un DAG pour écrire des métriques personnalisées dans CloudWatch](#)
- [Nettoyage de base de données Aurora PostgreSQL dans un environnement Amazon MWAA](#)
- [Exportation des métadonnées de l'environnement vers des fichiers CSV sur Amazon S3](#)
- [Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow](#)
- [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow](#)
- [Création d'un plugin personnalisé avec Oracle](#)
- [Modifier le fuseau horaire d'un DAG sur Amazon MWAA](#)
- [Actualisation d'un CodeArtifact jeton](#)
- [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#)
- [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#)
- [Invocation DAGs avec une fonction Lambda](#)
- [Invocation DAGs dans différents environnements Amazon MWAA](#)
- [Utilisation d'Amazon MWAA avec Amazon RDS pour Microsoft SQL Server](#)
- [Utilisation d'Amazon MWAA avec Amazon EKS](#)
- [Connexion à Amazon ECS à l'aide du ECSOperator](#)
- [Utiliser dbt avec Amazon MWAA](#)
- [AWS blogs et tutoriels](#)

Utilisation d'un DAG pour importer des variables dans la CLI

L'exemple de code suivant importe des variables à l'aide de la CLI sur Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Dépendances](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Autorisations

Vos Compte AWS besoins ont accès à la AmazonMWAAirflowCliAccess politique. Pour en savoir plus, reportez-vous à [Politique de la CLI Apache Airflow : Amazon MWAAirflow CliAccess](#).

Dépendances

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

L'exemple de code suivant utilise trois entrées : le nom de votre environnement Amazon MWAA (inmwaa_env), le nom Région AWS de votre environnement (inaws_region) et le fichier local contenant les variables que vous souhaitez importer (invar_file).

```
import boto3
import json
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwaa_env=''
aws_region=''
var_file=''

try:
    opts, args = getopt.getopt(argv, 'e:v:r:', ['environment', 'variable-
file','region'])
    #if len(opts) == 0 and len(opts) > 3:
    if len(opts) != 3:
        print ('Usage: -e MWAA environment -v variable file location and filename -r
aws region')
    else:
        for opt, arg in opts:
            if opt in ("-e"):
                mwaa_env=arg
            elif opt in ("-r"):
                aws_region=arg
            elif opt in ("-v"):
                var_file=arg

    boto3.setup_default_session(region_name="{}".format(aws_region))
    mwaa_env_name = "{}".format(mwaa_env)

    client = boto3.client('mwaa')
    mwaa_cli_token = client.create_cli_token(
        Name=mwaa_env_name
    )

    with open ("{}".format(var_file), "r") as myfile:
```



```

        fileconf = myfile.read().replace('\n', '')

    json_dictionary = json.loads(fileconf)
    for key in json_dictionary:
        print(key, " ", json_dictionary[key])
        val = (key + " " + json_dictionary[key])
        mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
        mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
        raw_data = "variables set {0}".format(val)
        mwaa_response = requests.post(
            mwaa_webserver_hostname,
            headers={
                'Authorization': mwaa_auth_token,
                'Content-Type': 'text/plain'
            },
            data=raw_data
        )
        mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
        mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')
        print(mwaa_response.status_code)
        print(mwaa_std_err_message)
        print(mwaa_std_out_message)

except:
    print('Use this script with the following options: -e MWAA environment -v variable
file location and filename -r aws region')
    print("Unexpected error:", sys.exc_info()[0])
    sys.exit(2)

```

Quelle est la prochaine étape ?

- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Création d'une connexion SSH à l'aide du **SSHOperator**

L'exemple suivant décrit comment vous pouvez utiliser le DAG (SSHOperator in a directed acyclic graph) pour vous connecter à une EC2 instance Amazon distante depuis votre environnement

Amazon Managed Workflows for Apache Airflow. Vous pouvez utiliser une approche similaire pour vous connecter à n'importe quelle instance distante avec un accès SSH.

Dans l'exemple suivant, vous chargez une clé secrète SSH (.pem) dans le dags répertoire de votre environnement sur Amazon S3. Ensuite, vous installez les dépendances nécessaires en utilisant `requirements.txt` et en créant une nouvelle connexion Apache Airflow dans l'interface utilisateur. Enfin, vous écrivez un DAG qui crée une connexion SSH avec l'instance distante.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Copiez votre clé secrète sur Amazon S3](#)
- [Création d'une nouvelle connexion Apache Airflow](#)
- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- Une clé secrète SSH. L'exemple de code suppose que vous disposez d'une EC2 instance Amazon et d'une instance .pem située dans la même région que votre environnement Amazon MWAA. Si vous n'avez pas de clé, reportez-vous à la section [Créer ou importer une paire de clés](#) dans le guide de EC2 l'utilisateur Amazon.

Autorisations

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Prérequis

Ajoutez le paramètre suivant pour `requirements.txt` installer le `apache-airflow-providers-ssh` package sur le serveur Web. Une fois que votre environnement est mis à jour et qu'Amazon MWAA a correctement installé la dépendance, vous obtiendrez un nouveau type de connexion SSH dans l'interface utilisateur.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
apache-airflow-providers-ssh
```

Note

-c définit l'URL des contraintes dans `requirements.txt`. Cela garantit qu'Amazon MWAA installe la version de package adaptée à votre environnement.

Copiez votre clé secrète sur Amazon S3

Utilisez la AWS Command Line Interface commande suivante pour copier votre `.pem` clé dans le `dags` répertoire de votre environnement dans Amazon S3.

```
aws s3 cp your-secret-key.pem s3://amzn-s3-demo-bucket/dags/
```

Amazon MWAA copie le contenu `dags`, y compris la `.pem` clé, dans le `/usr/local/airflow/dags/` répertoire local. Apache Airflow peut ainsi accéder à la clé.

Création d'une nouvelle connexion Apache Airflow

Pour créer une nouvelle connexion SSH à l'aide de l'interface utilisateur d'Apache Airflow

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Dans la liste des environnements, choisissez Open Airflow UI pour votre environnement.
3. Sur la page de l'interface utilisateur d'Apache Airflow, choisissez Admin dans la barre de navigation principale pour développer la liste déroulante, puis sélectionnez Connexions.
4. Sur la page Lister les connexions, choisissez + ou le bouton Ajouter un nouvel enregistrement pour ajouter une nouvelle connexion.
5. Sur la page Ajouter une connexion, ajoutez les informations suivantes :

- a. Pour ID de connexion, entrez **ssh_new**.
- b. Pour Type de connexion, choisissez SSH dans la liste déroulante.

 Note

Si le type de connexion SSH n'est pas disponible dans la liste, Amazon MWAA n'a pas installé le package requis. `apache-airflow-providers-ssh` Mettez à jour votre `requirements.txt` fichier pour inclure ce package, puis réessayez.

- c. Pour Host, entrez l'adresse IP de l' EC2 instance Amazon à laquelle vous souhaitez vous connecter. Par exemple, **12.345.67.89**.
- d. Dans Nom d'utilisateur, saisissez **ec2-user** si vous vous connectez à une EC2 instance Amazon. Votre nom d'utilisateur peut être différent en fonction du type d'instance distante à laquelle vous souhaitez qu'Apache Airflow se connecte.
- e. Pour Extra, entrez la paire clé-valeur suivante au format JSON :

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

Cette paire clé-valeur indique à Apache Airflow de rechercher la clé secrète dans le répertoire local. `/dags`

Exemple de code

Le DAG suivant utilise le `SSHOperator` pour se connecter à votre EC2 instance Amazon cible, puis exécute la commande `hostname` Linux pour imprimer le nom de l'instance. Vous pouvez modifier le DAG pour exécuter n'importe quelle commande ou script sur l'instance distante.

1. Ouvrez un terminal et naviguez jusqu'au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. En cas de succès, vous obtiendrez un résultat similaire à ce qui suit dans les journaux des `ssh_task` tâches du `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
```

Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow Snowflake

Les exemples d'appels suivants AWS Secrets Manager pour obtenir une clé secrète pour une connexion Apache Airflow Snowflake sur Amazon Managed Workflows for Apache Airflow. Cela suppose que vous avez terminé les étapes de [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- Une chaîne de connexion Apache Airflow dans Secrets Manager, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)

Autorisations

- Autorisations du Gestionnaire de Secrets Manager répertoriées dans [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
apache-airflow-providers-snowflake==1.3.0
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui appelle Secrets Manager pour obtenir le secret.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement `soussnowflake_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",
    )
```

Quelle est la prochaine étape ?

- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Utilisation d'un DAG pour écrire des métriques personnalisées dans CloudWatch

Vous pouvez utiliser l'exemple de code suivant pour écrire un graphe acyclique dirigé (DAG) qui exécute un `PythonOperator` afin de récupérer des métriques au niveau du système d'exploitation pour un environnement Amazon MWAA. Le DAG publie ensuite les données sous forme de métriques personnalisées sur Amazon CloudWatch.

Les métriques personnalisées au niveau du système d'exploitation vous offrent une visibilité supplémentaire sur la manière dont les employés de votre environnement utilisent les ressources telles que la mémoire virtuelle et le processeur. Vous pouvez utiliser ces informations pour sélectionner la [classe d'environnement](#) la mieux adaptée à votre charge de travail.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Dépendances](#)

- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous avez besoin des éléments suivants :

- Un [environnement Amazon MWA](#).

Autorisations

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Dépendances

- Aucune dépendance supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Exemple de code

1. Dans votre invite de commande, accédez au dossier dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous le nom `dag-custom-metrics.py`. `MWAA-ENV-NAME` Remplacez-le par le nom de votre environnement.

```
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("MWA_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='MWA-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. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si le DAG fonctionne correctement, vous obtenez quelque chose de similaire à ce qui suit dans vos journaux Apache Airflow :

```

[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO -
cpu_stats scpustats(ctx_switches=3253992384, interrupts=1964237163,
soft_interrupts=492328209, syscalls=0)
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
16024199168.0 to metric virtual_memory_total

```

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
{'RequestId': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d', 'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amzn-requestid': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d',
'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
14356287488.0 to metric virtual_memory_available
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
{'RequestId': '6ef60085-07ab-4865-8abf-dc94f90cab46', 'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amzn-requestid': '6ef60085-07ab-4865-8abf-dc94f90cab46',
'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
1342296064.0 to metric virtual_memory_used
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
{'RequestId': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00', 'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amzn-requestid': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00',
'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
17:54:45 GMT'}, 'RetryAttempts': 0}}
...
[2022-08-16, 10:54:46 UTC] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-08-16, 10:54:46 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
dag_id=dag-custom-metrics, task_id=memory_test, execution_date=20220816T175444,
start_date=20220816T175445, end_date=20220816T175446
[2022-08-16, 10:54:46 UTC] {{local_task_job.py:154}} INFO - Task exited with return
code 0
```

Nettoyage de base de données Aurora PostgreSQL dans un environnement Amazon MWAA

Amazon Managed Workflows for Apache Airflow utilise une base de données Aurora PostgreSQL comme base de données de métadonnées Apache Airflow, dans laquelle le DAG s'exécute et les instances de tâches sont stockées. L'exemple de code suivant efface régulièrement les entrées de la base de données Aurora PostgreSQL dédiée à votre environnement Amazon MWAA.

Rubriques

- [Version](#)
- [Conditions préalables](#)
- [Dépendances](#)

- [Exemple de code](#)

Version

Les exemples de code présentés sur cette page sont spécifiques à Apache Airflow v2 pris en charge sur Amazon MWAA. Reportez-vous aux versions d'[Apache Airflow prises en charge](#).

Tip

Pour les utilisateurs d'Apache Airflow v3 : si vous souhaitez nettoyer une base de données (purger les anciens enregistrements des tables de métastore), exécutez la commande CLI. [db clean](#)

Conditions préalables

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Dépendances

Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

Le DAG suivant nettoie la base de données de métadonnées pour les tables spécifiées dans `TABLES_TO_CLEAN`. L'exemple supprime les données des tables spécifiées datant de plus de 30 jours. Pour ajuster la date à laquelle les entrées ont été supprimées, définissez `MAX_AGE_IN_DAYS` une valeur différente.

Apache Airflow v2.4 to 2.10.3

```
from airflow import DAG
from airflow.models.param import Param
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago
```

```

from datetime import datetime, timedelta

# Note: Database commands might time out if running longer than 5 minutes. If this
# occurs, please increase the MAX_AGE_IN_DAYS (or change
# timestamp parameter to an earlier date) for initial runs, then reduce on
# subsequent runs until the desired retention is met.

MAX_AGE_IN_DAYS = 30

# To clean specific tables, please provide a comma-separated list per
# https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-
# ref.html#clean
# A value of None will clean all tables

TABLES_TO_CLEAN = None

with DAG(
    dag_id="clean_db_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1),
    params={
        "timestamp": Param(
            default=(datetime.now()-timedelta(days=MAX_AGE_IN_DAYS)).strftime("%Y-
%m-%d %H:%M:%S"),
            type="string",
            minLength=1,
            maxLength=255,
        ),
    }
) as dag:
    if TABLES_TO_CLEAN:
        bash_command="airflow db clean --clean-before-timestamp
'{{ params.timestamp }}" --tables '"+TABLES_TO_CLEAN+" ' --skip-archive --yes"
    else:
        bash_command="airflow db clean --clean-before-timestamp
'{{ params.timestamp }}" --skip-archive --yes"

    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=bash_command
    )

```

Apache Airflow v2.2 and earlier

```

from airflow import settings
from airflow.utils.dates import days_ago
from airflow.models import DagTag, DagModel, DagRun, ImportError, Log, SlaMiss,
    RenderedTaskInstanceFields, TaskInstance, TaskReschedule, XCom
from airflow.decorators import dag, task
from airflow.utils.dates import days_ago
from time import sleep

from airflow.version import version
major_version, minor_version = int(version.split('.')[0]), int(version.split('.')[1])
if major_version >= 2 and minor_version >= 6:
    from airflow.jobs.job import Job
else:
    # The BaseJob class was renamed as of Apache Airflow v2.6
    from airflow.jobs.base_job import BaseJob as Job

# Delete entries for the past 30 days. Adjust MAX_AGE_IN_DAYS to set how far back
# this DAG cleans the database.
MAX_AGE_IN_DAYS = 30
MIN_AGE_IN_DAYS = 0
DECREMENT = -7

# This is a list of (table, time) tuples.
# table = the table to clean in the metadata database
# time = the column in the table associated to the timestamp of an entry
# or None if not applicable.
TABLES_TO_CLEAN = [[Job, Job.latest_heartbeat],
    [TaskInstance, TaskInstance.execution_date],
    [TaskReschedule, TaskReschedule.execution_date],
    [DagTag, None],
    [DagModel, DagModel.last_parsed_time],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [XCom, XCom.execution_date],
]

@task()
def cleanup_db_fn(x):

```

```

session = settings.Session()

if x[1]:
    for oldest_days_ago in range(MAX_AGE_IN_DAYS, MIN_AGE_IN_DAYS, DECREMENT):
        earliest_days_ago = max(oldest_days_ago + DECREMENT, MIN_AGE_IN_DAYS)
        print(f"deleting {str(x[0])} entries between {earliest_days_ago} and
{oldest_days_ago} days old...")
        earliest_date = days_ago(earliest_days_ago)
        oldest_date = days_ago(oldest_days_ago)
        query = session.query(x[0]).filter(x[1] >= earliest_date).filter(x[1] <=
oldest_date)
        query.delete(synchronize_session= False)
        session.commit()
        sleep(5)
    else:
        # No time column specified for the table. Delete all entries
        print("deleting", str(x[0]), "...")
        query = session.query(x[0])
        query.delete(synchronize_session= False)
        session.commit()

session.close()

@dag(
    dag_id="cleanup_db",
    schedule_interval="@weekly",
    start_date=days_ago(7),
    catchup=False,
    is_paused_upon_creation=False
)

def clean_db_dag_fn():
    t_last=None
    for x in TABLES_TO_CLEAN:
        t=cleanup_db_fn(x)
        if t_last:
            t_last >> t
        t_last = t

clean_db_dag = clean_db_dag_fn()

```


Exportation des métadonnées de l'environnement vers des fichiers CSV sur Amazon S3

Utilisez l'exemple de code suivant pour créer un graphe acyclique dirigé (DAG) qui interroge la base de données pour obtenir une série d'informations d'exécution du DAG et écrit les données dans des .csv fichiers stockés sur Amazon S3.

Vous souhaitez peut-être exporter des informations depuis la base de données Aurora PostgreSQL de votre environnement pour inspecter les données localement, les archiver dans un stockage d'objets ou les combiner avec des outils tels que l'opérateur Amazon [S3 vers Amazon Redshift](#) et le [nettoyage de la base de données, afin de déplacer les](#) métadonnées Amazon MWAA hors de l'environnement, tout en les préservant pour une analyse future.

Vous pouvez interroger la base de données pour n'importe quel objet répertorié dans les modèles [Apache Airflow](#). Cet exemple de code utilise trois modèles, `DagRun`, et `TaskFailTaskInstance`, qui fournissent des informations relatives aux exécutions de To DAG.

Rubriques

- [Version](#)
- [Conditions préalables](#)
- [Permissions](#)
- [Exigences](#)
- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- Un [nouveau compartiment Amazon S3](#) dans lequel vous souhaitez exporter vos informations de métadonnées.

Permissions

Amazon MWAA a besoin d'une autorisation pour que l'action Amazon S3 puisse `s3:PutObject` écrire les informations de métadonnées demandées dans Amazon S3. Ajoutez la déclaration de politique suivante au rôle d'exécution de votre environnement.

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
}
```

Cette politique limite l'accès en écriture uniquement à *amzn-s3-demo-bucket*.

Exigences

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

Les étapes suivantes décrivent comment créer un DAG qui interroge Aurora PostgreSQL et écrit le résultat dans votre nouveau compartiment Amazon S3.

1. Dans votre terminal, accédez au répertoire dans lequel votre code DAG est enregistré. Par exemple :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous le nom `demetadata_to_csv.py`. Vous pouvez modifier la valeur attribuée `MAX_AGE_IN_DAYS` à pour contrôler l'âge des enregistrements les plus anciens que votre DAG interroge dans la base de données de métadonnées.

```
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.end_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. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. En cas de réussite, vous obtiendrez un résultat similaire à ce qui suit dans les journaux des tâches associées à la export_db tâche :

```

[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

```

Vous pouvez désormais accéder aux .csv fichiers exportés et les télécharger dans votre nouveau compartiment Amazon S3 dans /files/export/.

Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow

Les exemples d'appels suivants AWS Secrets Manager pour obtenir une clé secrète pour une variable Apache Airflow sur Amazon Managed Workflows for Apache Airflow. Cela suppose que vous avez terminé les étapes de [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- Chaîne de variable Apache Airflow dans Secrets Manager, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)

Autorisations

- Autorisations du Gestionnaire de Secrets Manager répertoriées dans [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Prérequis

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui appelle Secrets Manager pour obtenir le secret.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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
    )
```

Quelle est la prochaine étape ?

- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow

Les exemples d'appels suivants AWS Secrets Manager pour obtenir une clé secrète pour une connexion Apache Airflow sur Amazon Managed Workflows for Apache Airflow. Cela suppose que vous avez terminé les étapes de [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- Une chaîne de connexion Apache Airflow dans Secrets Manager, comme indiqué dans. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)

Autorisations

- Autorisations du Gestionnaire de Secrets Manager, telles qu'elles sont répertoriées dans [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#).

Prérequis

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui appelle Secrets Manager pour obtenir le secret.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `secrets-manager.py`.

```
"""  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
  
Permission is hereby granted, free of charge, to any person obtaining a copy of  
this software and associated documentation files (the "Software"), to deal in  
the Software without restriction, including without limitation the rights to
```


use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.

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

"""

```
from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook
```

```
from datetime import timedelta
import os
```

```
### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'
```

```
default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}
```

```
### Gets the secret myconn from Secrets Manager
```

```
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type(region_name='us-east-1')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString
```

```
### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
```

```
with DAG(
    dag_id=os.path.basename(__file__).replace(".py", ""),
    default_args=default_args,
```

```
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Quelle est la prochaine étape ?

- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).

Création d'un plugin personnalisé avec Oracle

L'exemple suivant explique les étapes de création d'un plug-in personnalisé à l'aide d'Oracle pour Amazon MWAA et peut être combiné avec d'autres plug-ins et binaires personnalisés dans votre fichier plugins.zip.

Table des matières

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code](#)
- [Créez le plugin personnalisé](#)
 - [Dépendances de téléchargement](#)
 - [Plug-in personnalisé](#)
 - [Plugins.zip](#)
- [Options de configuration d'Airflow](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- La journalisation des travailleurs est activée à n'importe quel niveau de journal, CRITICAL ou dans la section précédente pour votre environnement. Pour plus d'informations sur les types de journaux Amazon MWAA et sur la façon de gérer vos groupes de journaux, consultez [the section called "Accès aux journaux Airflow"](#)

Autorisations

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_oracle
apache-airflow-providers-oracle
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui testera le plugin personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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
    )
```

Créez le plugin personnalisé

Cette section décrit comment télécharger les dépendances, créer le plugin personnalisé et le fichier `plugins.zip`.

Dépendances de téléchargement

Amazon MWAA extraira le contenu du fichier `plugins.zip` dans `/usr/local/airflow/plugins` chaque planificateur et conteneur de travail Amazon MWAA. Ceci est utilisé pour ajouter des fichiers binaires à votre environnement. Les étapes suivantes décrivent comment assembler les fichiers nécessaires au plugin personnalisé.

Extraire l'image du conteneur Amazon Linux

1. Dans votre invite de commande, extrayez l'image du conteneur Amazon Linux et exécutez le conteneur localement. Exemples :

```
docker pull amazonlinux
docker run -it amazonlinux:latest /bin/bash
```

Votre invite de commande peut appeler une ligne de commande bash. Exemples :

```
bash-4.2#
```

2. Installez la fonction asynchrone I/O native pour Linux (libaio).

```
yum -y install libaio
```

3. Gardez cette fenêtre ouverte pour les étapes suivantes. Nous copierons les fichiers suivants localement : `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1`.

Télécharger le dossier client

1. Installez le package de décompression localement. Exemples :

```
sudo yum install unzip
```

2. Créez un répertoire `oracle_plugin`. Exemples :

```
mkdir oracle_plugin
cd oracle_plugin
```

3. Utilisez la commande `curl` suivante pour télécharger le [instantclient-basic-linuxfichier .x64-18.5.0.0.0dbru.zip](#) depuis [Oracle](#) Instant Client Downloads pour Linux x86-64 (64 bits).

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. Décompressez le fichier `client.zip`. Exemples :

```
unzip *.zip
```

Extraire des fichiers depuis Docker

1. Dans une nouvelle invite de commande, affichez et notez votre identifiant de conteneur Docker. Exemples :

```
docker container ls
```

Votre invite de commande peut renvoyer tous les conteneurs et leurs IDs. Exemples :

```
debc16fd6970
```

2. Dans votre `oracle_plugin` répertoire, extrayez les `lib64/libaio.so.1.0.1` fichiers `lib64/libaio.so.1``lib64/libaio.so.1.0.0`, dans le `instantclient_18_5` dossier local. Exemples :

```
docker cp debc16fd6970:/lib64/libaio.so.1 instantclient_18_5/  
docker cp debc16fd6970:/lib64/libaio.so.1.0.0 instantclient_18_5/  
docker cp debc16fd6970:/lib64/libaio.so.1.0.1 instantclient_18_5/
```

Plug-in personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier des plugins au démarrage. Ceci est utilisé pour définir et modifier les variables d'environnement. Les étapes suivantes décrivent l'exemple de code du plugin personnalisé.

- Copiez le contenu de l'exemple de code suivant et enregistrez-le localement `sousenv_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

Les étapes suivantes expliquent comment créer `leplugins.zip`. Le contenu de cet exemple peut être combiné avec vos autres plugins et fichiers binaires dans un seul `plugins.zip` fichier.

Compressez le contenu du répertoire du plugin

1. Dans votre invite de commande, accédez au `oracle_plugin` répertoire. Exemples :

```
cd oracle_plugin
```

2. Compressez le `instantclient_18_5` répertoire dans le fichier `plugins.zip`. Exemples :

```
zip -r ../plugins.zip ./
```

Votre invite de commande affiche :

```
oracle_plugin$ ls  
client.zip  instantclient_18_5
```

3. Supprimez le `client.zip` fichier. Exemples :

```
rm client.zip
```

Compressez le fichier `env_var_plugin_oracle.py`

1. Ajoutez le `env_var_plugin_oracle.py` fichier à la racine du fichier `plugins.zip`. Exemples :

```
zip plugins.zip env_var_plugin_oracle.py
```

2. Votre fichier `plugins.zip` inclut désormais les éléments suivants :

```
env_var_plugin_oracle.py  
instantclient_18_5/
```

Options de configuration d'Airflow

Si vous utilisez Apache Airflow v2, ajoutez-le en `core.lazy_load_plugins : False` tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, reportez-vous à la section [Utilisation des options de configuration pour charger des plug-ins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).
- Découvrez comment charger le `plugins.zip` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Modifier le fuseau horaire d'un DAG sur Amazon MWAA

Apache Airflow planifie votre graphe acyclique dirigé (DAG) en UTC+0 par défaut. [Les étapes suivantes expliquent comment modifier le fuseau horaire dans lequel Amazon MWAA exécute votre appareil DAGs avec Pendulum](#). Cette rubrique explique éventuellement comment créer un plugin personnalisé pour modifier le fuseau horaire des journaux Apache Airflow de votre environnement.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Permissions](#)
- [Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow](#)
- [Créer une plugins.zip](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Permissions

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow

Apache Airflow exécute les fichiers Python du plugins répertoire au démarrage. Avec le plugin suivant, vous pouvez remplacer le fuseau horaire de l'exécuteur, qui modifie le fuseau horaire dans lequel Apache Airflow écrit les logs.

1. Créez un répertoire portant le nom `plugins` de votre plugin personnalisé, puis naviguez jusqu'au répertoire. Par exemple :

```
$ mkdir plugins
$ cd plugins
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement `dag-timezone-plugin.py` dans le `plugins` dossier.

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. Dans le `plugins` répertoire, créez un fichier Python vide nommé `__init__.py`. Votre `plugins` répertoire doit être similaire au suivant :

```
plugins/
|-- __init__.py
|-- dag-timezone-plugin.py
```

Créer une `plugins.zip`

Les étapes suivantes expliquent comment créer `plugins.zip`. Le contenu de cet exemple peut être combiné avec d'autres plugins et binaires dans un seul `plugins.zip` fichier.

1. Dans votre invite de commande, accédez au `plugins` répertoire de l'étape précédente. Par exemple :

```
cd plugins
```

2. Comprimez le contenu dans votre `plugins` répertoire.

```
zip -r ../plugins.zip ./
```

3. Téléchargez `plugins.zip` dans votre compartiment S3

```
aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

Exemple de code

Pour modifier le fuseau horaire par défaut (UTC+0) dans lequel le DAG s'exécute, nous utiliserons une bibliothèque appelée [Pendulum](#), une bibliothèque Python permettant de travailler avec des date/heure tenant compte du fuseau horaire.

1. Dans votre invite de commande, accédez au répertoire dans lequel vous DAGs êtes enregistré. Par exemple :

```
cd dags
```

2. Copiez le contenu de l'exemple suivant et enregistrez-le `soustz-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. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. En cas de succès, vous obtiendrez un résultat similaire à ce qui suit dans les journaux des tâches du tz_test DAG : tz_aware_task

```
[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
```

Quelle est la prochaine étape ?

- Découvrez comment charger le `plugins.zip` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Actualisation d'un CodeArtifact jeton

Si vous utilisez CodeArtifact pour installer des dépendances Python, Amazon MWAA a besoin d'un jeton actif. Pour autoriser Amazon MWAA à accéder à un CodeArtifact référentiel lors de l'exécution, vous pouvez utiliser un [script de démarrage](#) et le définir [PIP_EXTRA_INDEX_URL](#) avec le jeton.

La rubrique suivante décrit comment créer un script de démarrage qui utilise l'opération [get_authorization_token](#) CodeArtifact API pour récupérer un nouveau jeton chaque fois que votre environnement démarre ou se met à jour.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- Un [CodeArtifact référentiel dans](#) lequel vous stockez les dépendances de votre environnement.

Autorisations

Pour actualiser le CodeArtifact jeton et écrire le résultat dans Amazon S3, Amazon MWAA doit disposer des autorisations suivantes dans le rôle d'exécution.

- L'`codeartifact:GetAuthorizationToken` action permet à Amazon MWAA de récupérer un nouveau jeton auprès de CodeArtifact. La politique suivante accorde l'autorisation pour chaque CodeArtifact domaine que vous créez. Vous pouvez restreindre davantage l'accès à vos domaines

en modifiant la valeur de la ressource dans l'instruction et en spécifiant uniquement les domaines auxquels vous souhaitez que votre environnement accède.

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- L'`sts:GetServiceBearerToken` action est requise pour appeler l'opération `CodeArtifactGetAuthorizationToken` API. Cette opération renvoie un jeton qui doit être utilisé lors de l'utilisation d'un gestionnaire de packages tel que `pip` with `CodeArtifact`. Pour utiliser un gestionnaire de packages avec un `CodeArtifact` référentiel, le rôle d'exécution de votre environnement doit être autorisé, `sts:GetServiceBearerToken` comme indiqué dans la déclaration de politique suivante.

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

Exemple de code

Les étapes suivantes décrivent comment créer un script de démarrage qui met à jour le `CodeArtifact` jeton.

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement `souscode_artifact_startup_script.sh`.

```
#!/bin/sh

# Startup script for MWAA, refer to https://docs.aws.amazon.com/mwaa/latest/
# userguide/using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
```

```
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION
--domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION
--domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"
echo "CodeArtifact startup setup complete"
```

2. Accédez au dossier dans lequel vous avez enregistré le script. Utilisez-le dans une nouvelle fenêtre d'invite pour télécharger le script dans votre compartiment. Remplacez *amzn-s3-demo-bucket* par vos informations.

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

En cas de succès, Amazon S3 affiche le chemin URL de l'objet :

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

Une fois le script chargé, votre environnement le met à jour et l'exécute au démarrage.

Quelle est la prochaine étape ?

- Découvrez comment utiliser des scripts de démarrage pour personnaliser votre environnement dans [the section called “Utilisation d'un script de démarrage”](#).
- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).
- Découvrez comment charger le plugins.zip fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Création d'un plugin personnalisé avec Apache Hive et Hadoop

Amazon MWAA extrait le contenu d'un `plugins.zip` à `/usr/local/airflow/plugins`. Cela peut être utilisé pour ajouter des fichiers binaires à vos conteneurs. En outre, Apache Airflow exécute le contenu des fichiers Python contenus dans le `plugins` dossier au démarrage, ce qui vous permet de définir et de modifier des variables d'environnement. L'exemple suivant explique les étapes de création d'un plugin personnalisé à l'aide d'Apache Hive et Hadoop dans un environnement Amazon Managed Workflows for Apache Airflow et peut être combiné avec d'autres plugins et binaires personnalisés.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Dépendances de téléchargement](#)
- [Plug-in personnalisé](#)
- [Plugins.zip](#)
- [Exemple de code](#)
- [Options de configuration d'Airflow](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Autorisations

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Dépendances de téléchargement

Amazon MWAA extraira le contenu du fichier `plugins.zip` dans `/usr/local/airflow/plugins` chaque planificateur et conteneur de travail Amazon MWAA. Ceci est utilisé pour ajouter des fichiers binaires à votre environnement. Les étapes suivantes décrivent comment assembler les fichiers nécessaires au plugin personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel vous souhaitez créer votre plugin. Exemples :

```
cd plugins
```

2. Téléchargez [Hadoop](#) depuis un [miroir](#), par exemple :

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

3. Téléchargez [Hive](#) depuis un [miroir](#), par exemple :

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

4. Créez un répertoire. Exemples :

```
mkdir hive_plugin
```

5. Extrayez Hadoop.


```
tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
```

6. Extrayez Hive.

```
tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
```

Plug-in personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier des plugins au démarrage. Ceci est utilisé pour définir et modifier les variables d'environnement. Les étapes suivantes décrivent l'exemple de code du plugin personnalisé.

1. Dans votre invite de commande, accédez au `hive_plugin` répertoire. Exemples :

```
cd hive_plugin
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement `hive_plugin.py` dans le `hive_plugin` répertoire.

```
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. Copiez le contenu du texte suivant et enregistrez-le localement `.airflowignore` dans le `hive_plugin` répertoire.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

Les étapes suivantes expliquent comment créer `plugins.zip`. Le contenu de cet exemple peut être combiné avec d'autres plugins et binaires dans un seul `plugins.zip` fichier.

1. Dans votre invite de commande, accédez au `hive_plugin` répertoire de l'étape précédente.

Exemples :

```
cd hive_plugin
```

2. Comprimez le contenu de votre `plugins` dossier.

```
zip -r ../hive_plugin.zip ./
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui testera le plugin personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.

Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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'
    )
```

Options de configuration d'Airflow

Si vous utilisez Apache Airflow v2, ajoutez-le en `core.lazy_load_plugins : False` tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, reportez-vous à la section [Utilisation des options de configuration pour charger des plug-ins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).
- Découvrez comment charger le `plugins.zip` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator

L'exemple suivant explique comment appliquer un correctif à Apache Airflow PythonVirtualenvOperator avec un plugin personnalisé sur Amazon Managed Workflows for Apache Airflow.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code de plugin personnalisé](#)
- [Plugins.zip](#)
- [Exemple de code](#)
- [Options de configuration d'Airflow](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Autorisations

Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à `votrequirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
virtualenv
```

Exemple de code de plugin personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier des plugins au démarrage. Ce plugin corrigera le module intégré `PythonVirtualenvOperator` au cours de ce processus de démarrage pour le rendre compatible avec Amazon MWAA. Les étapes suivantes affichent l'exemple de code du plugin personnalisé.

1. Dans votre invite de commande, accédez au plugins répertoire de la section précédente.
Exemples :

```
cd plugins
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `virtual_python_plugin.py`.

```
"""
```

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

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

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

```
"""
```

```
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List
```

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

```
airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd
```

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

Plugins.zip

Les étapes suivantes expliquent comment créer le `plugins.zip`.

1. Dans votre invite de commande, accédez au répertoire figurant `virtual_python_plugin.py` dans la section précédente. Exemples :

```
cd plugins
```

2. Comprimez le contenu de votre plugins dossier.

```
zip plugins.zip virtual_python_plugin.py
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG pour le plugin personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Exemples :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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,
    )
```

Options de configuration d'Airflow

Si vous utilisez Apache Airflow v2, ajoutez-le en `core.lazy_load_plugins : False` tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, reportez-vous à la section [Utilisation des options de configuration pour charger des plug-ins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).
- Découvrez comment charger le `plugins.zip` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Invocation DAGs avec une fonction Lambda

L'exemple de code suivant utilise une [AWS Lambda](#) fonction pour obtenir un jeton CLI Apache Airflow et invoquer un graphe acyclique dirigé (DAG) dans un environnement Amazon MWAA.

Rubriques

- [Version](#)

- [Conditions préalables](#)
- [Permissions](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Pour utiliser cet exemple de code, vous devez :

- Utilisez le [mode d'accès au réseau public](#) pour votre [environnement Amazon MWAA](#).
- Utilisez une [fonction Lambda utilisant le dernier](#) environnement d'exécution Python.

Note

Si la fonction Lambda et votre environnement Amazon MWAA se trouvent dans le même VPC, vous pouvez utiliser ce code sur un réseau privé. Pour cette configuration, le rôle d'exécution de la fonction Lambda doit être autorisé à appeler l'opération d'API Amazon Elastic Compute Cloud (Amazon EC2) `CreateNetworkInterface`. Vous pouvez fournir cette autorisation à l'aide de la politique [AWSLambdaVPCLambdaAccessExecutionRole](#) AWS-managed.

Permissions

Pour utiliser l'exemple de code présenté sur cette page, le rôle d'exécution de votre environnement Amazon MWAA doit avoir accès pour effectuer l'`airflow:CreateCliTokenaction`. Vous pouvez fournir cette autorisation à l'aide de la politique `AmazonMWAAirflowCliAccess` AWS-managed :

JSON

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



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

Pour plus d'informations, consultez [Politique de la CLI Apache Airflow : Amazon MWAAirflow CliAccess](#).

Dépendances

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

1. Ouvrez la AWS Lambda console à l'adresse <https://console.aws.amazon.com/lambda/>.
2. Choisissez votre fonction Lambda dans la liste des fonctions.
3. Sur la page des fonctions, copiez le code suivant et remplacez-le par le nom de vos ressources :
 - YOUR_ENVIRONMENT_NAME— Le nom de votre environnement Amazon MWAA.
 - YOUR_DAG_NAME— Le nom du DAG que vous souhaitez invoquer.

```
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. Choisissez Déployer.
5. Choisissez Test pour appeler votre fonction à l'aide de la console Lambda.
6. Pour vérifier que votre Lambda a correctement appelé votre DAG, utilisez la console Amazon MWAA pour accéder à l'interface utilisateur Apache Airflow de votre environnement, puis procédez comme suit :
 - a. Sur la DAGspage, localisez votre nouveau DAG cible dans la liste des DAGs.
 - b. Sous Dernière exécution, vérifiez l'horodatage de la dernière exécution du DAG. Cet horodatage doit correspondre étroitement à l'horodatage le plus récent de votre autre `invoke_dag` environnement.
 - c. Sous Tâches récentes, vérifiez que la dernière exécution a été réussie.

Invocation DAGs dans différents environnements Amazon MWAA

L'exemple de code suivant crée un jeton d'interface de ligne de commande Apache Airflow. Le code utilise ensuite un graphe acyclique dirigé (DAG) dans un environnement Amazon MWAA pour appeler un DAG dans un autre environnement Amazon MWAA.

Rubriques

- [Version](#)

- [Conditions préalables](#)
- [Permissions](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Pour utiliser l'exemple de code présenté sur cette page, vous avez besoin des éléments suivants :

- Deux [environnements Amazon MWAA](#) avec accès au serveur Web du réseau public, y compris votre environnement actuel.
- Un exemple de DAG chargé dans le bucket Amazon Simple Storage Service (Amazon S3) de votre environnement cible.

Permissions

Pour utiliser l'exemple de code présenté sur cette page, le rôle d'exécution de votre environnement doit être autorisé à créer un jeton d'interface de ligne de commande Apache Airflow. Vous pouvez joindre la politique AWS-managed AmazonMWAAAirflowCliAccess pour accorder cette autorisation.

JSON

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

```
}  
]  
}
```

Pour plus d'informations, consultez [Politique de la CLI Apache Airflow : Amazon MWAAirflow CliAccess](#).

Dépendances

Pour utiliser cet exemple de code avec Apache Airflow v2 et versions ultérieures, aucune dépendance supplémentaire n'est requise. [aws-mwaa-docker-images](#) À utiliser pour installer Apache Airflow.

Exemple de code

L'exemple de code suivant suppose que vous utilisez un DAG dans votre environnement actuel pour appeler un DAG dans un autre environnement.

1. Dans votre terminal, accédez au répertoire dans lequel votre code DAG est enregistré. Par exemple :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous le nom `invoke_dag.py`. Remplacez les valeurs suivantes par vos informations.
 - `your-new-environment-name`— Le nom de l'autre environnement dans lequel vous souhaitez appeler le DAG.
 - `your-target-dag-id`— L'ID du DAG dans l'autre environnement que vous souhaitez invoquer.

```
from airflow.decorators import dag, task  
import boto3  
from datetime import datetime, timedelta  
import os, requests  
  
DAG_ID = os.path.basename(__file__).replace(".py", "")  
  
@task()
```

```
def invoke_dag_task(**kwargs):
    client = boto3.client('mwaa')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwaa/cli"
    body = 'days 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. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Si le DAG s'exécute correctement, vous obtiendrez un résultat similaire à ce qui suit dans les journaux des tâches pour `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
```

Pour vérifier que votre DAG a bien été invoqué, accédez à l'interface utilisateur Apache Airflow de votre nouvel environnement, puis procédez comme suit :

- a. Sur la DAGspage, localisez votre nouveau DAG cible dans la liste des DAGs.
- b. Sous Dernière exécution, vérifiez l'horodatage de la dernière exécution du DAG. Cet horodatage doit correspondre étroitement à l'horodatage le plus récent de votre autre `invoke_dag` environnement.
- c. Sous Tâches récentes, vérifiez que la dernière exécution a été réussie.

Utilisation d'Amazon MWAA avec Amazon RDS pour Microsoft SQL Server

Vous pouvez utiliser Amazon Managed Workflows pour Apache Airflow pour vous connecter à un serveur [RDS pour SQL](#). L'exemple de code suivant est utilisé DAGs dans un environnement Amazon Managed Workflows for Apache Airflow pour se connecter et exécuter des requêtes sur un Amazon RDS pour Microsoft SQL Server.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Dépendances](#)
- [Connexion Apache Airflow v2](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- Amazon MWAA et RDS for SQL Server s'exécutent dans le même Amazon VPC/

- Les groupes de sécurité VPC d'Amazon MWAA et du serveur sont configurés avec les connexions suivantes :
 - Une règle entrante pour le port 1433 ouvert pour Amazon RDS dans le groupe de sécurité Amazon MWAA
 - Ou une règle sortante pour le port d'1433ouverture d'Amazon MWAA vers RDS
- Apache Airflow Connection for RDS for SQL Server reflète le nom d'hôte, le port, le nom d'utilisateur et le mot de passe de la base de données Amazon RDS SQL Server créée lors du processus précédent.

Dépendances

Pour utiliser l'exemple de code présenté dans cette section, ajoutez la dépendance suivante à `votrequirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
apache-airflow-providers-microsoft-mssql==1.0.1
apache-airflow-providers-odbc==1.0.1
pymssql==2.2.1
```

Connexion Apache Airflow v2

Si vous utilisez une connexion dans Apache Airflow v2, assurez-vous que l'objet de connexion Airflow inclut les paires clé-valeur suivantes :

1. Identifiant de connexion : `mssql_default`
2. Type de connexion : Amazon Web Services
3. Hôte : `YOUR_DB_HOST`
4. Schéma :
5. Identifiant : `admin`
6. Mot de passe :
7. Hafen : `1433`
8. Supplémentaire :

Exemple de code

1. Dans votre invite de commande, accédez au répertoire dans lequel votre code DAG est stocké.
Par exemple :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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
```

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` fichier dans cet exemple dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment télécharger le code DAG dans cet exemple dans le dags dossier de votre compartiment Amazon S3 dans [Ajouter ou mettre à jour DAGs](#).
- Explorez des exemples de scripts et d'autres exemples de [modules pymssql](#).
- Pour en savoir plus sur l'exécution de code SQL dans une base de données Microsoft SQL spécifique à l'aide du [mssql_operator](#), consultez le guide de référence Apache Airflow.

Utilisation d'Amazon MWAA avec Amazon EKS

L'exemple suivant montre comment utiliser Amazon Managed Workflows pour Apache Airflow avec Amazon EKS.

Rubriques

- [Version](#)
- [Conditions préalables](#)
- [Création d'une clé publique pour Amazon EC2](#)
- [Créer le cluster](#)
- [Création d'un espace mwaa de noms](#)

- [Création d'un rôle pour l'espace de mwaa noms](#)
- [Création et attachement d'un rôle IAM pour le cluster Amazon EKS](#)
- [Créez le fichier requirements.txt](#)
- [Création d'un mappage d'identité pour Amazon EKS](#)
- [Créer le kubeconfig](#)
- [Création d'un DAG](#)
- [Ajoutez le DAG et kube_config.yaml au compartiment Amazon S3](#)
- [Activer et déclencher l'exemple](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Pour utiliser l'exemple présenté dans cette rubrique, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).
- eksctl. Pour en savoir plus, reportez-vous à la section [Installer eksctl](#).
- kubectl. Pour en savoir plus, reportez-vous à [Installer et configurer kubectl](#). Dans certains cas, cela est installé avec eksctl.
- Une paire de clés EC2 dans la région où vous créez votre environnement Amazon MWAA. Pour en savoir plus, reportez-vous à la section [Création ou importation d'une paire de clés](#).

Note

Lorsque vous utilisez une eksctl commande, vous pouvez inclure un `--profile` pour spécifier un profil autre que le profil par défaut.

Création d'une clé publique pour Amazon EC2

Utilisez la commande suivante pour créer une clé publique à partir de votre paire de clés privées.

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

Pour en savoir plus, reportez-vous à la section [Récupération de la clé publique de votre paire de clés](#).

Créer le cluster

Utilisez la commande suivante pour créer le cluster. Si vous souhaitez donner un nom personnalisé au cluster ou le créer dans une autre région, remplacez le nom et les valeurs de région. Vous devez créer le cluster dans la même région que celle dans laquelle vous créez l'environnement Amazon MWAA. Remplacez les valeurs des sous-réseaux pour qu'elles correspondent aux sous-réseaux de votre réseau Amazon VPC que vous utilisez pour Amazon MWAA. Remplacez la valeur de `ssh-public-key` qu'elle corresponde à la clé que vous utilisez. Vous pouvez utiliser une clé existante d'Amazon EC2 qui se trouve dans la même région, ou créer une nouvelle clé dans la même région où vous créez votre environnement Amazon MWAA.

```
eksctl create cluster \  
--name mwaa-eks \  
--region us-west-2 \  
--version 1.18 \  
--nodegroup-name linux-nodes \  
--nodes 3 \  
--nodes-min 1 \  
--nodes-max 4 \  
--with-oidc \  
--ssh-access \  
--ssh-public-key MyPublicKey \  
--managed \  
--vpc-public-subnets "subnet-1111111111111111, subnet-2222222222222222" \  
--vpc-private-subnets "subnet-3333333333333333, subnet-4444444444444444"
```

La création du cluster prend un certain temps. Une fois terminé, vous pouvez vérifier que le cluster a été créé correctement et que le fournisseur IAM OIDC est configuré à l'aide de la commande suivante :

```
eksctl utils associate-iam-oidc-provider \  
--region us-west-2 \  
--cluster mwaa-eks \  
--approve
```

Création d'un espace **mwaa** de noms

Après avoir confirmé que le cluster a bien été créé, utilisez la commande suivante pour créer un espace de noms pour les pods.

```
kubectl create namespace mwaa
```

Création d'un rôle pour l'espace de **mwaa** noms

Après avoir créé l'espace de noms, créez un rôle et une liaison de rôles pour un utilisateur Amazon MWAA sur EKS qui peut exécuter des pods dans un espace de noms MWAA. Si vous avez utilisé un autre nom pour l'espace de noms, remplacez `mwaa -n mwaa` par le nom que vous avez utilisé.

```
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
```

Vérifiez que le nouveau rôle peut accéder au cluster Amazon EKS en exécutant la commande suivante. Assurez-vous d'utiliser le nom correct si vous n'avez pas utilisé *mwaa* :

```
kubectl get pods -n mwaa --as mwaa-service
```

Vous recevez un message qui dit :

```
No resources found in mwaa namespace.
```

Création et attachement d'un rôle IAM pour le cluster Amazon EKS

Vous devez créer un rôle IAM, puis le lier au cluster Amazon EKS (k8s) afin qu'il puisse être utilisé pour l'authentification via IAM. Le rôle est uniquement utilisé pour se connecter au cluster et ne dispose d'aucune autorisation pour les appels de console ou d'API.

Créez un nouveau rôle pour l'environnement Amazon MWAA en suivant les étapes décrites dans [Rôle d'exécution Amazon MWAA](#). Toutefois, au lieu de créer et de joindre les politiques décrites dans cette rubrique, associez la stratégie suivante :

JSON

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

```

    "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
    ${MWAAS_ENV_NAME}"
  },
  {
    "Effect": "Deny",
    "Action": "s3:ListAllMyBuckets",
    "Resource": [
      "arn:aws:s3:::${MWAAS_S3_BUCKET}",
      "arn:aws:s3:::${MWAAS_S3_BUCKET}/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject*",
      "s3:GetBucket*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::${MWAAS_S3_BUCKET}",
      "arn:aws:s3:::${MWAAS_S3_BUCKET}/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogStream",
      "logs:CreateLogGroup",
      "logs:PutLogEvents",
      "logs:GetLogEvents",
      "logs:GetLogRecord",
      "logs:GetLogGroupFields",
      "logs:GetQueryResults",
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-east-1:111122223333:log-group:airflow-
    ${MWAAS_ENV_NAME}-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": "cloudwatch:PutMetricData",
    "Resource": "*"
  }

```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ChangeMessageVisibility",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-1::airflow-celery-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey*",
        "kms:Encrypt"
      ],
      "NotResource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringLike": {
          "kms:ViaService": [
            "sqs.us-east-1.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster"
      ],
      "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/
${EKS_CLUSTER_NAME}"
    }
  ]
}

```


Après avoir créé le rôle, modifiez votre environnement Amazon MWAA pour utiliser le rôle que vous avez créé comme rôle d'exécution pour l'environnement. Pour modifier le rôle, modifiez l'environnement à utiliser. Vous sélectionnez le rôle d'exécution sous Autorisations.

Problèmes connus :

- Il existe un problème connu lié au fait que les rôles ARNs associés aux sous-chemins ne peuvent pas s'authentifier auprès d'Amazon EKS. La solution consiste à créer le rôle de service manuellement plutôt que d'utiliser celui créé par Amazon MWAA lui-même. Pour en savoir plus, reportez-vous à la section [Les rôles avec des chemins ne fonctionnent pas lorsque le chemin est inclus dans leur ARN dans la carte de configuration aws-auth](#)
- Si la liste des services Amazon MWAA n'est pas disponible dans IAM, vous devez choisir une autre politique de service, telle qu'Amazon EC2, puis mettre à jour la politique de confiance du rôle pour qu'elle corresponde à ce qui suit :

JSON

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

Pour en savoir plus, consultez [Comment utiliser les politiques de confiance avec les rôles IAM](#).

Créez le fichier requirements.txt

Pour utiliser l'exemple de code présenté dans cette section, assurez-vous d'avoir ajouté l'une des options de base de données suivantes à votre `requirements.txt`. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Création d'un mappage d'identité pour Amazon EKS

Utilisez l'ARN du rôle que vous avez créé dans la commande suivante afin de créer un mappage d'identité pour Amazon EKS. Remplacez la région *us-east-1* par la région dans laquelle vous avez créé l'environnement. Remplacez l'ARN du rôle, puis remplacez-le par le *mwaa-execution-role* rôle d'exécution de votre environnement.

```
eksctl create iamidentitymapping \
--region us-east-1 \
--cluster mwaa-eks \
--arn arn:aws:iam::123456789012:role/mwaa-execution-role \
--username mwaa-service
```

Créer le **kubeconfig**

Utilisez la commande suivante pour créer kubeconfig :

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

Si vous avez utilisé un profil spécifique lors de l'exécution, update-kubeconfig vous devez supprimer la env : section ajoutée au fichier kube_config.yaml afin qu'il fonctionne correctement avec Amazon MWAA. Pour ce faire, supprimez les éléments suivants du fichier, puis enregistrez-le :

```
env:
- name: AWS_PROFILE
  value: profile_name
```

Création d'un DAG

Utilisez l'exemple de code suivant pour créer un fichier Python, par exemple `mwaa_pod_example.py` pour le DAG.

```
"""
```

```
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
"""
```

```
from airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
    KubernetesPodOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}

dag = DAG(
    'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
    namespace="mwaa",
    image="ubuntu:18.04",
    cmds=["bash"],
    arguments=["-c", "ls"],
    labels={"foo": "bar"},
    name="mwaa-pod-test",
    task_id="pod-task",
    get_logs=True,
    dag=dag,
    is_delete_operator_pod=False,
```

```
config_file=kube_config_path,  
in_cluster=False,  
cluster_context='aws'  
)
```

Ajoutez le DAG et **kube_config.yaml** au compartiment Amazon S3

Placez le DAG que vous avez créé et le `kube_config.yaml` fichier dans le compartiment Amazon S3 pour l'environnement Amazon MWAA. Vous pouvez placer des fichiers dans votre compartiment à l'aide de la console Amazon S3 ou du AWS Command Line Interface.

Activer et déclencher l'exemple

Dans Apache Airflow, activez l'exemple, puis déclenchez-le.

Une fois qu'il a été exécuté et terminé avec succès, utilisez la commande suivante pour vérifier le pod :

```
kubectl get pods -n mwaa
```

Vous obtenez un résultat similaire à ce qui suit :

```
NAME READY STATUS RESTARTS AGE  
mwaa-pod-test-aa11bb22cc3344445555666677778888 0/1 Completed 0 2m23s
```

Vous pouvez ensuite vérifier la sortie du pod à l'aide de la commande suivante. Remplacez la valeur du nom par la valeur renvoyée par la commande précédente :

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc3344445555666677778888
```

Connexion à Amazon ECS à l'aide du **ECSOperator**

Cette rubrique décrit comment vous pouvez utiliser le `ECSOperator` pour vous connecter à un conteneur Amazon Elastic Container Service (Amazon ECS) depuis Amazon MWAA. Au cours des étapes suivantes, vous allez ajouter les autorisations requises au rôle d'exécution de votre environnement, utiliser un CloudFormation modèle pour créer un cluster Amazon ECS Fargate, puis créer et télécharger un DAG qui se connecte à votre nouveau cluster.

Rubriques

- [Version](#)
- [Conditions préalables](#)
- [Permissions](#)
- [Création d'un cluster Amazon ECS](#)
- [Exemple de code](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Permissions

- Le rôle d'exécution de votre environnement nécessite une autorisation pour exécuter des tâches dans Amazon ECS. Vous pouvez soit associer la politique FullAccess AWS gérée par [AmazonECS_](#) à votre rôle d'exécution, soit créer et associer la politique suivante à votre rôle d'exécution.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask",
        "ecs:DescribeTasks"
      ],
      "Resource": "*"
    }
  ]
}
```

```

    },
    {
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "ecs-tasks.amazonaws.com"
        }
      }
    }
  ]
}

```

- Outre l'ajout des autorisations requises pour exécuter des tâches dans Amazon ECS, vous devez également modifier la déclaration de politique relative aux CloudWatch journaux dans votre rôle d'exécution Amazon MWAA afin de permettre l'accès au groupe de journaux de tâches Amazon ECS, comme indiqué ci-dessous. Le groupe de journaux Amazon ECS est créé par le CloudFormation modèle dans [the section called “Création d'un cluster Amazon ECS”](#).

```

{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
  ],
  "Resource": [
    "arn:aws:logs:us-east-1:123456789012:log-group:airflow-environment-name-*",
    "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
  ]
}

```

Pour plus d'informations sur le rôle d'exécution Amazon MWAA et sur la manière d'associer une politique, reportez-vous à [Rôle d'exécution](#).

Création d'un cluster Amazon ECS

À l'aide du CloudFormation modèle suivant, vous allez créer un cluster Amazon ECS Fargate à utiliser avec votre flux de travail Amazon MWAA. Pour plus d'informations, consultez la section [Création d'une définition de tâche](#) dans le manuel Amazon Elastic Container Service Developer Guide.

1. Créez un fichier JSON avec le code suivant et enregistrez-le `sousecs-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"
    }
  }
}
}
```

- Dans votre invite de commande, utilisez la AWS CLI commande suivante pour créer une nouvelle pile. Vous devez remplacer les valeurs `SecurityGroups` et `SubnetIds` des valeurs pour les groupes de sécurité et les sous-réseaux de votre environnement Amazon MWAA.

```
aws cloudformation create-stack \
--stack-name my-ecs-stack --template-body file://ecs-mwaa-cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=your-mwaa-security-group \
ParameterKey=SubnetIds,ParameterValue=your-mwaa-subnet-1\\,your-mwaa-subnet-1 \
--capabilities CAPABILITY_IAM
```

Vous pouvez également utiliser le script shell suivant. Le script récupère les valeurs requises pour les groupes de sécurité et les sous-réseaux de votre environnement à l'aide de la [get-environment](#) AWS CLI commande, puis crée la pile en conséquence. Pour exécuter le script, procédez comme suit.

- a. Copiez et enregistrez le script `ecs-stack-helper.sh` dans le même répertoire que votre CloudFormation modèle.

```
#!/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. Exécutez le script à l'aide des commandes suivantes. Remplacez `environment-name` et `stack-name` par vos informations.

```
chmod +x ecs-stack-helper.sh
./ecs-stack-helper.bash environment-name stack-name
```

En cas de succès, vous vous référerez à la sortie suivante indiquant votre nouvel ID de CloudFormation pile.

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

Une fois votre CloudFormation stack terminé et AWS approvisionné vos ressources Amazon ECS, vous êtes prêt à créer et à télécharger votre DAG.

Exemple de code

1. Ouvrez une invite de commande et naviguez jusqu'au répertoire dans lequel votre code DAG est stocké. Par exemple :

```
cd dags
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous le `mwaas-ecs-operator.py` nom, puis téléchargez votre nouveau DAG sur 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="mwaas-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
information.
CONTAINER_NAME="mwaas-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="mwaas-ecs-zero",
```

```
awslogs_stream_prefix=f"ecs/{CONTAINER_NAME}",
)
```

Note

Dans l'exemple DAG, pour `awslogs_group`, vous devrez peut-être modifier le groupe de journaux avec le nom de votre groupe de journaux de tâches Amazon ECS. L'exemple suppose un groupe de journaux nommé `mwa-ecs-zero`. Pour `awslogs_stream_prefix`, utilisez le préfixe du flux du journal des tâches Amazon ECS. L'exemple suppose un préfixe de flux de journal `ecs`.

3. Exécutez la AWS CLI commande suivante pour copier le DAG dans le bucket de votre environnement, puis déclenchez le DAG à l'aide de l'interface utilisateur d'Apache Airflow.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. En cas de succès, vous obtiendrez un résultat similaire à ce qui suit dans les journaux des `ecs_operator_task` tâches du `ecs_fargate_dag` DAG :

```
[2022-01-01, 12:00:00 UTC] [{ecs.py:300}] INFO - Running ECS Task -
Task definition: arn:aws:ecs:us-west-2:123456789012:task-definition/mwa-ecs-test-
task:1 - on cluster mwa-ecs-test-cluster
[2022-01-01, 12:00:00 UTC] [{ecs-operator-test.py:302}] INFO - ECSOperator
overrides:
{'containerOverrides': [{'name': 'mwa-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

```

Utiliser dbt avec Amazon MWAA

Cette rubrique explique comment utiliser dbt et Postgres avec Amazon MWAA. Au cours des étapes suivantes, vous allez ajouter les dépendances requises à votre `requirements.txt` et télécharger un exemple de projet dbt dans le compartiment Amazon S3 de votre environnement. Ensuite, vous utiliserez un exemple de DAG pour vérifier qu'Amazon MWAA a installé les dépendances, puis vous utiliserez le `BashOperator` pour exécuter le projet dbt.

Rubriques

- [Version](#)
- [Conditions préalables](#)
- [Dépendances](#)
- [Importer un projet dbt sur Amazon S3](#)
- [Utiliser un DAG pour vérifier l'installation de la dépendance dbt](#)
- [Utiliser un DAG pour exécuter un projet dbt](#)

Version

[Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 en Python 3.10 et Apache Airflow v3 en Python 3.11.](#)

Conditions préalables

Avant de pouvoir effectuer les étapes suivantes, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#) utilisant Apache Airflow v2.2.2. Cet exemple a été écrit et testé avec la version 2.2.2. Vous devrez peut-être modifier l'exemple pour l'utiliser avec d'autres versions d'Apache Airflow.
- Un exemple de projet de dette. Pour commencer à utiliser dbt avec Amazon MWAA, vous pouvez créer un fork et cloner le [projet de démarrage dbt à partir du référentiel dbt-labs](#). GitHub

Dépendances

Pour utiliser Amazon MWAA avec dbt, ajoutez le script de démarrage suivant à votre environnement. Pour en savoir plus, consultez la section [Utilisation d'un script de démarrage avec Amazon MWAA](#).

```
#!/bin/bash

if [[ "${MWAA_AIRFLOW_COMPONENT}" != "worker" ]]
then
    exit 0
fi

echo "-----"
echo "Installing virtual Python env"
echo "-----"

pip3 install --upgrade pip

echo "Current Python version:"
python3 --version
echo "..."

sudo pip3 install --user virtualenv
sudo mkdir python3-virtualenv
cd python3-virtualenv
sudo python3 -m venv dbt-env
sudo chmod -R 777 *

echo "-----"
echo "Activating venv in"
$DBT_ENV_PATH
    echo "-----"

source dbt-env/bin/activate
pip3 list

echo "-----"
echo "Installing libraries..."
echo "-----"

# do not use sudo, as it will install outside the venv
pip3 install dbt-redshift==1.6.1 dbt-postgres==1.6.1

echo "-----"
echo "Venv libraries..."
echo "-----"

pip3 list
```



```
dbt --version

echo "-----"
echo "Deactivating venv..."
echo "-----"

deactivate
```

Dans les sections suivantes, vous allez télécharger le répertoire de votre projet dbt sur Amazon S3 et exécuter un DAG qui vérifie si Amazon MWAA a correctement installé les dépendances dbt requises.

Importer un projet dbt sur Amazon S3

Pour pouvoir utiliser un projet dbt avec votre environnement Amazon MWAA, vous pouvez télécharger l'intégralité du répertoire du projet dans le dossier de dags votre environnement. Lorsque l'environnement est mis à jour, Amazon MWAA télécharge le répertoire dbt dans le dossier `localusr/local/airflow/dags/`.

Pour télécharger un projet dbt sur Amazon S3

1. Accédez au répertoire dans lequel vous avez cloné le projet de démarrage dbt.
2. Exécutez la AWS CLI commande Amazon S3 suivante pour copier de manière récursive le contenu du projet dans le dags dossier de votre environnement à l'aide du `--recursive` paramètre. La commande crée un sous-répertoire appelé dbt que vous pouvez utiliser pour tous vos projets dbt. Si le sous-répertoire existe déjà, les fichiers du projet sont copiés dans le répertoire existant et aucun nouveau répertoire n'est créé. La commande crée également un sous-répertoire dans le dbt répertoire pour ce projet de démarrage spécifique.

```
aws s3 cp dbt-starter-project s3://amzn-s3-demo-bucket/dags/dbt/dbt-starter-project
--recursive
```

Vous pouvez utiliser différents noms pour les sous-répertoires de projets afin d'organiser plusieurs projets dbt dans le répertoire parentdbt.

Utiliser un DAG pour vérifier l'installation de la dépendance dbt

Le DAG suivant utilise une commande `BashOperator` et une commande `bash` pour vérifier si Amazon MWAA a correctement installé les dépendances dbt spécifiées dans `requirements.txt`

```
from airflow import DAG
    from airflow.operators.bash_operator import BashOperator
    from airflow.utils.dates import days_ago

    with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
start_date=days_ago(1)) as dag:
        cli_command = BashOperator(
            task_id="bash_command",
            bash_command="/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt --version"
        )
```

Procédez comme suit pour accéder aux journaux des tâches et vérifier que dbt et ses dépendances ont été installés.

1. Accédez à la console Amazon MWAA, puis choisissez Open Airflow UI dans la liste des environnements disponibles.
2. Dans l'interface utilisateur d'Apache Airflow, recherchez le `dbt-installation-test` DAG dans la liste, puis choisissez la date dans la `Last Run` colonne pour ouvrir la dernière tâche réussie.
3. À l'aide de Graph View, choisissez la `bash_command` tâche pour ouvrir les détails de l'instance de tâche.
4. Choisissez Log pour ouvrir les journaux des tâches, puis vérifiez que les journaux répertorient correctement la version de dbt dans `requirements.txt` laquelle nous avons spécifiée.

Utiliser un DAG pour exécuter un projet dbt

Le DAG suivant utilise un `BashOperator` pour copier les projets dbt que vous avez chargés sur Amazon S3 depuis le `usr/local/airflow/dags/` répertoire local vers le `/tmp` répertoire accessible en écriture, puis exécute le projet dbt. Les commandes bash supposent un projet dbt de démarrage intitulé. `dbt-starter-project` Modifiez le nom du répertoire en fonction du nom du répertoire de votre projet.

```
from airflow import DAG
    from airflow.operators.bash_operator import BashOperator
    from airflow.utils.dates import days_ago

    import os
```

```
DAG_ID = os.path.basename(__file__).replace(".py", "")

# assumes all files are in a subfolder of DAGs called dbt

with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="source /usr/local/airflow/python3-virtualenv/dbt-env/bin/activate;\
cp -R /usr/local/airflow/dags/dbt /tmp;\
echo 'listing project files:';\
ls -R /tmp;\
cd /tmp/dbt/mwaa_dbt_test_project;\
/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt run --project-dir /tmp/dbt/\
mwaa_dbt_test_project --profiles-dir ..;\
cat /tmp/dbt_logs/dbt.log;\
rm -rf /tmp/dbt/mwaa_dbt_test_project"
    )
```

AWS blogs et tutoriels

- [Utilisation d'Amazon EKS et Amazon MWAA pour Apache Airflow v2.x](#)

Bonnes pratiques relatives aux flux de travail gérés par Amazon pour Apache Airflow

Ce guide décrit les meilleures pratiques que nous recommandons lors de l'utilisation d'Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Optimisation des performances pour Apache Airflow sur Amazon MWAA](#)
- [Gestion des dépendances Python dans requirements.txt](#)

Optimisation des performances pour Apache Airflow sur Amazon MWAA

Cette rubrique décrit comment optimiser les performances d'un environnement Amazon Managed Workflows pour Apache Airflow à l'aide de. [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#)

Table des matières

- [Ajout d'une option de configuration d'Apache Airflow](#)
- [Planificateur Apache Airflow](#)
 - [Parameters](#)
 - [Restrictions](#)
- [Dossiers DAG](#)
 - [Parameters](#)
- [fichiers DAG](#)
 - [Parameters](#)
- [Tâches](#)
 - [Parameters](#)

Ajout d'une option de configuration d'Apache Airflow

Utilisez la procédure suivante pour ajouter une option de configuration Airflow à votre environnement.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Choisissez Ajouter une configuration personnalisée dans le volet des options de configuration d'Airflow.
6. Choisissez une configuration dans la liste déroulante et entrez une valeur, ou entrez une configuration personnalisée et entrez une valeur.
7. Choisissez Ajouter une configuration personnalisée pour chaque configuration que vous souhaitez ajouter.
8. Choisissez Enregistrer.

Pour en savoir plus, reportez-vous à [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).

Planificateur Apache Airflow

Le planificateur Apache Airflow est un composant essentiel d'Apache Airflow. Un problème avec le planificateur peut empêcher l'analyse et la planification des tâches. Pour plus d'informations sur le réglage du planificateur Apache Airflow, reportez-vous à la section [Optimisation des performances de votre planificateur sur le site Web de documentation](#) d'Apache Airflow.

Parameters

Cette section décrit les options de configuration disponibles pour le planificateur Apache Airflow (Apache Airflow v2 et versions ultérieures) et leurs cas d'utilisation.

Apache Airflow v3

Configuration	Cas d'utilisation
celery.sync_parallelism Nombre de processus utilisés par Celery Executor pour synchroniser l'état des tâches.	Vous pouvez utiliser cette option pour éviter les conflits de files d'attente en limitant les processus utilisés par Celery Executor. Par défaut, une valeur est définie sur 1 pour éviter les erreurs lors de la transmission des

Configuration	Cas d'utilisation
Par défaut : 1	journaux des tâches à CloudWatch Logs. Si la valeur est définie sur 0, vous devez utiliser le nombre maximum de processus, mais cela peut entraîner des erreurs lors de la remise des journaux de tâches.
<u>scheduler.scheduler_idle_sleep_time</u> Le nombre de secondes à attendre entre deux traitements consécutifs de fichiers DAG dans la « boucle » du planificateur. Par défaut : 1	Vous pouvez utiliser cette option pour libérer l'utilisation du processeur sur le planificateur en augmentant le temps de veille du planificateur une fois qu'il a fini de récupérer les résultats de l'analyse DAG, de rechercher et de mettre en file d'attente des tâches et d'exécuter les tâches en file d'attente dans l'exécuteur. L'augmentation de cette valeur consomme le nombre de threads du planificateur exécutés sur un environnement dans Apache Airflow <code>dag_processor.parsing_processes</code> v2 et Apache Airflow v3. Cela peut réduire la capacité d'analyse DAGs des planificateurs et augmenter le temps nécessaire DAGs au remplissage sur le serveur Web.
<u>scheduler.max_dagruns_to_create_per_loop</u> Le nombre maximum de « boucles » DAGs à créer DagRun pour chaque « boucle » du planificateur. Par défaut : 10	Vous pouvez utiliser cette option pour libérer des ressources pour planifier des tâches en diminuant le nombre maximum de « boucles » DagRun pour la « boucle » du planificateur.

Configuration	Cas d'utilisation
<u>dag_processor.parsing_processes</u> Le nombre de threads que le planificateur peut exécuter en parallèle pour planifier. DAGs Par défaut : Utiliser $(2 * \text{number of vCPUs}) - 1$	Vous pouvez utiliser cette option pour libérer des ressources en diminuant le nombre de processus que le planificateur exécute en parallèle pour analyser. DAGs Nous recommandons de maintenir ce nombre à un faible niveau si l'analyse DAG a un impact sur la planification des tâches. Vous devez spécifier une valeur inférieure au nombre de vCPU de votre environnement. Pour en savoir plus, reportez-vous à la section <u>Limites</u>.

Apache Airflow v2

Configuration	Cas d'utilisation
<u>celery.sync_parallelism</u> Nombre de processus utilisés par Celery Executor pour synchroniser l'état des tâches. Par défaut : 1	Vous pouvez utiliser cette option pour éviter les conflits de files d'attente en limitant les processus utilisés par Celery Executor. Par défaut, une valeur est définie sur 1 pour éviter les erreurs lors de la transmission des journaux des tâches à CloudWatch Logs. Si la valeur est définie sur 0, vous devez utiliser le nombre maximum de processus, mais cela peut entraîner des erreurs lors de la remise des journaux de tâches.
<u>scheduler.idle_sleep_time</u> Le nombre de secondes à attendre entre deux traitements consécutifs de fichiers DAG dans la « boucle » du planificateur. Par défaut : 1	Vous pouvez utiliser cette option pour libérer l'utilisation du processeur sur le planificateur en augmentant le temps de veille du planificateur une fois qu'il a fini de récupérer les résultats de l'analyse DAG, de rechercher et de mettre en file d'attente des tâches et

Configuration	Cas d'utilisation
scheduler.max_dagruns_to_create_per_loop Le nombre maximum de « boucles » DAGs à créer DagRunspour chaque « boucle » du planificateur. Par défaut : 10	d'exécuter les tâches en file d'attente dans l'exécuteur. L'augmentation de cette valeur consomme le nombre de threads du planificateur exécutés sur un environnement dans Apache Airflow <code>scheduler.parsing_processes</code> v2 et Apache Airflow v3. Cela peut réduire la capacité d'analyse DAGs des planificateurs et augmenter le temps nécessaire DAGs au remplissage sur le serveur Web. Vous pouvez utiliser cette option pour libérer des ressources pour planifier des tâches en diminuant le nombre maximum de « boucles » DagRunspour la « boucle » du planificateur.
scheduler.parsing_processes Le nombre de threads que le planificateur peut exécuter en parallèle pour planifier. DAGs Par défaut : Utiliser $(2 * \text{number of vCPUs}) - 1$	Vous pouvez utiliser cette option pour libérer des ressources en diminuant le nombre de processus que le planificateur exécute en parallèle pour analyser. DAGs Nous recommandons de maintenir ce nombre à un faible niveau si l'analyse DAG a un impact sur la planification des tâches. Vous devez spécifier une valeur inférieure au nombre de vCPU de votre environnement. Pour en savoir plus, reportez-vous à la section Limites.

Restrictions

Cette section décrit les limites à prendre en compte lors de l'ajustement des paramètres par défaut du planificateur.

`scheduler.parsing_processes`, `scheduler.max_threads` (version 2 uniquement)

Deux threads sont autorisés par vCPU pour une classe d'environnement. Au moins un thread doit être réservé au planificateur d'une classe d'environnement. Si vous remarquez un retard dans la planification des tâches, vous devrez peut-être augmenter votre [classe d'environnement](#). Par exemple, un environnement de grande taille possède une instance de conteneur Fargate à 4 vCPU pour son planificateur. Cela signifie qu'un maximum de 7 threads peuvent être utilisés pour d'autres processus. C'est-à-dire deux threads multipliés par quatre vCPUs, moins un pour le planificateur lui-même. La valeur que vous spécifiez dans `scheduler.max_threads` (v2 uniquement) ne `scheduler.parsing_processes` doit pas dépasser le nombre de threads disponibles pour une classe d'environnement, comme indiqué :

- `mw1.small` — Ne doit pas dépasser le nombre de 1 threads pour les autres processus. Le thread restant est réservé au planificateur.
- `mw1.medium` — Ne doit pas dépasser le nombre de 3 threads pour les autres processus. Le thread restant est réservé au planificateur.
- `mw1.large` — Ne doit pas dépasser le nombre de 7 threads pour les autres processus. Le thread restant est réservé au planificateur.

Dossiers DAG

Le planificateur Apache Airflow analyse en permanence le DAGs dossier de votre environnement. Tous `plugins.zip` les fichiers contenus ou les fichiers Python (`.py`) contenant des instructions d'importation « `airflow` ». Tous les objets DAG Python qui en résultent sont ensuite placés dans un fichier `DagBag` pour ce fichier afin d'être traités par le planificateur afin de déterminer quelles tâches, le cas échéant, doivent être planifiées. L'analyse des fichiers DAG a lieu indépendamment du fait que les fichiers contiennent ou non des objets DAG viables.

Parameters

Cette section décrit les options de configuration disponibles pour le DAGs dossier (Apache Airflow v2 et versions ultérieures) et leurs cas d'utilisation.

Apache Airflow v3

Configuration	Cas d'utilisation
dag_processor.refresh_interval	

Configuration	Cas d'utilisation
Le nombre de secondes pendant lesquelles le DAGs dossier doit être scanné à la recherche de nouveaux fichiers. Par défaut : 300 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes nécessaires à l'analyse du DAGs dossier. Nous vous recommandons d'augmenter cette valeur si les temps d'analyse sont longs <code>total_parse_time</code> metrics, ce qui peut être dû au grand nombre de fichiers dans votre DAGs dossier.
<u>dag_processor.min_file_process_interval</u> Le nombre de secondes après lequel le planificateur analyse un DAG et les mises à jour du DAG sont reflétées. Par défaut : 30 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes pendant lesquelles le planificateur attend avant d'analyser un DAG. Par exemple, si vous spécifiez une valeur de 30, le fichier DAG est analysé toutes les 30 secondes. Nous vous recommandons de maintenir ce chiffre à un niveau élevé afin de réduire l'utilisation du processeur dans votre environnement.

Apache Airflow v2

Configuration	Cas d'utilisation
<u>scheduler.dag_dir_list_interval</u> Le nombre de secondes pendant lesquelles le DAGs dossier doit être scanné à la recherche de nouveaux fichiers. Par défaut : 300 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes nécessaires à l'analyse du DAGs dossier. Nous vous recommandons d'augmenter cette valeur si les temps d'analyse sont longs <code>total_parse_time</code> metrics, ce qui peut être dû au grand nombre de fichiers dans votre DAGs dossier.

Configuration	Cas d'utilisation
<u>scheduler.min_file_process_interval</u> Le nombre de secondes après lequel le planificateur analyse un DAG et les mises à jour du DAG sont reflétées. Par défaut : 30 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes pendant lesquelles le planificateur attend avant d'analyser un DAG. Par exemple, si vous spécifiez une valeur de 30, le fichier DAG est analysé toutes les 30 secondes. Nous vous recommandons de maintenir ce chiffre à un niveau élevé afin de réduire l'utilisation du processeur dans votre environnement.

fichiers DAG

Dans le cadre de la boucle du planificateur Apache Airflow, les fichiers DAG individuels sont analysés pour extraire les objets Python du DAG. Dans Apache Airflow v2 et versions ultérieures, le planificateur analyse un nombre maximum de processus d'[analyse simultanément](#). Le nombre de secondes spécifié dans `scheduler.min_file_process_interval` (v2) ou `dag_processor.min_file_process_interval` (v3) doit s'écouler avant que le même fichier ne soit à nouveau analysé.

Parameters

Cette section décrit les options de configuration disponibles pour les fichiers DAG Apache Airflow (Apache Airflow v2 et versions ultérieures) et leurs cas d'utilisation.

Apache Airflow v3

Configuration	Cas d'utilisation
<u>dag_processor.dag_file_processor_timeout</u> Nombre de secondes avant l'expiration du délai DagFileProcessor de traitement d'un fichier DAG.	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le temps nécessaire avant l'expiration des DagFileProcessor de délais. Nous vous recommandons d'augmenter cette valeur si vous rencontrez des délais d'attente dans les journaux de

Configuration	Cas d'utilisation
Par défaut : 50 secondes	traitement de votre DAG qui empêchent le chargement de fichiers viables DAGs .
<u>core.dagbag_import_timeout</u> Expiration du délai en secondes avant l'importation d'un fichier Python. Par défaut : 30 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le délai d'expiration du planificateur lors de l'importation d'un fichier Python pour extraire les objets DAG. Cette option est traitée dans le cadre de la « boucle » du planificateur et doit contenir une valeur inférieure à celle spécifiée dans <code>dag_processor.dag_file_processor_timeout</code>
<u>core.min_serialized_dag_update_interval</u> Nombre minimal de secondes après lequel les données sérialisées DAGs dans la base de données sont mises à jour. Par défaut : 30	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes après lesquelles les numéros de série de la base de DAGs données sont mis à jour. Nous vous recommandons d'augmenter cette valeur si vous en avez un grand nombre ou si vous DAGs en avez un complexe DAGs. L'augmentation de cette valeur réduit la charge sur le planificateur et la base de données lors de la DAGs sérialisation.

Configuration	Cas d'utilisation
<u>core.min_serialized_dag_fetch_interval</u> Le nombre de secondes pendant lesquelles un DAG sérialisé est extrait à nouveau de la base de données alors qu'il est déjà chargé dans le DagBag Par défaut : 10	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes pendant lesquelles un DAG sérialisé est à nouveau extrait. La valeur doit être supérieure à la valeur spécifiée dans <code>core.min_serialized_dag_update_interval</code> pour réduire les taux d'« écriture » de la base de données. L'augmentation de cette valeur réduit la charge sur le serveur Web et la base de données lors de la DAGs sérialisation.

Apache Airflow v2

Configuration	Cas d'utilisation
<u>core.dag_file_processor_timeout</u> Nombre de secondes avant l'expiration du délai DagFileProcessor de traitement d'un fichier DAG. Par défaut : 50 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le temps nécessaire avant l'expiration des DagFileProcessor de délais. Nous vous recommandons d'augmenter cette valeur si vous rencontrez des délais d'attente dans les journaux de traitement de votre DAG qui empêchent le chargement de fichiers viables DAGs .
<u>core.dagbag_import_timeout</u> Expiration du délai en secondes avant l'importation d'un fichier Python. Par défaut : 30 secondes	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le délai d'expiration du planificateur lors de l'importation d'un fichier Python pour extraire les objets DAG. Cette option est traitée dans le cadre de la « boucle » du planificateur et doit contenir une valeur inférieure à celle spécifiée

Configuration	Cas d'utilisation
	<code>dans. core.dag_file_processor_timeout</code>
<u>core.min_serialized_dag_update_interval</u> Nombre minimal de secondes après lequel les données sérialisées DAGs dans la base de données sont mises à jour. Par défaut : 30	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes après lesquelles les numéros de série de la base de DAGs données sont mis à jour. Nous vous recommandons d'augmenter cette valeur si vous en avez un grand nombre ou si vous DAGs en avez un complexe DAGs. L'augmentation de cette valeur réduit la charge sur le planificateur et la base de données lors de la DAGs sérialisation.
<u>core.min_serialized_dag_fetch_interval</u> Le nombre de secondes pendant lesquelles un DAG sérialisé est extrait à nouveau de la base de données alors qu'il est déjà chargé dans le. DagBag Par défaut : 10	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes pendant lesquelles un DAG sérialisé est à nouveau extrait. La valeur doit être supérieure à la valeur spécifiée dans <code>core.min_serialized_dag_update_interval</code> pour réduire les taux d' « écriture » de la base de données. L'augmentation de cette valeur réduit la charge sur le serveur Web et la base de données lors de la DAGs sérialisation.

Tâches

Le planificateur et les outils de travail d'Apache Airflow sont tous deux impliqués dans les tâches de mise en file d'attente et de suppression des files d'attente. Le planificateur fait passer les tâches analysées prêtes à être planifiées du statut Aucune au statut Planifié. L'exécuteur, qui s'exécute également sur le conteneur du planificateur de Fargate, met ces tâches en file d'attente et définit leur statut sur Queued. Lorsque les travailleurs ont de la capacité, ils retirent la tâche de la file d'attente et

lui attribuent le statut En cours d'exécution, ce qui change ensuite son statut en Succès ou Échec en fonction de la réussite ou de l'échec de la tâche.

Parameters

Cette section décrit les options de configuration disponibles pour les tâches Apache Airflow et leurs cas d'utilisation.

Les options de configuration par défaut qu'Amazon MWAA remplace sont indiquées. *red*

Apache Airflow v3

Configuration	Cas d'utilisation
core.parallélisme Nombre maximal d'instances de tâches pouvant avoir un Running statut. Par défaut : défini dynamiquement en fonction de $(\text{maxWorkers} * \text{maxCeleryWorkers}) / \text{schedulers} * 1.5$.	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre d'instances de tâches pouvant être exécutées simultanément. La valeur spécifiée doit être le nombre de travailleurs disponibles multiplié par la densité des tâches des travailleurs. Nous vous recommandons de modifier cette valeur uniquement lorsque vous rencontrez un grand nombre de tâches bloquées dans l'état « En cours » ou « En file d'attente ».
core.execute_tasks_new_python_interpreter Détermine si Apache Airflow exécute des tâches en bifurquant le processus parent ou en créant un nouveau processus Python. Par défaut : True	Lorsqu'il est défini sur True, Apache Airflow reconnaît les modifications que vous apportez à vos plugins en tant que nouveau processus Python créé pour exécuter des tâches.
celery.worker_concurrency Amazon MWAA remplace l'installation de base d'Airflow pour cette option afin de dimensionner les travailleurs dans le cadre	<i>Any value specified for this option is ignored.</i>

Configuration	Cas d'utilisation
de son composant de dimensionnement automatique. Par défaut : Non applicable	
celery.worker_autoscale La simultanéité des tâches pour les travailleurs. Défauts : • mw1.micro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mW1. Large - 20,0 • mw1.xlarge - 40,0 • mW1,2xlarge - 80,0	Vous pouvez utiliser cette option pour libérer des ressources en réduisant la maximum simultanéité des minimum tâches entre les travailleurs. Les travailleurs acceptent jusqu'à la maximum limite des tâches simultanées configurées, que les ressources soient suffisantes ou non pour le faire. Si les tâches sont planifiées sans ressources suffisantes, elles échouent immédiatement. Nous recommandons de modifier cette valeur pour les tâches gourmandes en ressources en réduisant les valeurs à des valeurs inférieures aux valeurs par défaut afin d'augmenter la capacité par tâche.

Apache Airflow v2

Configuration	Cas d'utilisation
core.parallélisme Nombre maximal d'instances de tâches pouvant avoir un Running statut. Par défaut : défini dynamiquement en fonction de $(\text{maxWorkers} * \text{maxCeleryWorkers}) / \text{schedulers} * 1.5$.	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre d'instances de tâches pouvant être exécutées simultanément. La valeur spécifiée doit être le nombre de travailleurs disponibles multiplié par la densité des tâches des travailleurs. Nous vous recommandons de modifier cette valeur uniquement lorsque vous rencontrez

Configuration	Cas d'utilisation
	un grand nombre de tâches bloquées dans l'état « En cours » ou « En file d'attente ».
core.dag_concurrency Le nombre d'instances de tâches autorisées à s'exécuter simultanément pour chaque DAG. Par défaut : 10000	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre d'instances de tâches autorisées à s'exécuter simultanément. Par exemple, si vous avez cent tâches DAGs avec dix tâches parallèles et que vous souhaitez que toutes DAGs s'exécutent simultanément, vous pouvez calculer le parallélisme maximal en multipliant le nombre de travailleurs disponibles par la densité de tâches des travailleurs <code>celery.worker_concurrency</code>, divisé par le nombre de DAGs
core.execute_tasks_new_python_interpreter Détermine si Apache Airflow exécute des tâches en bifurquant le processus parent ou en créant un nouveau processus Python. Par défaut : True	Lorsqu'il est défini sur True, Apache Airflow reconnaît les modifications que vous apportez à vos plugins en tant que nouveau processus Python créé pour exécuter des tâches.
celery.worker_concurrency Amazon MWAA remplace l'installation de base d'Airflow pour cette option afin de dimensionner les travailleurs dans le cadre de son composant de dimensionnement automatique. Par défaut : Non applicable	<i>Any value specified for this option is ignored.</i>

Configuration	Cas d'utilisation
<u>celery.worker_autoscale</u> La simultanéité des tâches pour les travailleurs. Défauts : • mw1.micro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mW1. Large - 20,0 • mw1.xlarge - 40,0 • mW1,2xlarge - 80,0	Vous pouvez utiliser cette option pour libérer des ressources en réduisant la maximum simultanéité des minimum tâches entre les travailleurs. Les travailleurs acceptent jusqu'à la maximum limite des tâches simultanées configurées, que les ressources soient suffisantes ou non pour le faire. Si les tâches sont planifiées sans ressources suffisantes, elles échouent immédiatement. Nous recommandons de modifier cette valeur pour les tâches gourmandes en ressources en réduisant les valeurs à des valeurs inférieures aux valeurs par défaut afin d'augmenter la capacité par tâche.

Gestion des dépendances Python dans requirements.txt

Cette rubrique explique comment installer et gérer les dépendances Python dans un `requirements.txt` fichier pour un environnement Amazon Managed Workflows for Apache Airflow.

Table des matières

- [Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI](#)
- [Installation de dépendances Python à l'aide du format de fichier d'exigences PyPi .org](#)
 - [Option 1 : dépendances Python à partir de l'index des packages Python](#)
 - [Deuxième option : roues Python \(.whl\)](#)
 - [Utilisation du plugins.zip fichier dans un compartiment Amazon S3](#)
 - [Utilisation d'un fichier WHL hébergé sur une URL](#)
 - [Création d'un fichier WHL à partir d'un DAG](#)
 - [Troisième option : dépendances Python hébergées sur un dépôt privé conforme à PyPi / PEP-503](#)

- [Activation des journaux sur la console Amazon MWAA](#)
- [Accès aux journaux sur la console CloudWatch Logs](#)
- [Erreurs d'accès dans l'interface utilisateur d'Apache Airflow](#)
 - [Connectez-vous à Apache Airflow](#)
- [Exemples de requirements.txt scénarios](#)

Tests DAGs à l'aide de l'utilitaire Amazon MWAA CLI

- L'utilitaire d'interface de ligne de commande (CLI) reproduit localement un environnement Amazon Managed Workflows pour Apache Airflow.
- La CLI crée localement une image de conteneur Docker similaire à une image de production Amazon MWAA. Vous pouvez l'utiliser pour exécuter un environnement Apache Airflow local afin de développer et de tester DAGs des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-docker-images](#) sur GitHub.

Installation de dépendances Python à l'aide du format de fichier d'exigences PyPi .org

La section suivante décrit les différentes manières d'installer les dépendances Python conformément au [format de fichier d'exigences PyPi .org](#).

Option 1 : dépendances Python à partir de l'index des packages Python

La section suivante décrit comment spécifier les dépendances Python à partir de l'[index des packages Python](#) dans un `requirements.txt` fichier.

Apache Airflow v3

1. Testez localement. Ajoutez des bibliothèques supplémentaires de manière itérative pour trouver la bonne combinaison de packages et de leurs versions, avant de créer un `requirements.txt` fichier. Pour exécuter l'utilitaire Amazon MWAA CLI, reportez-vous à [aws-mwaa-docker-images](#) sur GitHub.
2. Consultez les suppléments du package Apache Airflow. Pour accéder à la liste des packages installés pour Apache Airflow v3 sur Amazon MWAA, consultez le site Web. [aws-mwaa-docker-images requirements.txt](#) GitHub

3. Ajoutez une déclaration de contraintes. Ajoutez le fichier de contraintes pour votre environnement Apache Airflow v3 en haut de votre fichier `requirements.txt`. Les fichiers de contraintes d'Apache Airflow spécifient les versions des fournisseurs disponibles au moment de la publication d'Apache Airflow.

Dans l'exemple suivant, remplacez `{environment-version}` par le numéro de version de votre environnement et `{Python-version}` par la version de Python compatible avec votre environnement.

Pour plus d'informations sur la version de Python compatible avec votre environnement Apache Airflow, reportez-vous à la section Versions d'[Apache Airflow](#).

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

Si le fichier de contraintes détermine que le `xyz==1.0` package n'est pas compatible avec les autres packages de votre environnement `pip3 install`, il n'empêche pas l'installation de bibliothèques incompatibles dans votre environnement. Si l'installation d'un package échoue, vous pouvez accéder aux journaux d'erreurs de chaque composant Apache Airflow (le planificateur, le programme de travail et le serveur Web) dans le flux de journal correspondant sur Logs. CloudWatch Pour plus d'informations sur les types de journaux, reportez-vous à [the section called "Accès aux journaux Airflow"](#).

4. Paquets Apache Airflow. Ajoutez les [extras du package](#) et la version (`==`). Cela permet d'éviter que des packages portant le même nom, mais dont la version est différente, ne soient installés sur votre environnement.

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

5. Bibliothèques Python. Ajoutez le nom du package et la version (`==`) dans votre `requirements.txt` fichier. Cela permet d'éviter qu'une future mise à jour de rupture de [PyPi.org](#) ne soit automatiquement appliquée.

```
library == version
```

Exemple Boto3 et psycpg2-binary

Cet exemple est fourni à des fins de démonstration. Les bibliothèques boto et psycpg2-binary sont incluses dans l'installation de base d'Apache Airflow v3 et n'ont pas besoin d'être spécifiées dans un fichier `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycpg2-binary==2.8.6
```

Si un package est spécifié sans version, Amazon MWAA installe la dernière version du package depuis PyPi .org. Cette version peut entrer en conflit avec les autres packages de votre `requirements.txt`.

Apache Airflow v2

1. Testez localement. Ajoutez des bibliothèques supplémentaires de manière itérative pour trouver la bonne combinaison de packages et de leurs versions, avant de créer un `requirements.txt` fichier. Pour exécuter l'utilitaire Amazon MWAA CLI, reportez-vous à [aws-mwaa-docker-images](#) sur GitHub.
2. Consultez les suppléments du package Apache Airflow. Pour accéder à la liste des packages installés pour Apache Airflow v2 sur Amazon MWAA, rendez-vous [aws-mwaa-docker-images requirements.txt](#) sur le site Web. GitHub
3. Ajoutez une déclaration de contraintes. Ajoutez le fichier de contraintes pour votre environnement Apache Airflow v2 en haut de votre `requirements.txt` fichier. Les fichiers de contraintes d'Apache Airflow spécifient les versions des fournisseurs disponibles au moment de la publication d'Apache Airflow.

À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. `--constraint` Si vous ne fournissez aucune contrainte, Amazon MWAA vous en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Dans l'exemple suivant, remplacez `{environment-version}` par le numéro de version de votre environnement et `{Python-version}` par la version de Python compatible avec votre environnement.

Pour plus d'informations sur la version de Python compatible avec votre environnement Apache Airflow, reportez-vous à la section Versions d'[Apache Airflow](#).

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

Si le fichier de contraintes détermine que le `xyz==1.0` package n'est pas compatible avec les autres packages de votre environnement `pip3 install`, il n'empêche pas l'installation de bibliothèques incompatibles dans votre environnement. Si l'installation d'un package échoue, vous pouvez accéder aux journaux d'erreurs de chaque composant Apache Airflow (le planificateur, le programme de travail et le serveur Web) dans le flux de journal correspondant sur Logs. CloudWatch Pour plus d'informations sur les types de journaux, reportez-vous à [the section called "Accès aux journaux Airflow"](#).

4. Paquets Apache Airflow. Ajoutez les [extras du package](#) et la version (`==`). Cela permet d'éviter que des packages portant le même nom, mais dont la version est différente, ne soient installés sur votre environnement.

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

5. Bibliothèques Python. Ajoutez le nom du package et la version (`==`) dans votre `requirements.txt` fichier. Cela permet d'éviter qu'une future mise à jour de rupture de [PyPi.org](#) ne soit automatiquement appliquée.

```
library == version
```

Exemple Boto3 et psycopg2-binary

Cet exemple est fourni à des fins de démonstration. Les bibliothèques `boto` et `psycopg2-binary` sont incluses dans l'installation de base d'Apache Airflow v2 et n'ont pas besoin d'être spécifiées dans un fichier `requirements.txt`

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

Si un package est spécifié sans version, Amazon MWAA installe la dernière version du package depuis PyPi .org. Cette version peut entrer en conflit avec les autres packages de `votrerrequirements.txt`.

Deuxième option : roues Python (.whl)

Une roue Python est un format de package conçu pour expédier des bibliothèques contenant des artefacts compilés. Les packages Wheel présentent plusieurs avantages en tant que méthode d'installation de dépendances dans Amazon MWAA :

- Installation plus rapide : les fichiers WHL sont copiés dans le conteneur sous la forme d'un seul fichier ZIP, puis installés localement, sans qu'il soit nécessaire de les télécharger.
- Moins de conflits : vous pouvez déterminer à l'avance la compatibilité des versions de vos packages. Par conséquent, il n'est pas nécessaire de trouver pip de manière récursive des versions compatibles.
- Plus de résilience : avec les bibliothèques hébergées en externe, les exigences en aval peuvent changer, ce qui entraîne une incompatibilité des versions entre les conteneurs d'un environnement Amazon MWAA. En ne dépendant pas d'une source externe pour les dépendances, chaque conteneur possède les mêmes bibliothèques, quel que soit le moment où chaque conteneur est instancié.

Nous recommandons les méthodes suivantes pour installer les dépendances Python à partir d'une archive Python Wheel (.whl) dans `votrerrequirements.txt`.

Méthodes

- [Utilisation du plugins.zip fichier dans un compartiment Amazon S3](#)
- [Utilisation d'un fichier WHL hébergé sur une URL](#)
- [Création d'un fichier WHL à partir d'un DAG](#)

Utilisation du **plugins.zip** fichier dans un compartiment Amazon S3

Le planificateur, les outils de traitement et le serveur Web d'Apache Airflow (pour Apache Airflow v2.2.2 et versions ultérieures) recherchent des plugins personnalisés lors du démarrage sur le conteneur Fargate géré pour votre environnement à l'adresse. `AWS/usr/local/airflow/plugins/` * Ce processus commence avant Amazon MWAA `pip3 install -r`

`requirements.txt` pour les dépendances Python et le démarrage du service Apache Airflow. Un `plugins.zip` fichier peut être utilisé pour tous les fichiers que vous ne souhaitez pas modifier en permanence pendant l'exécution de l'environnement ou pour lesquels vous ne souhaitez pas accorder l'accès aux utilisateurs qui écrivent DAGs. Par exemple, les fichiers de roue de la bibliothèque Python, les fichiers PEM de certificats et les fichiers de configuration YAML.

La section suivante décrit comment installer une roue qui se trouve dans le `plugins.zip` fichier de votre compartiment Amazon S3.

1. Téléchargez les fichiers WHL nécessaires Vous pouvez les utiliser [pip download](#) avec votre conteneur Amazon MWAA existant [aws-mwaa-docker-images](#) ou `requirements.txt` sur un autre conteneur [Amazon Linux 2](#) pour résoudre et télécharger les fichiers Python Wheel nécessaires.

```
pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
cd "$AIRFLOW_HOME/plugins"
zip "$AIRFLOW_HOME/plugins.zip" *
```

2. Spécifiez le chemin dans votre **`requirements.txt`**. Spécifiez le répertoire des plugins en haut de votre fichier `requirements.txt` en utilisant [--find-links](#) et demandez de pip ne pas les installer à partir d'autres sources en utilisant [--no-index](#), comme indiqué dans le code suivant :

```
--find-links /usr/local/airflow/plugins
--no-index
```

Exemple roue dans `requirements.txt`

L'exemple suivant suppose que vous avez chargé la roue dans un `plugins.zip` fichier à la racine de votre compartiment Amazon S3. Par exemple :

```
--find-links /usr/local/airflow/plugins
--no-index

numpy
```

Amazon MWAA récupère la `numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl` roue `plugins` dans le dossier et l'installe dans votre environnement.

Utilisation d'un fichier WHL hébergé sur une URL

La section suivante décrit comment installer une roue hébergée sur une URL. L'URL doit être accessible au public ou accessible depuis le VPC Amazon personnalisé que vous avez spécifié pour votre environnement Amazon MWAA.

- Fournissez une URL. Fournissez l'URL d'une roue dans `votrequirements.txt`.

Exemple archive de roues sur une URL publique

L'exemple suivant télécharge une roue depuis un site public.

```
--find-links https://files.pythonhosted.org/packages/  
--no-index
```

Amazon MWAA récupère la roue à partir de l'URL que vous avez spécifiée et l'installe dans votre environnement.

Note

URLs ne sont pas accessibles à partir de serveurs Web privés, exigences d'installation dans Amazon MWAA v2.2.2 et versions ultérieures.

Création d'un fichier WHL à partir d'un DAG

Si vous avez un serveur Web privé utilisant Apache Airflow v2.2.2 ou version ultérieure et que vous ne parvenez pas à installer les exigences car votre environnement n'a pas accès à des référentiels externes, vous pouvez utiliser le DAG suivant pour prendre vos exigences Amazon MWAA existantes et les intégrer à Amazon S3 :

```
from airflow import DAG  
from airflow.operators.bash_operator import BashOperator  
from airflow.utils.dates import days_ago  
  
S3_BUCKET = 'my-s3-bucket'  
S3_KEY = 'backup/plugins_whl.zip'  
  
with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,  
start_date=days_ago(1)) as dag:  
cli_command = BashOperator(  

```

```
task_id="bash_command",
bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/requirements/
requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3 cp /tmp/
plugins.zip s3://amzn-s3-demo-bucket/{S3_KEY}"
)
```

Après avoir exécuté le DAG, utilisez ce nouveau fichier comme Amazon MWAAplugins.zip, éventuellement, empaqueté avec d'autres plugins. Ensuite, mettez à jour votre requirements.txt liste précédée par --find-links /usr/local/airflow/plugins --no-index ou sans ajout--constraint.

Vous pouvez utiliser cette méthode pour utiliser les mêmes bibliothèques hors ligne.

Troisième option : dépendances Python hébergées sur un dépôt privé conforme à PyPi /PEP-503

La section suivante décrit comment installer un Apache Airflow extra hébergé sur une URL privée avec authentification.

1. Ajoutez votre nom d'utilisateur et votre mot de passe comme options de [configuration d'Apache Airflow](#). Par exemple :
 - foo.user : *YOUR_USER_NAME*
 - foo.pass : *YOUR_PASSWORD*
2. Créez votre requirements.txt dossier. Dans l'exemple suivant, remplacez les espaces réservés par votre URL privée, ainsi que par le nom d'utilisateur et le mot de passe que vous avez ajoutés comme options de configuration d'[Apache Airflow](#). Par exemple :

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
```

3. Ajoutez des bibliothèques supplémentaires à votre requirements.txt fichier. Par exemple :

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
my-private-package==1.2.3
```

Activation des journaux sur la console Amazon MWAA

Le [rôle d'exécution](#) de votre environnement Amazon MWAA nécessite une autorisation pour envoyer des CloudWatch journaux à Logs. Pour mettre à jour les autorisations d'un rôle d'exécution, reportez-vous à [Rôle d'exécution Amazon MWAA](#).

Vous pouvez activer les journaux Apache Airflow au niveau INFO, WARNING, ERROR, ou CRITICAL. Lorsque vous choisissez un niveau de journalisation, Amazon MWAA envoie des journaux correspondant à ce niveau et à tous les niveaux de gravité supérieurs. Par exemple, si vous activez les journaux au INFO niveau, Amazon MWAA envoie INFO les journaux et WARNING les niveaux de CRITICAL journalisation à CloudWatch Logs. ERROR Nous recommandons d'activer les journaux Apache Airflow au INFO niveau du planificateur afin qu'il puisse accéder aux journaux reçus pour le `requirements.txt`

Accès aux journaux sur la console CloudWatch Logs

Vous pouvez accéder aux journaux d'Apache Airflow pour le planificateur qui planifie vos flux de travail et analyse votre dossier. dags Les étapes suivantes décrivent comment ouvrir le groupe de journaux pour le planificateur sur la console Amazon MWAA et accéder aux journaux Apache Airflow sur la console Logs. CloudWatch

Pour accéder aux journaux d'un **requirements.txt**

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez le groupe de journaux du planificateur Airflow dans le volet de surveillance.
4. Choisissez le `requirements_install_ip` log in Log streams.
5. Reportez-vous à la liste des packages installés sur l'environnement à l'adresse `/usr/local/airflow/.local/bin`. Par exemple :

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Consultez la liste des packages et vérifiez si l'un d'entre eux a rencontré une erreur lors de l'installation. En cas de problème, vous pouvez obtenir un message d'erreur similaire à ce qui suit :

```
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))
```

Erreurs d'accès dans l'interface utilisateur d'Apache Airflow

Vous pouvez également vérifier l'interface utilisateur d'Apache Airflow pour déterminer si une erreur est liée à un autre problème. L'erreur la plus courante que vous pouvez rencontrer avec Apache Airflow sur Amazon MWAA est la suivante :

```
Broken DAG: No module named x
```

Si vous trouvez cette erreur dans l'interface utilisateur d'Apache Airflow, il vous manque probablement une dépendance requise dans votre `requirements.txt` fichier.

Connectez-vous à Apache Airflow

Vous avez besoin d'[Politique d'accès à l'interface utilisateur d'Apache Airflow : Amazon MWAAWeb ServerAccess](#) autorisations pour que votre Compte AWS identifiant Gestion des identités et des accès AWS (IAM) puisse accéder à votre interface utilisateur Apache Airflow.

Pour accéder à votre interface utilisateur Apache Airflow

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Open Airflow UI.

Exemples de **requirements.txt** scénarios

Vous pouvez mélanger et assortir différents formats dans votre `requirements.txt`. L'exemple suivant utilise une combinaison des différentes méthodes pour installer des options supplémentaires.

Exemple Des suppléments sur PyPi .org et une URL publique

Vous devez utiliser `--index-url` cette option lorsque vous spécifiez des packages provenant de PyPi .org, en plus des packages sur une URL publique, tels que le dépôt URLs personnalisé conforme à la norme PEP 503.

```
aws-batch == 0.6
    phoenix-letter >= 0.3

--index-url http://dist.repoze.org/zope2/2.10/simple
zopelib
```

Surveillance et métriques pour Amazon Managed Workflows pour Apache Airflow

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon Managed Workflows pour Apache Airflow et votre AWS solution. Nous vous recommandons de collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Cette rubrique décrit les ressources disponibles AWS pour surveiller votre environnement Amazon MWAA et répondre à des événements potentiels.

Note

Les métriques et la journalisation d'Apache Airflow sont soumises à la tarification standard d'[Amazon CloudWatch](#).

Pour plus d'informations sur la surveillance d'Apache Airflow, reportez-vous à la section [Logging & Monitoring](#) du site Web de documentation d'Apache Airflow.

Sections

- [Vue d'ensemble de la surveillance sur Amazon MWAA](#)
- [L'accès aux journaux d'audit se connecte AWS CloudTrail](#)
- [Accès aux journaux Airflow sur Amazon CloudWatch](#)
- [Tableaux de bord de surveillance et alarmes sur Amazon MWAA](#)
- [Métriques de l'environnement Apache Airflow dans CloudWatch](#)
- [Mesures relatives aux conteneurs, aux files d'attente et aux bases de données pour Amazon MWAA](#)

Vue d'ensemble de la surveillance sur Amazon MWAA

Cette page décrit les AWS services utilisés pour surveiller un environnement Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [CloudWatch Présentation d'Amazon](#)

- [Présentation de AWS CloudTrail](#)

CloudWatch Présentation d'Amazon

CloudWatch est un référentiel de métriques pour les AWS services que vous pouvez utiliser pour récupérer des statistiques basées sur les [métriques et les dimensions](#) publiées par un service. Vous pouvez utiliser ces métriques pour configurer des [alarmes](#), calculer des statistiques, puis présenter les données dans un [tableau de bord](#) qui vous permet d'évaluer l'état de votre environnement dans la CloudWatch console Amazon.

Apache Airflow est déjà configuré pour envoyer à Amazon des métriques [StatsD](#) pour un environnement Amazon Managed Workflows for Apache Airflow. CloudWatch

Pour en savoir plus, reportez-vous à [Qu'est-ce qu'Amazon CloudWatch ?](#).

Présentation de AWS CloudTrail

CloudTrail est un service d'audit qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon MWAA. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Amazon MWAA, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires disponibles dans les journaux d'audit.

Pour en savoir plus, reportez-vous à [Qu'est-ce que c'est AWS CloudTrail ?](#).

L'accès aux journaux d'audit se connecte AWS CloudTrail

AWS CloudTrail est activé sur votre ordinateur Compte AWS lorsque vous le créez. CloudTrail enregistre l'activité effectuée par une entité ou un AWS service IAM, tel qu'Amazon Managed Workflows pour Apache Airflow, qui est enregistrée en tant qu' CloudTrail événement. Vous pouvez consulter, rechercher et télécharger l'historique des événements des 90 derniers jours dans la CloudTrail console. CloudTrail capture tous les événements sur la console Amazon MWAA et tous les appels vers Amazon APIs MWAA. Il ne capture pas les actions en lecture seule, telles que `GetEnvironment`, ou `PublishMetrics` action. Cette page décrit comment l'utiliser CloudTrail pour surveiller les événements pour Amazon MWAA.

Table des matières

- [Création d'un parcours dans CloudTrail](#)

- [Accès aux événements avec l'historique des CloudTrail événements](#)
- [Exemple de parcours pour CreateEnvironment](#)
- [Quelle est la prochaine étape ?](#)

Création d'un parcours dans CloudTrail

Vous devez créer un parcours pour accéder à un enregistrement permanent des événements de votre site Compte AWS, y compris les événements relatifs à Amazon MWAA. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Si vous ne créez pas de trace, vous pouvez toujours accéder à l'historique des événements disponibles dans la CloudTrail console. Par exemple, à l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Amazon MWAA, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires. Pour en savoir plus, reportez-vous à la section [Création d'un parcours pour votre Compte AWS](#).

Accès aux événements avec l'historique des CloudTrail événements

Vous pouvez résoudre les incidents opérationnels et de sécurité survenus au cours des 90 derniers jours dans la CloudTrail console en consultant l'historique des événements. Par exemple, vous pouvez accéder aux événements liés à la création, à la modification ou à la suppression de ressources (telles que des utilisateurs IAM ou d'autres AWS ressources) dans votre Compte AWS région par région. Pour en savoir plus, reportez-vous à la section [Accès aux événements grâce à l'historique des CloudTrail événements](#).

1. Ouvrez la console [CloudTrail](#).
2. Choisissez Historique des événements.
3. Sélectionnez les événements que vous souhaitez consulter, puis sélectionnez Comparer les détails des événements.

Exemple de parcours pour **CreateEnvironment**

Un journal de suivi est une configuration qui permet la livraison d'événements sous forme de fichiers journaux vers un compartiment Amazon S3 que vous spécifiez.

CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations

sur l'action demandée, telles que la date et l'heure de l'action ou les paramètres de la demande. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics et ne sont pas répertoriés dans un ordre spécifique. L'exemple suivant est une entrée de journal pour l'`CreateEnvironment` action refusée en raison de l'absence d'autorisations. Les valeurs indiquées `AirflowConfigurationOptions` ont été supprimées pour des raisons de confidentialité.

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

Quelle est la prochaine étape ?

- Découvrez comment configurer d'autres AWS services pour les données d'événements collectées dans les CloudTrail journaux dans [Services et intégrations CloudTrail pris en charge](#).
- Découvrez comment être averti lors de la CloudTrail publication de nouveaux fichiers journaux dans un compartiment Amazon S3 dans [Configuration des notifications Amazon SNS](#) pour CloudTrail

Accès aux journaux Airflow sur Amazon CloudWatch

Amazon MWAA peut envoyer les journaux Apache Airflow à Amazon CloudWatch. Vous pouvez accéder aux journaux de plusieurs environnements à partir d'un seul emplacement afin d'identifier facilement les retards de tâches ou les erreurs de flux de travail d'Apache Airflow sans outils tiers supplémentaires. Les journaux Apache Airflow doivent être activés sur la console Amazon Managed Workflows for Apache Airflow pour accéder au traitement, aux tâches, au serveur Web et aux connexions des travailleurs d'Apache Airflow DAG. CloudWatch

Table des matières

- [Tarification](#)
- [Avant de commencer](#)

- [Types de journaux](#)
- [Activation des journaux Apache Airflow](#)
- [Accès aux journaux d'Apache Airflow](#)
- [Exemples de journaux du planificateur](#)
- [Quelle est la prochaine étape ?](#)

Tarification

- CloudWatch Les frais de journalisation standard s'appliquent. Pour plus d'informations, reportez-vous à la section [CloudWatch Tarification](#).

Avant de commencer

- Vous devez disposer d'un rôle permettant d'accéder aux connexions CloudWatch. Pour plus d'informations, consultez [Accès à un environnement Amazon MWAA](#).

Types de journaux

Amazon MWAA crée un groupe de journaux pour chaque option de journalisation Airflow que vous activez et transmet les journaux aux groupes de CloudWatch journaux associés à un environnement. Les groupes de journaux sont nommés au format suivant :**YourEnvironmentName-LogType**. Par exemple, si votre environnement est nommé **Airflow-v202-Public**, les journaux des tâches d'Apache Airflow sont envoyés à **Airflow-v202-Public-Task**.

Type de journal	Description
YourEnvironmentName- DAGProcessing	Les journaux du gestionnaire du processeur DAG (partie du planificateur qui traite les fichiers DAG).
YourEnvironmentName- Scheduler	Les journaux générés par le planificateur Airflow.
YourEnvironmentName- Task	Les journaux de tâches générés par un DAG.

Type de journal	Description
YourEnvironmentName- WebServer	Les journaux générés par l'interface Web Airflow.
YourEnvironmentName- Worker	Les journaux générés dans le cadre du flux de travail et de l'exécution du DAG.

Activation des journaux Apache Airflow

Vous pouvez activer les journaux Apache Airflow au niveau INFO, WARNING, ERROR, ou CRITICAL. Lorsque vous choisissez un niveau de journalisation, Amazon MWAA envoie des journaux correspondant à ce niveau et à tous les niveaux de gravité supérieurs. Par exemple, si vous activez les journaux au INFO niveau, Amazon MWAA envoie INFO les journaux et WARNING les niveaux de CRITICAL journalisation à CloudWatch Logs. ERROR

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez Modifier.
4. Choisissez Suivant.
5. Choisissez une ou plusieurs des options de journalisation suivantes :
 - a. Choisissez le groupe de journaux du planificateur Airflow dans le volet de surveillance.
 - b. Choisissez le groupe de journaux du serveur Web Airflow dans le volet de surveillance.
 - c. Choisissez le groupe de journaux de travail Airflow dans le volet de surveillance.
 - d. Choisissez le groupe de journaux de traitement Airflow DAG dans le volet Surveillance.
 - e. Choisissez le groupe de journaux de tâches Airflow dans le volet Surveillance.
 - f. Choisissez le niveau de journalisation dans Log level.
6. Choisissez Suivant.
7. Choisissez Enregistrer.

Accès aux journaux d'Apache Airflow

La section suivante décrit comment accéder aux journaux Apache Airflow dans la CloudWatch console.

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez un groupe de journaux dans le volet de surveillance.
4. Choisissez un flux de journal de connexion.

Exemples de journaux du planificateur

Vous pouvez accéder aux journaux d'Apache Airflow pour le planificateur qui planifie vos flux de travail et analyse votre dossier. dags Les étapes suivantes décrivent comment ouvrir le groupe de journaux pour le planificateur sur la console Amazon MWAA et accéder aux journaux Apache Airflow sur la console Logs. CloudWatch

Pour accéder aux journaux d'un **requirements.txt**

1. Ouvrez la page [Environnements](#) sur la console Amazon MWAA.
2. Choisissez un environnement.
3. Choisissez le groupe de journaux du planificateur Airflow dans le volet de surveillance.
4. Choisissez le requirements_install_ip log in Log streams.
5. Reportez-vous à la liste des packages installés sur l'environnement à l'adresse/usr/local/airflow/.local/bin. Par exemple :

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Consultez la liste des packages et vérifiez si l'un d'entre eux a rencontré une erreur lors de l'installation. En cas de problème, vous risquez de recevoir un message d'erreur similaire à ce qui suit :

```
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))
```

Quelle est la prochaine étape ?

- Découvrez comment configurer une CloudWatch alarme dans [Utiliser les CloudWatch alarmes Amazon](#).
- Découvrez comment créer un CloudWatch tableau de bord dans [Utilisation des CloudWatch tableaux de bord](#).

Tableaux de bord de surveillance et alarmes sur Amazon MWAA

Vous pouvez créer un tableau de bord personnalisé dans Amazon CloudWatch et ajouter des alarmes pour une métrique particulière afin de surveiller l'état de santé d'un environnement Amazon Managed Workflows for Apache Airflow. Lorsqu'une alarme apparaît sur un tableau de bord, elle devient rouge lorsqu'elle est en ALARM état, ce qui permet de surveiller plus facilement l'état de santé d'un environnement Amazon MWAA de manière proactive.

Apache Airflow expose les métriques de plusieurs processus, notamment le nombre de processus DAG, la taille du sac DAG, les tâches en cours d'exécution, les échecs et les réussites des tâches. Lorsque vous créez un environnement, Airflow envoie automatiquement les métriques d'un environnement Amazon MWAA à CloudWatch. Cette page explique comment créer un tableau de bord d'état de santé pour les métriques Airflow dans un CloudWatch environnement Amazon MWAA.

Table des matières

- [Métriques](#)
- [Vue d'ensemble des états d'alarme](#)
- [Exemples de tableaux de bord et d'alarmes personnalisés](#)
 - [À propos de ces indicateurs](#)
 - [À propos du tableau de bord](#)
 - [Utilisation de AWS didacticiels](#)
 - [En utilisant CloudFormation](#)
- [Supprimer des métriques et des tableaux de bord](#)

- [Quelle est la prochaine étape ?](#)

Métriques

Vous pouvez créer un tableau de bord personnalisé et une alarme pour toutes les métriques disponibles pour votre version d'Apache Airflow. Chaque métrique correspond à un indicateur de performance clé (KPI) d'Apache Airflow. Pour accéder à une liste de mesures, reportez-vous à :

- [Métriques de l'environnement Apache Airflow dans CloudWatch](#)

Vue d'ensemble des états d'alarme

Une alerte de métrique peut avoir les états suivants :

- OK – La métrique ou l'expression se trouve dans le seuil défini.
- ALARM – La métrique ou l'expression se trouve à l'extérieur du seuil défini.
- INSUFFICIENT_DATA – L'alerte vient de commencer, la métrique n'est pas disponible, ou la quantité de données n'est pas suffisante pour permettre à la métrique de déterminer le statut de l'alerte.

Exemples de tableaux de bord et d'alarmes personnalisés

Vous pouvez créer un tableau de bord de surveillance personnalisé qui affiche des graphiques des mesures sélectionnées pour votre environnement Amazon MWAA.

À propos de ces indicateurs

La liste suivante décrit chacune des métriques créées dans le tableau de bord personnalisé par le didacticiel et les définitions de modèles de cette section.

- QueuedTasks- Le nombre de tâches mises en file d'attente. Correspond à la `executor.queued_tasks` métrique Apache Airflow.
- TasksPending- Le nombre de tâches en attente dans l'exécuteur. Correspond à la `scheduler.tasks.pending` métrique Apache Airflow.

Note

Ne s'applique pas à Apache Airflow v2.2 et versions ultérieures.

- **RunningTasks**- Le nombre de tâches exécutées dans l'exécuteur. Correspond à la `executor.running_tasks` métrique Apache Airflow.
- **SchedulerHeartbeat**- Le nombre d'enregistrements effectués par Apache Airflow dans le cadre de la tâche du planificateur. Correspond aux métriques d'`scheduler_heartbeat` Apache Airflow.
- **TotalParseTime**- Le nombre de secondes nécessaires pour scanner et importer tous les fichiers DAG une fois. Correspond à la `dag_processing.total_parse_time` métrique Apache Airflow.

À propos du tableau de bord

L'image suivante montre le tableau de bord de surveillance créé par le didacticiel et la définition du modèle de cette section.

Utilisation de AWS didacticiels

Vous pouvez utiliser le AWS didacticiel suivant pour créer automatiquement un tableau de bord d'état de santé pour tous les environnements Amazon MWAA actuellement déployés. Il génère également des CloudWatch alarmes en cas de mauvais état des travailleurs et de défaillances du rythme cardiaque du planificateur dans tous les environnements Amazon MWAA.

- [CloudWatch Automatisation du tableau de bord pour Amazon MWAA](#)

En utilisant CloudFormation

Vous pouvez utiliser la définition du CloudFormation modèle présentée dans cette section pour créer un tableau de bord de surveillance CloudWatch, puis ajouter des alarmes sur la CloudWatch console pour recevoir des notifications lorsqu'une métrique dépasse un certain seuil. Pour créer la pile à l'aide de cette définition de modèle, reportez-vous à la section [Création d'une pile sur la CloudFormation console](#). Pour ajouter une alarme au tableau de bord, reportez-vous à la section [Utilisation des alarmes](#).

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Creates MWAA Cloudwatch Dashboard
```


Parameters:**DashboardName:**

Description: Enter the name of the CloudWatch Dashboard

Type: String

EnvironmentName:

Description: Enter the name of the MWA 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": [
          [
            "AmazonMWA",
            "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
      }
    }
  ]
}'
```

Supprimer des métriques et des tableaux de bord

Si vous supprimez un environnement Amazon MWAA, le tableau de bord correspondant est également supprimé. CloudWatch les métriques sont conservées pendant quinze (15) mois et ne peuvent pas être supprimées. La CloudWatch console limite la recherche de métriques à deux (2) semaines après la dernière ingestion d'une métrique afin de garantir que les instances les plus récentes soient affichées pour votre environnement Amazon MWAA. Pour en savoir plus, consultez [Amazon CloudWatch FAQs](#).

Quelle est la prochaine étape ?

- Découvrez comment créer un DAG qui interroge la base de données de métadonnées Amazon Aurora PostgreSQL pour votre environnement et y publie des métriques personnalisées. CloudWatch [Utilisation d'un DAG pour écrire des métriques personnalisées dans CloudWatch](#)

Métriques de l'environnement Apache Airflow dans CloudWatch

Apache Airflow v2 et v3 sont déjà configurés pour collecter et envoyer à Amazon des [métriques StatsD](#) pour un environnement Amazon Managed Workflows for Apache Airflow. CloudWatch La liste complète des métriques envoyées par Apache Airflow est disponible sur la page [Metrics du guide](#) de référence d'Apache Airflow. Cette page décrit les métriques Apache Airflow disponibles dans la CloudWatch console et explique comment y accéder. CloudWatch

Table des matières

- [Conditions](#)
- [Dimensions](#)
- [Accès aux métriques dans la CloudWatch console](#)
- [Les métriques d'Apache Airflow sont disponibles dans CloudWatch](#)
 - [Compteurs Apache Airflow](#)
 - [Jauges Apache Airflow](#)

- [Minuteries Apache Airflow](#)
- [Choix des indicateurs à signaler](#)
- [Quelle est la prochaine étape ?](#)

Conditions

Namespace

Un espace de noms est un conteneur pour les CloudWatch métriques d'un AWS service. Pour Amazon MWAA, l'espace de noms est AmazonMWAA.

CloudWatch métriques

Une CloudWatch métrique représente un ensemble chronologique de points de données spécifiques à CloudWatch.

Métriques d'Apache Airflow

Les [métriques](#) spécifiques à Apache Airflow.

Dimension

Une dimension est une name/value paire qui fait partie de l'identité d'une métrique.

Unit

Une statistique possède une unité de mesure. Pour Amazon MWAA, les unités incluent le nombre, les secondes et les millisecondes. Pour Amazon MWAA, les unités sont définies en fonction des unités indiquées dans les métriques Airflow d'origine.

Dimensions

Cette section décrit le regroupement de CloudWatch dimensions pour les métriques Apache Airflow dans. CloudWatch

Dimension	Description			
JOUR	Indique un nom de DAG Apache Airflow spécifique.			

Dimension	Description			
Nom de fichier DAG	Indique un nom de fichier DAG Apache Airflow spécifique.			
Fonction	Cette dimension est utilisée pour améliorer le regroupement des métriques dans CloudWatch.			
Tâche	Indique une tâche Apache Airflow exécutée par le planificateur. A toujours une valeur deJob.			
Opérateur	Indique un opérateur Apache Airflow spécifique.			
Pool	Indique un pool de travailleurs Apache Airflow spécifique.			
Sous-tâche	Indique une tâche Apache Airflow spécifique.			
HostName	Indique le nom d'hôte d'un processus Apache Airflow spécifique en cours d'exécution.			

Accès aux métriques dans la CloudWatch console

Cette section décrit comment accéder aux métriques de performance CloudWatch pour un DAG spécifique.

Pour accéder aux indicateurs de performance d'une dimension

1. Ouvrez la [page Metrics](#) sur la CloudWatch console.
2. Sélectionnez votre Région AWS.
3. Choisissez l'espace de noms AmazonMWAA.
4. Dans l'onglet Toutes les mesures, sélectionnez une dimension. Par exemple, DAG, Environnement.
5. Choisissez une CloudWatch métrique pour une dimension. Par exemple, TaskInstanceSuccesses ou TaskInstanceDuration. Choisissez Représenter graphiquement tous les résultats de recherche.
6. Choisissez l'onglet Graphed metrics pour accéder aux statistiques de performance des métriques Apache Airflow, telles que DAG, Environment, Task.

Les métriques d'Apache Airflow sont disponibles dans CloudWatch

Cette section décrit les métriques et les dimensions d'Apache Airflow envoyées à CloudWatch

Compteurs Apache Airflow

Les métriques Apache Airflow présentées dans cette section contiennent des données sur les compteurs [Apache Airflow](#).

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
SLAMissed	sla_missed	Nombre	Fonction, planificateur	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
 Note Disponible uniquement pour Apache Airflow v2.4.3 à v2.10.3.				
Échoué SLACallback	sla_callback_notification_failure	Nombre	Fonction, planificateur	
 Note Disponible uniquement pour Apache Airflow v2.4.3 à v2.10.3.				
Mises à jour	jeu de données.mises à jour	Nombre	Fonction, planificateur	
 Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.				
Orphelin	jeu de données orphelin	Nombre	Fonction, planificateur	
 Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.				

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
FailedCeleryTaskExecution  Note Disponible pour Apache Airflow v2.4.3 et versions ultérieures.	celery.execute_command.failure	Nombre	Fonction, céleri	
FilePathQueueUpdateCount  Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.	dag_processing.file_path_queue_update_count	Nombre	Fonction, planificateur	
CriticalSectionBusy	scheduler.critical_section_busy	Nombre	Fonction, planificateur	
DagBagSize	taille_du_sac	Nombre	Fonction, traitement DAG	
DagCallbackExceptions	dag.callback_exceptions	Nombre	DAG, Tous	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
SLAEmailTentatives ratées	échec de notification par e-mail	Nombre	Fonction, planificateur	
 Note Non disponible pour Apache Airflow v3.0.6 et versions ultérieures.				
TaskInstanceFinished	ti.finish. {day_id}. {identifiant de tâche}. {état}	Nombre	DAG, {day_id} Tâche, {task_id} État, {état}	
JobEnd	{job_name} _fin	Nombre	Job, {job_name}	
JobHeartbeatFailure	{job_name} }_heartbea t_failure	Nombre	Job, {job_name}	
JobStart	{nom_tâche} _démarrer	Nombre	Job, {job_name}	
ManagerStalls	dag_proce ssing.man ager_stalls	Nombre	Fonction, traitement DAG	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
OperatorFailures	opérateur_failures_{nom_opérateur}	Nombre	Opérateur, {operator_name}	
OperatorSuccesses	operator_successes_{nom_opérateur}	Nombre	Opérateur, {operator_name}	
OtherCallbackCount	dag_processing.other_callback_count	Nombre	Fonction, planificateur	
Processus	dag_processing.processes	Nombre	Fonction, traitement DAG	
SchedulerHeartbeat	scheduler_heartbeat	Nombre	Fonction, planificateur	
StartedTaskInstances	ti.start.{day_id}. {identifiant de tâche}	Nombre	DAG, Tous Tâche, tout	



Note

Disponible dans Apache Airflow v2.6.3 et versions ultérieures.

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
SlaCallbackCount	dag_processing.sla_callback_count  Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.	Nombre	Fonction, planificateur	
TasksKilledExternally	scheduler.tasks.killed_external	Nombre	Fonction, planificateur	
TaskTimeoutError	celery.task_timeout_error	Nombre	Fonction, céli	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TaskInstanceCreatedUsingOperator	task_instance_created- {nom_opérateur}	Nombre	Opérateur, {operator_name}	
TaskInstancePreviouslySucceeded	précédemment_réussi	Nombre	DAG, Tous Tâche, tout	
TaskInstanceFailures	ti_failures	Nombre	DAG, Tous Tâche, tout	
TaskInstanceSuccesses	ti_success	Nombre	DAG, Tous Tâche, tout	
TaskRemovedFromDAG	task_removed_from_dag. {day_id}	Nombre	DAG, {day_id}	
TaskRestoredToDAG	task_restored_to_day. {day_id}	Nombre	DAG, {day_id}	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TriggersSucceeded  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	déclencheurs réussis	Nombre	Fonction, déclencheur	
TriggersFailed  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	déclencheurs. échec	Nombre	Fonction, déclencheur	
TriggersBlockedMainThread  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	triggers. blocked_main_thread	Nombre	Fonction, déclencheur	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TriggerHeartbeat  Note Disponible pour Apache Airflow v2.8.1 et versions ultérieures.	déclencheur du battement de cœur	Nombre	Fonction, déclencheur	
TaskInstanceCreatedUsingOperator	airflow.task_instance_created_{operator_name}  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	Nombre	Opérateur, {operator_name}	
ZombiesKilled	zombies tués	Nombre	DAG, Tous Tâche, tout	

Jauges Apache Airflow

Les métriques Apache Airflow présentées dans cette section contiennent des données sur les jauges [Apache Airflow](#).

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
DAGFileRe freshError	dag_file_refresh_error	Nombre	Fonction, traitement DAG	
ImportErrors	dag_processing.imp ort_errors	Nombre	Fonction, traitement DAG	
Exception Failures	smart_sensor_opera tor.exception_failures	Nombre	Fonction, opérateur de capteur intelligent	
ExecutedTasks	smart_sensor_opera tor.tâches exécutées	Nombre	Fonction, opérateur de capteur intelligent	
InfraFailures	smart_sensor_opera tor.infra_failures	Nombre	Fonction, opérateur de capteur intelligent	
LoadedTasks	smart_sensor_opera tor.loaded_tasks	Nombre	Fonction, opérateur	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
			de capteur intelligent	
TotalParseTime	dag_processing.total_parse_time	Secondes	Fonction, traitement DAG	
Triggered DagRuns	jeu de données .triggered_dagruns	Nombre	Fonction, planificateur	

 Note

Disponible dans Apache Airflow v2.6.3 et versions ultérieures.

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TriggersRunning	déclencheurs. Exécution. <i>{hostname}</i> <i>}</i>	Nombre	Fonction, déclencheur HostName, <i>{hostname}</i>	
Note Disponibl e dans Apache Airflow v2.7.2 et versions ultérieur es.				
PoolDeferredSlots	pool.deferred_slots. {pool_name}	Nombre	Piscine, {pool_name}	
Note Disponibl e dans Apache Airflow v2.7.2 et versions ultérieur es.				

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
DAGFileProcessingLastRunSecondsAgo	dag_processing.last_run. Il y a quelques secondes. {dag_file_name}	Secondes	Nom du fichier DAG, {dag_file_name}	
OpenSlots	exécuteur.open_slots	Nombre	Fonction, exécuteur	
OrphanedTasksAdopted	scheduler.orphaned_tasks.adopted	Nombre	Fonction, planificateur	
OrphanedTasksCleared	scheduler.orphaned_tasks.cleared	Nombre	Fonction, planificateur	
PokedExceptions	smart_sensor_operator.poked_exception	Nombre	Fonction, opérateur de capteur intelligent	
PokedSuccess	smart_sensor_operator.poked_success	Nombre	Fonction, opérateur de capteur intelligent	
PokedTasks	smart_sensor_operator.poked_tasks	Nombre	Fonction, opérateur de capteur intelligent	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
PoolFailures	pool.open_slots. {nom_piscine}	Nombre	Piscine, {pool_name}	
PoolStarvingTasks	pool.starving_tasks. {nom_piscine}	Nombre	Piscine, {pool_name}	
PoolOpenSlots	pool.open_slots. {nom_piscine}	Nombre	Piscine, {pool_name}	
PoolQueuedSlots	pool.queued_slots. {nom_piscine}	Nombre	Piscine, {pool_name}	
PoolRunningSlots	pool.running_slots. {nom_piscine}	Nombre	Piscine, {pool_name}	
Processor Timeouts	dag_processing.processor_timeouts	Nombre	Fonction, traitement DAG	
QueuedTasks	executor.queued_tasks	Nombre	Fonction, exécuteur	
RunningTasks	exécuteur.running_tasks	Nombre	Fonction, exécuteur	
TasksExecutable	scheduler.tasks_executable	Nombre	Fonction, planificateur	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TasksPending	scheduler.tasks.pending	Nombre	Fonction, planificateur	
 Note Ne s'applique pas à Apache Airflow v2.2 et versions ultérieures.				
TasksRunning	scheduler.tasks.running	Nombre	Fonction, planificateur	
TasksStarving	scheduler.tasks.starving	Nombre	Fonction, planificateur	
TasksWithoutDagRun	scheduler.tasks.without_dagrun	Nombre	Fonction, planificateur	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
DAGFileProcessingLastNumOfDbQueries	dag_processing.last_num_of_db_queries. {dag_filename}	Nombre	Nom du fichier DAG, {dag_filename}	
Note Disponibl e dans Apache Airflow v2.10.1 et versions ultérieur es.				
PoolScheduledSlots	pool.scheduled_slots. {nom_piscine}	Nombre	Piscine, {pool_name}	
Note Disponibl e dans Apache Airflow v2.10.1 et versions ultérieur es.				

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TaskCpuUsage	utilisation du processeur. {day_id}. {identifiant de tâche}	Pourcentage	DAG, {day_id} Tâche, {task_id}	
 Note Disponible dans Apache Airflow v2.10.1 et versions ultérieures.				
TaskMemoryUsage	mem.usage. {day_id}. {identifiant de tâche}	Pourcentage	DAG, {day_id} Tâche, {task_id}	
 Note Disponible dans Apache Airflow v2.10.1 et versions ultérieures.				

Minuteries Apache Airflow

Les métriques d'Apache Airflow présentées dans cette section contiennent des données sur les temporisateurs d'[Apache Airflow](#).

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
Collectez DBDags	collecte_db_dags	Millisecondes	Fonction, traitement DAG	
CriticalS ectionDuration	scheduler .critical_section_ duration	Millisecondes	Fonction, planificateur	
CriticalS ectionQue ryDuration	scheduler .critical_section_ query_duration	Millisecondes	Fonction, planificateur	
 Note Disponibl e pour Apache Airflow v2.5.1 et versions ultérieur es.				
DAGDepend encyVérifiez	dagrun.de pendency-check. {day_id}	Millisecondes	DAG, {day_id}	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
DAGDurationFailed	dagrun.duration.failed. {day_id}	Millisecondes	DAG, {day_id}	
DAGDurationSuccessful	dagrun.duration.success. {day_id}	Millisecondes	DAG, {day_id}	
DAGFileProcessingLastDuration	dag_processing.last_duration. {dag_filename}	Secondes	Nom du fichier DAG, {dag_file name}	
DAGScheduleRetard	dagrun.schedule_delay. {day_id}	Millisecondes	DAG, {day_id}	
FirstTaskSchedulingDelay	dagrun.{dag_id}.first_task_scheduling_delay	Millisecondes	DAG, {day_id}	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
Scheduler LoopDuration	scheduler .schedule r_loop_duration	Millisecondes	Fonction, planificateur	
Note Disponibl e pour Apache Airflow v2.5.1 et versions ultérieur es.				
TaskInsta nceDuration	jour. {day_id}. {task_id} .durée	Millisecondes	DAG, {day_id} Tâche, {task_id}	

CloudWatch métrique	Métrique Apache Airflow	Unit	Dimension	
TaskInstanceQueuedDuration	jour. {dag_id}. {task_id} .queued_duration  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	Millisecondes	DAG, {day_id} Tâche, {task_id}	
TaskInstanceScheduledDuration	jour. {dag_id}. {task_id} .durée_planifiée  Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.	Millisecondes	DAG, {day_id} Tâche, {task_id}	

Choix des indicateurs à signaler

[Vous pouvez choisir les métriques Apache Airflow qui sont émises ou bloquées par Apache Airflow CloudWatch, à l'aide des options de configuration Amazon MWAA suivantes :](#)

- **`metrics.metrics_allow_list`**— Une liste de préfixes séparés par des virgules que vous pouvez utiliser pour sélectionner les métriques vers CloudWatch auxquelles votre environnement émet. Utilisez cette option si vous souhaitez qu'Apache Airflow n'envoie pas toutes les métriques disponibles et sélectionne un sous-ensemble d'éléments. Par exemple, `scheduler,executor,dagrun`.
- **`metrics.metrics_block_list`**— Une liste de préfixes séparés par des virgules pour filtrer les métriques commençant par les éléments de la liste. Par exemple, `scheduler,executor,dagrun`.

Si vous configurez les deux `metrics.metrics_allow_list` et `metrics.metrics_block_list`, Apache Airflow les ignore. `metrics.metrics_block_list` Si vous configurez `metrics.metrics_block_list` mais non `metrics.metrics_allow_list`, Apache Airflow filtre les éléments que vous spécifiez. `metrics.metrics_block_list`

Note

Les options `metrics.metrics_block_list` de configuration `metrics.metrics_allow_list` et s'appliquent uniquement à Apache Airflow v2.6.3 et versions ultérieures. Pour les versions précédentes d'Apache Airflow, utilisez `metrics.statsd_allow_list` et `metrics.statsd_block_list` à la place.

Quelle est la prochaine étape ?

- Découvrez le fonctionnement de l'API Amazon MWAA utilisé pour publier les indicateurs de santé de l'environnement sur [PublishMetrics](#).

Mesures relatives aux conteneurs, aux files d'attente et aux bases de données pour Amazon MWAA

Outre les métriques d'Apache Airflow, vous pouvez surveiller les composants sous-jacents de vos environnements Amazon Managed Workflows pour Apache Airflow à l'aide d'un outil qui collecte des données brutes et les traite pour en faire des métriques lisibles en temps quasi réel. Grâce à ces indicateurs environnementaux, vous bénéficierez d'une meilleure visibilité sur les indicateurs de performance clés pour vous aider à dimensionner correctement vos environnements et à résoudre les problèmes liés à vos flux de travail. Ces métriques s'appliquent à toutes les versions d'Apache Airflow prises en charge sur Amazon MWAA.

Amazon MWAA fournira l'utilisation du processeur et de la mémoire pour chaque conteneur Amazon Elastic Container Service (Amazon ECS) et instance Amazon Aurora PostgreSQL, ainsi que les métriques Amazon Simple Queue Service (Amazon SQS) pour le nombre de messages et l'âge du message le plus ancien, les métriques Amazon Relational Database Service (Amazon RDS) pour la base de données connexions, profondeur de la file d'attente du disque, opérations d'écriture, latence et débit, et métriques du proxy Amazon RDS. Ces indicateurs incluent également le nombre de travailleurs de base, de travailleurs supplémentaires, de planificateurs et de serveurs Web.

Ces statistiques sont conservées pendant 15 mois, afin que vous puissiez accéder aux informations historiques, avoir une meilleure idée des raisons pour lesquelles un calendrier échoue et résoudre les problèmes sous-jacents. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoyer des notifications ou prendre des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le [guide de CloudWatch l'utilisateur Amazon](#).

Rubriques

- [Conditions](#)
- [Dimensions](#)
- [Accès aux métriques dans la CloudWatch console](#)
- [Liste des métriques](#)

Conditions

Namespace

Un espace de noms est un conteneur pour les CloudWatch métriques d'un AWS service. Pour Amazon MWAA, l'espace de noms est. AWS/MWAA

CloudWatch métriques

Une CloudWatch métrique représente un ensemble chronologique de points de données spécifiques à CloudWatch.

Dimension

Une dimension est une name/value paire qui fait partie de l'identité d'une métrique.

Unit

Une statistique possède une unité de mesure. Pour Amazon MWAA, les unités incluent le nombre.

Dimensions

Cette section décrit le regroupement des CloudWatch dimensions pour les métriques Amazon MWAA dans CloudWatch.

Dimension	Description
Cluster	Mesures relatives aux trois conteneurs Amazon ECS minimum utilisés par un environnement Amazon MWAA pour exécuter les composants Apache Airflow : planificateur, serveur de travail et serveur Web.
File d'attente	Mesures pour les files d'attente Amazon SQS qui dissocient le planificateur des travailleurs. Lorsque les travailleurs lisent les messages, ils sont considérés comme étant en vol et ne sont pas disponibles pour les autres travailleurs. Les messages peuvent être lus par les autres

Dimension	Description
	employés s'ils ne sont pas supprimés avant le délai de visibilité de 12 heures.
Base de données	Métriques des clusters Aurora utilisés par Amazon MWAA. Cela inclut des métriques pour l'instance de base de données principale et une réplique de lecture pour prendre en charge les opérations de lecture. Amazon MWAA publie des métriques de base de données pour les instances READER et WRITER.

Accès aux métriques dans la CloudWatch console

Cette section décrit comment accéder à vos métriques Amazon MWAA dans CloudWatch.

Pour accéder aux indicateurs de performance d'une dimension

1. Ouvrez la [page Metrics](#) sur la CloudWatch console.
2. Sélectionnez votre Région AWS.
3. Choisissez l'espace de noms AWS/MWAA.
4. Dans l'onglet Toutes les mesures, choisissez une dimension. Par exemple, Cluster.
5. Choisissez une CloudWatch métrique pour une dimension. Par exemple, NumSchedulers ou CPUUtilization. Choisissez ensuite Représenter graphiquement tous les résultats de recherche.
6. Choisissez l'onglet Graphed metrics pour accéder aux métriques de performance.

Liste des métriques

Les tableaux suivants répertorient les métriques du service de cluster, de file d'attente et de base de données pour Amazon MWAA. Pour accéder aux descriptions des métriques directement émises par Amazon ECS, Amazon SQS ou Amazon RDS, choisissez le lien de documentation correspondant.

Rubriques

- [Métriques du cluster](#)
- [Métriques de base de données](#)

- [Métriques de file d'attente](#)
- [Métriques Application Load Balancer](#)

Métriques du cluster

Les mesures suivantes s'appliquent à chaque planificateur, collaborateur de base, collaborateur supplémentaire et serveur Web. Pour plus d'informations et une description de chaque métrique de cluster, reportez-vous à la section [Mesures et dimensions disponibles](#) dans le manuel Amazon ECS Developer Guide.

Namespace	Métrique	Unit
AWS/MWAA	CPUUtilization	Pourcentage
AWS/MWAA	MemoryUtilization	Pourcentage

Évaluation du nombre de conteneurs de travailleurs et de serveurs Web supplémentaires

Vous pouvez utiliser les métriques des composants fournies dans la dimension Cluster, comme décrit dans la procédure suivante, pour évaluer le nombre de travailleurs supplémentaires, ou de serveurs Web, qu'un environnement utilise à un moment donné. Pour ce faire, vous pouvez représenter graphiquement la métrique CPUUtilization ou la MemoryUtilization métrique et définir le type de statistique sur Sample Count. La valeur résultante est le nombre total de RUNNING tâches pour le AdditionalWorker composant. Comprendre le nombre d'instances de travail supplémentaires utilisées par votre environnement peut vous aider à évaluer l'évolution de votre environnement et à optimiser le nombre de travailleurs supplémentaires.

Workers

Pour évaluer le nombre de travailleurs supplémentaires à l'aide du AWS Management Console

1. Choisissez l'espace de noms AWS/MWAA.
2. Dans l'onglet Toutes les mesures, choisissez la dimension Cluster.
3. Dans la dimension Cluster, pour le AdditionalWorker, choisissez soit la métrique, CPUUtilization soit la MemoryUtilization métrique.

4. Dans l'onglet Mesures graphiques, définissez la période sur 1 minute et la statistique sur le nombre d'échantillons.

webservers

Pour évaluer le nombre de serveurs Web supplémentaires à l'aide du AWS Management Console

1. Choisissez l'espace de noms AWS/MWAA.
2. Dans l'onglet Toutes les mesures, choisissez la dimension Cluster.
3. Dans la dimension Cluster, pour le AdditionalWebservers, choisissez soit la métrique, CPUUtilization soit la MemoryUtilization métrique.
4. Dans l'onglet Mesures graphiques, définissez la période sur 1 minute et la statistique sur le nombre d'échantillons.

Pour plus d'informations, reportez-vous à la section [Nombre de RUNNING tâches de service](#) dans le manuel Amazon Elastic Container Service Developer Guide.

Métriques de base de données

Les mesures suivantes s'appliquent à chaque instance de base de données associée à l'environnement Amazon MWAA.

Namespace	Métrique	Unit
AWS/MWAA	CPUUtilization	Pourcentage
AWS/MWAA	DatabaseConnections	Nombre
AWS/MWAA	DiskQueueDepth	Nombre
AWS/MWAA	FreeableMemory	Octets
AWS/MWAA	VolumeWriteIOPS	Comptez toutes les cinq minutes

Namespace	Métrique	Unit
AWS/MWAA	WriteIOPS	Nombre par seconde
AWS/MWAA	WriteLatency	Secondes
AWS/MWAA	WriteThroughput	Octets par seconde

Métriques de file d'attente

Pour plus d'informations sur les unités et les descriptions des métriques de file d'attente suivantes, reportez-vous à la section [CloudWatch Mesures disponibles pour Amazon SQS](#) dans le manuel Amazon Simple Queue Service Developer Guide.

Namespace	Métrique	Unit
AWS/MWAA	ApproximateAgeOfOldestTask	Secondes
AWS/MWAA	RunningTasks	Nombre
AWS/MWAA	QueuedTasks	Nombre

Métriques Application Load Balancer

Les métriques Application Load Balancer s'appliquent aux serveurs Web exécutés dans votre environnement. Amazon MWAA utilise ces indicateurs pour dimensionner vos serveurs Web en fonction du volume de trafic. Pour plus d'informations sur les unités et les descriptions des mesures d'équilibrage de charge suivantes, reportez-vous aux [CloudWatch métriques de votre Application Load Balancer dans le Guide](#) de l'utilisateur des Application Load Balancers.

Namespace	Métrique	Unit
AWS/MWAA	ActiveConnectionCount	Nombre

Sécurité dans les flux de travail gérés par Amazon pour Apache Airflow

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous (le client). Le [modèle de responsabilité partagée](#) décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de [AWS conformité Programmes](#) de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon MWAA, consultez la section [AWS Services concernés par le programme de conformitéAWS](#) .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon Managed Workflows pour Apache Airflow. Utilisez-le pour configurer Amazon MWAA afin d'atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources Amazon MWAA.

Dans cette section :

- [Protection des données dans les flux de travail gérés par Amazon pour Apache Airflow](#)
- [Gestion des identités et des accès AWS](#)
- [Validation de conformité pour les flux de travail gérés par Amazon pour Apache Airflow](#)
- [Résilience dans les flux de travail gérés par Amazon pour Apache Airflow](#)
- [Sécurité de l'infrastructure dans Amazon MWAA](#)
- [Analyse de configuration et de vulnérabilité dans Amazon MWAA](#)
- [Bonnes pratiques de sécurité sur Amazon MWAA](#)

Protection des données dans les flux de travail gérés par Amazon pour Apache Airflow

Le modèle de [responsabilité AWS partagée Le modèle](#) s'applique à la protection des données dans Amazon Managed Workflows pour Apache Airflow. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. Vous êtes responsable du contrôle de votre contenu hébergé sur cette infrastructure. Ce contenu inclut la configuration de la sécurité et les tâches de gestion Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la [FAQ sur la confidentialité des données](#). Pour plus d'informations sur la protection des données en Europe, consultez le [modèle de responsabilitéAWS partagée et le billet de blog sur le RGPD](#) sur le blog sur la AWS sécurité.

Pour des raisons de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des comptes utilisateur individuels avec Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous vous recommandons le certificat TLS 1.2 ou une version ultérieure.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut au sein AWS des services.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données personnelles stockées dans Amazon S3.

Nous vous recommandons vivement de ne jamais placer d'informations confidentielles ou sensibles, telles que des adresses électroniques de vos clients, dans des balises ou des champs de forme libre tels qu'un champ Nom. Cela inclut lorsque vous travaillez avec Amazon MWAA ou d'autres AWS services à l'aide de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous saisissez dans les balises ou les champs de forme libre utilisés pour les noms peuvent être utilisées pour les journaux de facturation ou de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement sur Amazon MWAA

Les rubriques suivantes décrivent comment Amazon MWAA protège vos données au repos et en transit. Utilisez ces informations pour découvrir comment Amazon MWAA s'intègre AWS KMS pour chiffrer les données au repos, et comment les données sont chiffrées à l'aide du protocole TLS (Transport Layer Security) en transit.

Rubriques

- [Chiffrement au repos](#)
- [Chiffrement en transit](#)

Chiffrement au repos

Sur Amazon MWAA, les données au repos sont des données que le service enregistre sur un support persistant.

Vous pouvez utiliser une [clé AWS détenue](#) pour le chiffrement des données au repos, ou éventuellement fournir une [clé gérée par le client](#) pour un chiffrement supplémentaire lorsque vous créez un environnement. Si vous choisissez d'utiliser une clé KMS gérée par le client, elle doit se trouver dans le même compte que les autres AWS ressources et services que vous utilisez dans votre environnement.

Pour utiliser une clé KMS gérée par le client, vous devez joindre la déclaration de politique requise pour CloudWatch accéder à votre politique de clé. Lorsque vous utilisez une clé KMS gérée par le client pour votre environnement, Amazon MWAA [octroie quatre subventions](#) en votre nom. Pour plus d'informations sur les subventions qu'Amazon MWAA attache à une clé KMS gérée par le client, reportez-vous à la section Clés gérées par le [client](#) pour le chiffrement des données.

Si vous ne spécifiez pas de clé KMS gérée par le client, Amazon MWAA utilise par défaut une clé KMS AWS détenue pour chiffrer et déchiffrer vos données. Nous vous recommandons d'utiliser une clé KMS AWS détenue pour gérer le chiffrement des données sur Amazon MWAA.

Note

Vous payez pour le stockage et l'utilisation de clés KMS AWS détenues ou gérées par le client sur Amazon MWAA. Pour plus d'informations, reportez-vous à la section [AWS KMS Tarification](#).

Artefacts de chiffrement

Vous spécifiez les artefacts de chiffrement utilisés pour le chiffrement au repos en spécifiant une [clé AWS détenue ou une clé gérée par le client](#) lorsque vous créez votre environnement Amazon MWAA. Amazon MWAA ajoute les [subventions](#) nécessaires à la clé que vous avez spécifiée.

Amazon S3 — Les données Amazon S3 sont chiffrées au niveau de l'objet à l'aide du chiffrement côté serveur (SSE). Le chiffrement et le déchiffrement Amazon S3 ont lieu dans le compartiment Amazon S3 où sont stockés votre code DAG et les fichiers de support. Les objets sont chiffrés lorsqu'ils sont chargés sur Amazon S3 et déchiffrés lorsqu'ils sont téléchargés sur votre environnement Amazon MWAA. Par défaut, si vous utilisez une clé KMS gérée par le client, Amazon MWAA l'utilise pour lire et déchiffrer les données de votre compartiment Amazon S3.

CloudWatch Journaux — Si vous utilisez une clé KMS AWS détenue, les journaux Apache Airflow envoyés à CloudWatch Logs sont chiffrés à l'aide de SSE avec la clé KMS AWS détenue par CloudWatch Logs. Si vous utilisez une clé KMS gérée par le client, vous devez ajouter une [politique de clé](#) à votre clé KMS pour autoriser CloudWatch Logs à utiliser votre clé.

Amazon SQS — Amazon MWAA crée une file d'attente Amazon SQS pour votre environnement. Amazon MWAA gère le chiffrement des données transmises vers et depuis la file d'attente à l'aide de SSE avec une clé KMS AWS détenue ou une clé KMS gérée par le client que vous spécifiez. Vous devez ajouter des autorisations Amazon SQS à votre rôle d'exécution, que vous utilisiez une clé KMS AWS détenue ou gérée par le client.

Aurora PostgreSQL — Amazon MWAA crée un cluster PostgreSQL pour votre environnement. Aurora PostgreSQL chiffre le contenu à l'aide d'AWS une clé KMS détenue ou gérée par le client à l'aide de SSE. Si vous utilisez une clé KMS gérée par le client, Amazon RDS ajoute au moins deux autorisations à la clé : une pour le cluster et une pour l'instance de base de données. Amazon RDS peut créer des subventions supplémentaires si vous choisissez d'utiliser votre clé KMS gérée par le client sur plusieurs environnements. Pour plus d'informations, consultez la section [Protection des données dans Amazon RDS](#).

Chiffrement en transit

Les données en transit sont considérées comme des données qui peuvent être interceptées lorsqu'elles circulent sur le réseau.

Le protocole TLS (Transport Layer Security) chiffre les objets Amazon MWAA en transit entre les composants Apache Airflow de votre environnement et d'autres services intégrés à Amazon MWAA,

tels AWS qu'Amazon S3. Pour plus d'informations sur le chiffrement Amazon S3, consultez la section [Protection des données à l'aide du chiffrement](#).

Utilisation de clés gérées par le client pour le chiffrement

Vous pouvez éventuellement fournir une [clé gérée par le client](#) pour le chiffrement des données dans votre environnement. Vous devez créer la clé KMS gérée par le client dans la même région que votre instance d'environnement Amazon MWAA et votre compartiment Amazon S3 dans lequel vous stockez les ressources pour vos flux de travail. Si la clé KMS gérée par le client que vous spécifiez se trouve dans un compte différent de celui que vous utilisez pour configurer un environnement, vous devez spécifier la clé à l'aide de son ARN pour l'accès entre comptes. Pour plus d'informations sur la création de clés, reportez-vous à la section [Création de clés](#) du manuel du AWS Key Management Service développeur.

Ce qui est pris en charge

AWS KMS fonctionnalité	Pris en charge	
Un identifiant de AWS KMS clé ou un ARN .	Oui	
Un alias AWS KMS clé .	Non	
Une clé AWS KMS multirégionale .	Non	

Utilisation de subventions pour le chiffrement

Cette rubrique décrit les autorisations qu'Amazon MWAA accorde à une clé KMS gérée par le client en votre nom pour chiffrer et déchiffrer vos données.

Comment ça marche

[Deux mécanismes de contrôle d'accès basés sur les ressources sont pris en charge par AWS KMS les clés KMS gérées par le client : une politique clé et une autorisation.](#)

Une politique clé est utilisée lorsque l'autorisation est principalement statique et utilisée en mode de service synchrone. Une autorisation est utilisée lorsque des autorisations plus dynamiques et plus détaillées sont requises, par exemple lorsqu'un service doit définir différentes autorisations d'accès pour lui-même ou pour d'autres comptes.

Amazon MWAA utilise et associe quatre politiques de subvention à votre clé KMS gérée par le client. Cela est dû aux autorisations granulaires requises pour qu'un environnement crypte les données au repos provenant des CloudWatch journaux, de la file d'attente Amazon SQS, de la base de données Aurora PostgreSQL, des secrets Secrets Manager, du compartiment Amazon S3 et des tables DynamoDB.

Lorsque vous créez un environnement Amazon MWAA et que vous spécifiez une clé KMS gérée par le client, Amazon MWAA associe les politiques d'octroi à votre clé KMS gérée par le client. Ces politiques permettent à Amazon MWAA `airflow.us-east-1.amazonaws.com` d'utiliser votre clé KMS gérée par le client pour chiffrer en votre nom les ressources détenues par Amazon MWAA.

Amazon MWAA crée et attache des autorisations supplémentaires à une clé KMS spécifiée en votre nom. Cela inclut des politiques visant à retirer une subvention si vous supprimez votre environnement, à utiliser votre clé KMS gérée par le client pour le chiffrement côté client (CSE) et pour le rôle d' AWS Fargate exécution qui doit accéder aux secrets protégés par votre clé gérée par le client dans Secrets Manager.

Politiques relatives aux subventions

Amazon MWAA ajoute les subventions [politiques basées sur les ressources](#) suivantes en votre nom à une clé KMS gérée par le client. Ces politiques permettent au bénéficiaire et au principal (Amazon MWAA) d'effectuer les actions définies dans la politique.

Subvention 1 : utilisée pour créer des ressources de plan de données

```
{
  "Name": "mwaagrantforenvmgmtrole-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey",
```

```

    "kms:RetireGrant"
  ]
}

```

Subvention 2 : utilisée pour l'**ControllerLambdaExecutionRole**accès

```

{
  "Name": "mwaa-grant-for-lambda-exec-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}

```

Subvention 3 : utilisée pour l'**CfnManagementLambdaExecutionRole**accès

```

{
  "Name": " mwaa-grant-for-cfn-mgmt-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ]
}

```

Grant 4 : utilisé pour le rôle d'exécution de Fargate pour accéder aux secrets du backend

```

{
  "Name": "mwaa-fargate-access-for-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [

```

```
"kms:Encrypt",
"kms:Decrypt",
"kms:ReEncrypt*",
"kms:GenerateDataKey*",
"kms:DescribeKey",
"kms:RetireGrant"
]
}
```

Associer des politiques clés à une clé gérée par le client

Si vous choisissez d'utiliser votre propre clé KMS gérée par le client avec Amazon MWAA, vous devez associer la politique suivante à la clé afin de permettre à Amazon MWAA de l'utiliser pour chiffrer vos données.

Si la clé KMS gérée par le client que vous avez utilisée pour votre environnement Amazon MWAA n'est pas déjà configurée pour fonctionner CloudWatch, vous devez mettre à jour la [politique relative aux clés](#) pour autoriser les journaux chiffrés. CloudWatch Pour plus d'informations, reportez-vous au service [Chiffrer les données du journal lors de l' CloudWatch utilisation du AWS Key Management Service service](#).

L'exemple suivant représente une politique clé pour CloudWatch Logs. Remplacez les valeurs d'échantillon fournies pour la région.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:*:*"
    }
  }
}
```

```
}
```

Gestion des identités et des accès AWS

Gestion des identités et des accès AWS (IAM) est un AWS service qui aide un administrateur à contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources Amazon Managed Workflows for Apache Airflow. IAM est un AWS service que vous pouvez utiliser sans frais supplémentaires.

Cette rubrique fournit un aperçu de base de la façon dont Amazon MWAA utilise Gestion des identités et des accès AWS (IAM). Pour en savoir plus sur la gestion de l'accès à Amazon MWAA, reportez-vous à [Gestion de l'accès à un environnement Amazon MWAA](#).

Table des matières

- [Public ciblé](#)
- [Authentification avec des identités](#)
- [Gestion des accès à l'aide de politiques](#)
- [Permettre aux utilisateurs d'accéder à leurs propres autorisations](#)
- [Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow](#)
- [Comment Amazon MWAA fonctionne avec IAM](#)

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir [Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow](#))
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir [Comment Amazon MWAA fonctionne avec IAM](#))
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir [Exemples de politiques basées sur l'identité Amazon MWAA](#))

Authentification avec des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez [Connexion à votre Compte AWS](#) dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez [Signature AWS Version 4 pour les demandes d'API](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez [Tâches qui requièrent les informations d'identification de l'utilisateur racine](#) dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir [Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires](#) dans le guide de l'utilisateur IAM.

[Les groupes IAM](#) spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez [Cas d'utilisation pour les utilisateurs IAM](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en [passant d'un rôle utilisateur à un](#)

[rôle IAM \(console\)](#) ou en appelant une opération AWS CLI ou AWS API. Pour plus d'informations, consultez [Méthodes pour endosser un rôle](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

politiques basées sur l'identité

Les stratégies basées sur l'identité sont des documents de stratégie d'autorisations JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

Politiques basées sur une ressource

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la [présentation de la liste de contrôle d'accès \(ACL\)](#) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) — Spécifiez les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez [Politiques de contrôle de service](#) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir [Politiques de contrôle des ressources \(RCPs\)](#) dans le guide de AWS Organizations l'utilisateur.
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez [Politiques de session](#) dans le Guide de l'utilisateur IAM.

Types de politique multiple

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section [Logique d'évaluation des politiques](#) dans le guide de l'utilisateur IAM.

Permettre aux utilisateurs d'accéder à leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

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

Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon MWAA et IAM.

Je ne suis pas autorisé à effectuer une action dans Amazon MWAA

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole` action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon MWAA.

Certains services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon MWAA. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources Amazon MWAA

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon MWAA prend en charge ces fonctionnalités, consultez [Comment Amazon MWAA fonctionne avec IAM](#).
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section [Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section [Fournir un accès à des ressources Comptes AWS détenues par des tiers](#) dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Comment Amazon MWAA fonctionne avec IAM

Amazon MWAA utilise des politiques basées sur l'identité IAM pour accorder des autorisations aux actions et aux ressources Amazon MWAA. Pour obtenir des exemples recommandés de politiques IAM personnalisées que vous pouvez utiliser pour contrôler l'accès à vos ressources Amazon MWAA, reportez-vous à [the section called "Accès à un environnement Amazon MWAA"](#)

Pour obtenir un accès détaillé à la manière dont Amazon MWAA et d'autres AWS services fonctionnent avec IAM, reportez-vous à la section [AWS Services That Work with IAM dans le guide de l'utilisateur IAM](#).

Politiques basées sur l'identité Amazon MWAA

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Amazon MWAA prend en charge des actions, des ressources et des clés de condition spécifiques.

Les étapes suivantes expliquent comment créer une nouvelle politique JSON à l'aide de la console IAM. Cette politique fournit un accès en lecture seule à vos ressources Amazon MWAA.

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Politiques (Politiques).

Si vous sélectionnez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.

3. En haut de la page, sélectionnez Créer une politique.
4. Dans la section Éditeur de politique, choisissez l'option JSON.
5. Entrez le document de politique JSON suivant :

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

6. Choisissez Suivant.

 Note

Vous pouvez basculer à tout moment entre les options des éditeurs visuel et JSON. Toutefois, si vous apportez des modifications ou si vous choisissez Suivant dans l'éditeur visuel, IAM peut restructurer votre politique afin de l'optimiser pour l'éditeur visuel. Pour plus d'informations, consultez la page [Restructuration de politique](#) dans le Guide de l'utilisateur IAM.

7. Sur la page Vérifier et créer, saisissez un Nom de politique et une Description (facultative) pour la politique que vous créez. Vérifiez les Autorisations définies dans cette politique pour voir les autorisations accordées par votre politique.
8. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.

Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, reportez-vous à la [référence des éléments de stratégie JSON IAM](#) dans le guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément Action d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les déclarations de politique doivent inclure un élément Action ou NotAction. L'élément Action répertorie les actions autorisées par la politique. L'élément NotAction répertorie les actions qui ne sont pas autorisées.

Les actions définies pour Amazon MWAA reflètent les tâches que vous pouvez effectuer à l'aide d'Amazon MWAA. Les actions stratégiques dans Detective ont le préfixe suivant :`airflow:`.

Vous pouvez utiliser des caractères génériques (*) pour spécifier plusieurs actions. Au lieu de répertorier ces actions séparément, vous pouvez autoriser l'accès à toutes les actions se terminant par le mot, par exemple `environment`.

Pour obtenir la liste des actions Amazon MWAA, reportez-vous à la section [Actions définies par Amazon Managed Workflows pour Apache](#) Airflow dans le guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité Amazon MWAA

Pour accéder aux politiques d'Amazon MWAA, reportez-vous à [Gestion de l'accès à un environnement Amazon MWAA](#).

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou à modifier les ressources Amazon MWAA. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API AWS Management Console AWS CLI, ou.

Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. L'administrateur associe ensuite ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Important

Nous vous recommandons d'utiliser des rôles IAM et des informations d'identification temporaires pour accéder à vos ressources Amazon MWAA. Évitez d'associer des politiques d'autorisation directement à vos utilisateurs IAM.

Pour savoir comment créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, reportez-vous à la section [Création de politiques dans l'onglet JSON du guide](#) de l'utilisateur IAM.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console Amazon MWAA](#)
- [Permettre aux utilisateurs d'accéder à leurs propres autorisations](#)

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Amazon MWAA dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez

les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [politiques gérées par AWS](#) ou [politiques gérées par AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.

- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politiques avec IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Sécurisation de l'accès aux API avec MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon MWAA

Pour utiliser la console Amazon MWAA, l'utilisateur ou le rôle doit avoir accès aux actions pertinentes, qui correspondent aux actions correspondantes dans l'API.

Pour accéder aux politiques d'Amazon MWAA, reportez-vous à [Gestion de l'accès à un environnement Amazon MWAA](#).

Permettre aux utilisateurs d'accéder à leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

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

Validation de conformité pour les flux de travail gérés par Amazon pour Apache Airflow

Pour savoir si un [programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de [AWS conformité Programmes AWS](#) de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#) .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez [AWS la documentation de sécurité](#).

Résilience dans les flux de travail gérés par Amazon pour Apache Airflow

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à débit élevé et à forte redondance. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, reportez-vous à la section [Infrastructure AWS globale](#).

Sécurité de l'infrastructure dans Amazon MWAA

En tant que service géré, Amazon Managed Workflows pour Apache Airflow est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon MWAA via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Analyse de configuration et de vulnérabilité dans Amazon MWAA

La configuration et les contrôles informatiques sont une responsabilité partagée entre vous AWS et vous, notre client.

Amazon Managed Workflows for Apache Airflow corrige et met régulièrement à niveau Apache Airflow sur vos environnements. Assurez-vous que les politiques d'accès appropriées sont utilisées pour votre VPCs.

Pour plus de détails, consultez les ressources suivantes :

- [Validation de conformité pour les flux de travail gérés par Amazon pour Apache Airflow](#)
- [Modèle de responsabilité partagée](#)
- [Amazon Web Services : Présentation des procédures de sécurité](#)
- [Sécurité de l'infrastructure dans Amazon MWAA](#)
- [Bonnes pratiques de sécurité sur Amazon MWAA](#)

Bonnes pratiques de sécurité sur Amazon MWAA

Amazon MWAA fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lorsque vous développez et mettez en œuvre vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

- Utilisez les politiques d'autorisation les moins permissives. Accordez des autorisations uniquement aux ressources ou aux actions dont les utilisateurs ont besoin pour effectuer des tâches.
- AWS CloudTrail À utiliser pour surveiller l'activité des utilisateurs sur votre compte.
- Assurez-vous que la politique du compartiment et l'objet Amazon S3 ACLs autorisent les utilisateurs de l'environnement Amazon MWAA associé à placer des objets dans le compartiment. Cela garantit que les utilisateurs autorisés à ajouter des flux de travail au compartiment sont également autorisés à exécuter les flux de travail dans Airflow.
- Utilisez les compartiments Amazon S3 associés aux environnements Amazon MWAA. Votre compartiment Amazon S3 peut porter n'importe quel nom. Ne stockez pas d'autres objets dans le compartiment et n'utilisez pas le compartiment avec un autre service.

Bonnes pratiques de sécurité dans Apache Airflow

Apache Airflow n'est pas un système multi-tenant. Bien qu'il existe [des mesures de contrôle d'accès](#) pour limiter certaines fonctionnalités à des utilisateurs spécifiques, mises [en œuvre par Amazon MWAA](#), les créateurs de DAG ont la possibilité d'écrire DAGs des textes susceptibles de modifier les privilèges des utilisateurs d'Apache Airflow et d'interagir avec la base de métadonnées sous-jacente.

Nous vous recommandons de suivre les étapes suivantes lorsque vous utilisez Apache Airflow sur Amazon MWAA afin de garantir la sécurité de la base de métadonnées de votre environnement.

DAGs

- Utilisez des environnements distincts pour des équipes distinctes disposant d'un accès d'écriture DAG ou de la possibilité d'ajouter des fichiers à votre /dags dossier Amazon S3, en supposant que tout ce qui est accessible par le biais du [rôle d'exécution Amazon MWAA](#) ou des connexions [Apache Airflow](#) sera également accessible aux utilisateurs autorisés à écrire dans l'environnement.

- Ne fournissez pas d'accès direct au DAGs dossier Amazon S3. Utilisez plutôt CI/CD des outils DAGs pour écrire sur Amazon S3, avec une étape de validation garantissant que le code DAG est conforme aux directives de sécurité de votre équipe.
- Empêchez les utilisateurs d'accéder au compartiment Amazon S3 de votre environnement. Utilisez plutôt une usine DAG qui génère en DAGs fonction d'un fichier de définition YAML, JSON ou autre stocké dans un emplacement distinct de votre compartiment Amazon MWA Amazon S3 dans lequel vous stockez. DAGs
- Stockez les [secrets dans Secrets Manager](#). Cela n'empêchera pas les utilisateurs capables DAGs d'écrire de lire les secrets, mais cela les empêchera de modifier les secrets utilisés par votre environnement.

Détecter les modifications apportées aux privilèges des utilisateurs d'Apache Airflow

Vous pouvez utiliser CloudWatch Logs Insights pour détecter les cas de DAGs modification des privilèges des utilisateurs d'Apache Airflow. Pour ce faire, vous pouvez utiliser une règle EventBridge planifiée, une fonction Lambda et CloudWatch Logs Insights pour envoyer des notifications aux CloudWatch métriques chaque fois que l'un de vos utilisateurs DAGs modifie les privilèges d'utilisateur d'Apache Airflow.

Conditions préalables

Pour effectuer les étapes suivantes, vous aurez besoin des éléments suivants :

- Un environnement Amazon MWA dans lequel tous les types de journaux Apache Airflow sont activés au niveau INFO du journal. Pour plus d'informations, consultez [the section called "Accès aux journaux Airflow"](#).

Pour configurer les notifications relatives aux modifications apportées aux privilèges des utilisateurs d'Apache Airflow

1. [Créez une fonction Lambda](#) qui exécute la chaîne de requête CloudWatch Logs Insights suivante sur les cinq groupes de journaux de l'environnement Amazon MWA (DAGProcessing,, Scheduler TaskWebServer, et). Worker

```
fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
by @log
```

2. [Créez une EventBridge règle qui s'exécute selon un calendrier](#), avec la fonction Lambda que vous avez créée à l'étape précédente comme cible de la règle. Configurez votre planning à l'aide d'une expression cron ou rate à exécuter à intervalles réguliers.

Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow

Cette rubrique décrit les versions d'Apache Airflow prises en charge par Amazon Managed Workflows pour Apache Airflow, ainsi que les meilleures pratiques pour la mise à niveau vers la dernière version.

Rubriques

- [À propos des versions Amazon MWAA](#)
- [Dernière version](#)
- [Versions d'Apache Airflow](#)
- [Composants d'Apache Airflow](#)
- [Mise à niveau de la version d'Apache Airflow](#)
- [Rétrogradation de la version d'Apache Airflow](#)
- [Versions obsolètes d'Apache Airflow](#)
- [Support des versions d'Apache Airflow et FAQ](#)

À propos des versions Amazon MWAA

Amazon MWAA crée des images de conteneur qui regroupent les versions d'Apache Airflow avec d'autres binaires et bibliothèques Python courants. L'image utilise l'installation de base d'Apache Airflow pour la version que vous spécifiez. Lorsque vous créez un environnement, vous spécifiez la version de l'image à utiliser. Une fois qu'un environnement est créé, il continue d'utiliser la version d'image spécifiée jusqu'à ce que vous le mettiez à niveau vers une version ultérieure.

Dernière version

Amazon MWAA prend en charge plusieurs versions d'Apache Airflow. Si vous ne spécifiez pas de version d'image lorsque vous créez un environnement, Amazon MWAA crée un environnement en utilisant la dernière version prise en charge d'Apache Airflow.

Versions d'Apache Airflow

Les versions d'Apache Airflow suivantes sont prises en charge sur Amazon Managed Workflows pour Apache Airflow.

Note

- À compter du 30 décembre 2025, Amazon MWAA cessera de prendre en charge les versions 2.4.3, 2.5.1 et 2.6.3 d'Apache Airflow. Pour plus d'informations, consultez [Support des versions d'Apache Airflow et FAQ](#).
- À partir de la version 2.2.2 d'Apache Airflow, Amazon MWAA prend en charge l'installation des exigences relatives à Python, des packages de fournisseurs et des plug-ins personnalisés directement sur le serveur Web Apache Airflow.
- À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. --constraint Si vous ne fournissez aucune contrainte, Amazon MWAA vous en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Pour plus d'informations sur la configuration des contraintes dans votre fichier d'exigences, reportez-vous à la section [Installation des dépendances Python](#).

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.11.0	20 mai 2025	7 janvier 2026	fichier de contraintes v2.11.0	Python 3.12
v3.0.6	29 août 2025	1er octobre 2025	fichier de contraintes v3.0.6	Python 3.12
v2.10.3	4 novembre 2024	18 décembre 2024	fichier de contraintes v2.10.3	Python 3.11

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.10.1	5 septembre 2024	26 septembre 2024	fichier de contraintes v2.10.1	Python 3.11
v2.9.2	10 juin 2024	9 juillet 2024	fichier de contraintes v2.9.2	Python 3.11
v2.8.1	19 janvier 2024	23 février 2024	fichier de contraintes v2.8.1	Python 3.11
v2.7.2	12 octobre 2023	6 novembre 2023	fichier de contraintes v2.7.2	Python 3.11

[Pour plus d'informations sur la migration de vos déploiements Apache Airflow autogérés ou sur la migration d'un environnement Amazon MWAA existant, y compris les instructions pour sauvegarder votre base de données de métadonnées, consultez le guide de migration Amazon MWAA.](#)

Composants d'Apache Airflow

Cette section décrit le nombre de planificateurs et de travailleurs Apache Airflow disponibles pour chaque version d'Apache Airflow sur Amazon MWAA, et fournit une liste des principales fonctionnalités d'Apache Airflow, en indiquant la version qui prend en charge chaque fonctionnalité.

Schedulers

Planificateurs pour Apache Airflow v2 et versions ultérieures :

Planificateur (par défaut)	Planificateur (min)	Planificateur (max)

Planificateur (par défaut)	Planificateur (min)	Planificateur (max)
2	2	5

Workers

Fonctionnaires pour Apache Airflow v2 et versions ultérieures :

Travailleurs (par défaut)	Travailleurs (min)	Travailleurs (max.)
10	1	25

Mise à niveau de la version d'Apache Airflow

Amazon MWAA prend en charge les mises à niveau de versions mineures. Cela signifie que vous pouvez mettre à niveau votre environnement d'une version `x.1.z` à `x.2.z`, mais pas vers une nouvelle version majeure, par exemple de `1.y.z` à `2.y.z`.

Pour plus d'informations et des instructions détaillées sur la mise à jour des ressources de votre flux de travail et la mise à niveau de l'environnement vers une nouvelle version, reportez-vous à [the section called “Modification de la version”](#).

Rétrogradation de la version d'Apache Airflow

Amazon MWAA prend en charge les rétrogradations mineures vers une version antérieure qui est toujours prise en charge au moment de la rétrogradation. Cela signifie que vous pouvez rétrograder votre environnement d'une version `x.2.z` à `x.1.z`, mais pas à une version majeure précédente, par exemple de `2.y.z` à `1.y.z`.

Pour plus d'informations et des instructions détaillées sur la mise à jour des ressources de votre flux de travail et la mise à niveau de l'environnement vers une nouvelle version, reportez-vous à [the section called “Modification de la version”](#).

Versions obsolètes d'Apache Airflow

Le tableau suivant répertorie les versions obsolètes d'Apache Airflow dans Amazon MWAA, ainsi que la date de end-of-support publication initiale et les dates de chaque version. Pour plus d'informations sur la migration vers une version plus récente, consultez le guide de [migration Amazon MWAA](#).

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Date Amazon MWAA end-of-support
v1.10.12	25 août 2020	24 novembre 2020	21 février 2024
v2.0.2	19 avril 2021	25 mai 2021	29 avril 2024
v2.2.2	15 novembre 2021	27 janvier 2022	27 juin 2024
v2.4.3	14 novembre 2022	05 janvier 2023	30 décembre 2025
v2.5.1	20 janvier 2023	11 avril 2023	30 décembre 2025
v2.6.3	10 juillet 2023	9 août 2023	30 décembre 2025

Support des versions d'Apache Airflow et FAQ

Conformément au [processus de publication et à la politique de version](#) de la communauté Apache Airflow, Amazon MWAA s'engage à prendre en charge au moins trois versions mineures d'Apache Airflow à tout moment. Nous annoncerons la date de fin de support d'une version mineure d'Apache Airflow donnée au moins 180 jours avant la date de fin du support.

Questions fréquentes (FAQ)

Q : Pendant combien de temps Amazon MWAA prend-il en charge une version d'Apache Airflow ?

R : Amazon MWAA prend en charge une version de correctif d'Apache Airflow pendant au moins 12 mois après sa première mise à disposition.

Q : Suis-je averti de la fin du support pour une version d'Apache Airflow sur Amazon MWAA ?

A : Oui. Si l'un des environnements Amazon MWAA de votre compte exécute la version approchant de la fin du support, Amazon MWAA envoie un avis indiquant la date Tableau de bord Health de fin du support.

Q : Que se passe-t-il à la date de fin de la prise en charge ?

R : À la date de fin du support, vous ne pourrez plus utiliser une version obsolète pour créer de nouveaux environnements Amazon MWAA. Vous pouvez continuer à accéder à vos environnements Amazon MWAA existants qui exécutent la version obsolète associée d'Apache Airflow à vos risques et périls. Pour effectuer une mise à niveau vers une version plus récente d'Apache Airflow sur Amazon MWAA, consultez le guide de migration [Amazon MWAA](#).

Important

Vous êtes responsable de la mise à jour de vos versions Amazon MWAA. AWS invite tous les clients à mettre à niveau leurs environnements Amazon MWAA vers la dernière version afin de bénéficier des garanties de sécurité, de confidentialité et de disponibilité les plus récentes. Si vous exploitez votre environnement sur une version ou un logiciel non pris en charge après la date d'obsolescence, appelée ancienne version, vous êtes exposé à un risque accru en matière de sécurité, de confidentialité et d'exploitation, y compris des interruptions de service. En exploitant votre environnement Amazon MWAA sur une ancienne version, vous confirmez que vous comprenez et assumez sciemment ces risques, et vous acceptez de terminer votre mise à niveau vers la dernière version dès que possible. Le fonctionnement continu de votre environnement sur une ancienne version est soumis à l'accord régissant votre utilisation des AWS services.

Les anciennes versions ne sont pas considérées comme étant généralement disponibles et AWS ne fournissent plus de support pour les anciennes versions. En conséquence, AWS peut imposer des limites à l'accès ou à l'utilisation de toute version héritée à tout moment, s'il est AWS déterminé que l'ancienne version présente un risque de sécurité ou de responsabilité, ou un risque de préjudice AWS, pour les services, ses filiales ou tout autre tiers. Votre décision de continuer à exécuter vos charges de travail sur une ancienne version peut entraîner l'indisponibilité, la corruption ou l'impossibilité de récupérer votre contenu. Les environnements exécutés sur une ancienne version sont soumis à des exceptions au contrat de niveau de service (SLA).

Les environnements et les logiciels associés exécutés sur une ancienne version peuvent contenir des bogues, des erreurs, des défauts et des composants dangereux. En conséquence, et nonobstant toute information contraire contenue dans le contrat ou dans les conditions d'utilisation, AWS l'ancienne version est fournie telle quelle.

Pour plus d'informations sur AWS le modèle de responsabilité partagée, reportez-vous à la section [Responsabilité partagée](#) dans le cadre AWS Well-Architected.

Amazon Managed Workflows pour les points de terminaison et quotas du service Apache Airflow

Amazon Managed Workflows pour Apache Airflow possède les quotas de service et les points de terminaison suivants. Les quotas de service, également appelés limites, représentent le nombre maximal de ressources ou d'opérations de service pour votre Compte AWS.

Table des matières

- [Points de terminaison de service](#)
- [Quotas de service](#)
- [Augmenter les quotas](#)

Points de terminaison de service

Pour accéder à la liste des points de terminaison pour Amazon MWAA, reportez-vous à [Amazon Managed Workflows for Apache Airflow endpoints and quotas](#).

Quotas de service

Nom du quota	Description	Quota par défaut	Ajustable
Environnements	Le nombre maximum d'environnements Amazon MWAA par compte et par région.	10	Oui
Travailleurs par environnement	Le nombre maximum de travailleurs par environnement Amazon MWAA.	25	Oui
Serveurs Web par environnement	Le nombre maximum de serveurs Web par environnement Amazon MWAA.	5	Oui

Augmenter les quotas

Vous pouvez demander l'augmentation d'un quota ajustable en soumettant une [demande d'augmentation de quota](#).

Questions fréquemment posées sur Amazon MWAA

Cette page décrit les questions courantes que vous pouvez rencontrer lors de l'utilisation d'Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [Versions prises en charge](#)
 - [Qu'est-ce qu'Amazon MWAA prend en charge pour Apache Airflow v2 ?](#)
 - [Quelle version de Python puis-je utiliser ?](#)
- [Cas d'utilisation](#)
 - [Puis-je utiliser Amazon MWAA avec Amazon SageMaker Unified Studio ?](#)
 - [Quand puis-je utiliser AWS Step Functions vs. Amazon MWAA ?](#)
- [Spécifications relatives à l'environnement](#)
 - [Quelle est la capacité de stockage des tâches disponible pour chaque environnement ?](#)
 - [Quel est le système d'exploitation par défaut utilisé pour les environnements Amazon MWAA ?](#)
 - [Puis-je utiliser une image personnalisée pour mon environnement Amazon MWAA ?](#)
 - [Amazon MWAA est-il conforme à la loi HIPAA ?](#)
 - [Amazon MWAA prend-il en charge les instances ponctuelles ?](#)
 - [Amazon MWAA prend-il en charge un domaine personnalisé ?](#)
 - [Puis-je accéder à mon environnement par SSH ?](#)
 - [Pourquoi une règle d'autoréférencement est-elle requise sur le groupe de sécurité VPC ?](#)
 - [Puis-je masquer des environnements appartenant à différents groupes dans IAM ?](#)
 - [Puis-je stocker des données temporaires sur le worker Apache Airflow ?](#)
 - [Puis-je spécifier plus de 25 travailleurs Apache Airflow ?](#)
 - [Amazon MWAA prend-il en charge le partage d'Amazon VPCs ou les sous-réseaux partagés ?](#)
 - [Puis-je créer ou intégrer des files d'attente Amazon SQS personnalisées pour gérer l'exécution des tâches et l'orchestration des flux de travail dans Apache Airflow ?](#)
- [Métriques](#)
 - [Quels indicateurs sont utilisés pour déterminer s'il convient de faire évoluer les effectifs ?](#)
 - [Puis-je créer des métriques personnalisées dans CloudWatch ?](#)
- [DAGs, Opérateurs, connexions et autres questions](#)

- [Puis-je utiliser le PythonVirtualenvOperator ?](#)
- [Combien de temps faut-il à Amazon MWAA pour reconnaître un nouveau fichier DAG ?](#)
- [Pourquoi mon fichier DAG n'est-il pas récupéré par Apache Airflow ?](#)
- [Puis-je supprimer un environnement plugins.zip ou le supprimer requirements.txt d'un environnement ?](#)
- [Pourquoi mes plugins n'apparaissent-ils pas dans le menu des plugins d'administration d'Apache Airflow v2.0.2 ?](#)
- [Puis-je utiliser les opérateurs du Service AWS de Migration de Base de Données \(DMS\) ?](#)
- [Lorsque j'accède à l'API REST Airflow à l'aide des AWS informations d'identification, puis-je augmenter la limite de régulation à plus de 10 transactions par seconde \(TPS\) ?](#)
- [Où s'exécute le serveur d'API d'exécution des tâches Airflow dans Amazon MWAA ?](#)

Versions prises en charge

Qu'est-ce qu'Amazon MWAA prend en charge pour Apache Airflow v2 ?

Pour savoir ce que prend en charge Amazon MWAA, reportez-vous à [Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow](#).

Quelle version de Python puis-je utiliser ?

Les versions d'Apache Airflow suivantes sont prises en charge sur Amazon Managed Workflows pour Apache Airflow.

Note

- À compter du 30 décembre 2025, Amazon MWAA cessera de prendre en charge les versions 2.4.3, 2.5.1 et 2.6.3 d'Apache Airflow. Pour plus d'informations, consultez [Support des versions d'Apache Airflow et FAQ](#).
- À partir de la version 2.2.2 d'Apache Airflow, Amazon MWAA prend en charge l'installation des exigences relatives à Python, des packages de fournisseurs et des plug-ins personnalisés directement sur le serveur Web Apache Airflow.
- À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une déclaration. --constraint Si vous ne fournissez aucune contrainte, Amazon MWAA

vous en indiquera une afin de garantir que les packages répertoriés dans vos exigences sont compatibles avec la version d'Apache Airflow que vous utilisez.

Pour plus d'informations sur la configuration des contraintes dans votre fichier d'exigences, reportez-vous à la section [Installation des dépendances Python](#).

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.11.0	20 mai 2025	7 janvier 2026	fichier de contraintes v2.11.0	Python 3.12
v3.0.6	29 août 2025	1er octobre 2025	fichier de contraintes v3.0.6	Python 3.12
v2.10.3	4 novembre 2024	18 décembre 2024	fichier de contraintes v2.10.3	Python 3.11
v2.10.1	5 septembre 2024	26 septembre 2024	fichier de contraintes v2.10.1	Python 3.11
v2.9.2	10 juin 2024	9 juillet 2024	fichier de contraintes v2.9.2	Python 3.11
v2.8.1	19 janvier 2024	23 février 2024	fichier de contraintes v2.8.1	Python 3.11

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Contraintes d'Apache Airflow	Version de Python
v2.7.2	12 octobre 2023	6 novembre 2023	fichier de contraintes v2.7.2	Python 3.11

[Pour plus d'informations sur la migration de vos déploiements Apache Airflow autogérés ou sur la migration d'un environnement Amazon MWAA existant, y compris les instructions pour sauvegarder votre base de données de métadonnées, consultez le guide de migration Amazon MWAA.](#)

Cas d'utilisation

Puis-je utiliser Amazon MWAA avec Amazon SageMaker Unified Studio ?

Oui. Avec un flux de travail Amazon SageMaker Unified Studio, vous pouvez configurer et exécuter une série de tâches dans Amazon SageMaker Unified Studio. Les flux de travail Amazon SageMaker Unified Studio utilisent Apache Airflow pour modéliser les procédures de traitement des données et orchestrer vos artefacts de code Amazon SageMaker Unified Studio. Pour plus d'informations, reportez-vous à la section [Workflows](#). Pour en savoir plus sur Amazon SageMaker, reportez-vous à [Qu'est-ce qu'Amazon SageMaker ?](#)

Quand puis-je utiliser AWS Step Functions vs. Amazon MWAA ?

1. Vous pouvez utiliser Step Functions pour traiter les commandes individuelles de vos clients, car Step Functions peut évoluer pour répondre à la demande d'une commande ou d'un million de commandes.
2. Si vous gérez un flux de travail de nuit qui traite les commandes de la veille, vous pouvez utiliser Step Functions ou Amazon MWAA. Amazon MWAA vous propose une option open source pour extraire le flux de travail des AWS ressources que vous utilisez.

Spécifications relatives à l'environnement

Quelle est la capacité de stockage des tâches disponible pour chaque environnement ?

Le stockage des tâches est limité à 20 Go et est spécifié par [Amazon ECS Fargate 1.4](#). La quantité de RAM est déterminée par la classe d'environnement que vous spécifiez. Pour plus d'informations sur les classes d'environnement, reportez-vous à [Configuration de la classe d'environnement Amazon MWAA](#).

Quel est le système d'exploitation par défaut utilisé pour les environnements Amazon MWAA ?

Les environnements Amazon MWAA sont créés sur des instances exécutant Amazon Linux 2 pour les versions 2.6 et antérieures, et sur des instances exécutant Amazon Linux 2023 pour les versions 2.7 et ultérieures.

Puis-je utiliser une image personnalisée pour mon environnement Amazon MWAA ?

Les images personnalisées ne sont pas prises en charge. Amazon MWAA utilise des images créées sur l'AMI Amazon Linux. Amazon MWAA installe les exigences supplémentaires en `pip3 -r install` les exécutant conformément aux exigences spécifiées dans le fichier `requirements.txt` que vous ajoutez au compartiment Amazon S3 pour l'environnement.

Amazon MWAA est-il conforme à la loi HIPAA ?

Amazon MWAA est éligible à la loi [HIPAA \(Health Insurance Portability and Accountability Act\)](#). Si vous avez mis en place un addendum HIPAA Business Associate (BAA) AWS, vous pouvez utiliser Amazon MWAA pour les flux de travail gérant les informations de santé protégées (PHI) dans les environnements créés le 14 novembre 2022 ou après cette date.

Amazon MWAA prend-il en charge les instances ponctuelles ?

Amazon MWAA ne prend actuellement pas en charge les types d'instances Spot Amazon EC2 à la demande pour Apache Airflow. Toutefois, un environnement Amazon MWAA peut déclencher des instances Spot sur, par exemple, Amazon EMR et Amazon EC2.

Amazon MWAA prend-il en charge un domaine personnalisé ?

Pour pouvoir utiliser un domaine personnalisé pour votre nom d'hôte Amazon MWAA, effectuez l'une des opérations suivantes :

- Pour les déploiements Amazon MWAA avec accès à un serveur Web public, vous pouvez utiliser Amazon avec CloudFront Lambda @Edge pour diriger le trafic vers votre environnement et y associer un nom de domaine personnalisé. CloudFront Pour plus d'informations et un exemple de configuration d'un domaine personnalisé pour un environnement public, consultez l'exemple de [domaine personnalisé Amazon MWAA pour serveur Web public](#) dans le référentiel d'exemples GitHub Amazon MWAA.
- Pour les déploiements Amazon MWAA avec accès à un serveur Web privé, reportez-vous à [the section called "Configuration d'un domaine personnalisé"](#)

Puis-je accéder à mon environnement par SSH ?

Bien que le SSH ne soit pas pris en charge dans un environnement Amazon MWAA, il est possible d'utiliser un DAG pour exécuter des commandes bash à l'aide du BashOperator. Par exemple :

```
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'] }}"
        )
```

Pour déclencher le DAG dans l'interface utilisateur d'Apache Airflow, utilisez :

```
{ "command" : "your bash command" }
```

Pourquoi une règle d'autoréférencement est-elle requise sur le groupe de sécurité VPC ?

En créant une règle d'autoréférencement, vous limitez la source au même groupe de sécurité dans le VPC, et celle-ci n'est pas ouverte à tous les réseaux. Pour en savoir plus, reportez-vous à [the section called “Sécurité dans votre VPC”](#).

Puis-je masquer des environnements appartenant à différents groupes dans IAM ?

Vous pouvez limiter l'accès en spécifiant un nom d'environnement Gestion des identités et des accès AWS, mais le filtrage d'accès n'est pas disponible dans la AWS console. Si un utilisateur peut accéder à un environnement, il peut accéder à tous les environnements.

Puis-je stocker des données temporaires sur le worker Apache Airflow ?

Vos opérateurs Apache Airflow peuvent stocker des données temporaires sur les travailleurs. Les utilisateurs d'Apache Airflow peuvent accéder aux fichiers temporaires contenus dans /tmp les conteneurs Fargate de votre environnement.

Note

Le stockage total des tâches est limité à 20 Go, selon [Amazon ECS Fargate 1.4](#). Rien ne garantit que les tâches suivantes s'exécuteront sur la même instance de conteneur Fargate, qui peut utiliser un dossier différent. /tmp

Puis-je spécifier plus de 25 travailleurs Apache Airflow ?

Oui. Bien que vous puissiez spécifier jusqu'à 25 travailleurs Apache Airflow sur la console Amazon MWAA, vous pouvez en configurer jusqu'à 50 dans un environnement en demandant une augmentation de quota. Pour plus d'informations, reportez-vous à la section [Demande d'augmentation de quota](#).

Amazon MWAA prend-il en charge le partage d'Amazon VPCs ou les sous-réseaux partagés ?

Amazon MWAA ne prend pas en charge le partage d'Amazon VPCs ou les sous-réseaux partagés. L'Amazon VPC que vous sélectionnez lorsque vous créez un environnement doit appartenir au compte qui tente de créer l'environnement. Cependant, vous pouvez acheminer le trafic d'un Amazon VPC du compte Amazon MWAA vers un VPC partagé. Pour plus d'informations et un exemple de routage du trafic vers un Amazon VPC partagé, reportez-vous à la section [Routage sortant centralisé vers Internet du guide des passerelles](#) de transit Amazon VPC.

Puis-je créer ou intégrer des files d'attente Amazon SQS personnalisées pour gérer l'exécution des tâches et l'orchestration des flux de travail dans Apache Airflow ?

Non, vous ne pouvez pas créer, modifier ou utiliser des files d'attente Amazon SQS personnalisées dans Amazon MWAA. Cela est dû au fait qu'Amazon MWAA provisionne et gère automatiquement sa propre file d'attente Amazon SQS pour chaque environnement Amazon MWAA.

Métriques

Quels indicateurs sont utilisés pour déterminer s'il convient de faire évoluer les effectifs ?

Amazon MWAA surveille le QueuedTaskset l'entrée RunningTasks CloudWatch afin de déterminer s'il convient d'adapter les travailleurs d'Apache Airflow à votre environnement. Pour en savoir plus, reportez-vous à [Surveillance et mesures](#).

Puis-je créer des métriques personnalisées dans CloudWatch ?

Pas sur CloudWatch console. Cependant, vous pouvez créer un DAG dans lequel des métriques personnalisées sont enregistrées CloudWatch. Pour plus d'informations, consultez [the section called "Utiliser un DAG pour écrire des métriques personnalisées"](#).

DAGs, Opérateurs, connexions et autres questions

Puis-je utiliser le **PythonVirtualenvOperator** ?

Le `PythonVirtualenvOperator` n'est pas explicitement pris en charge sur Amazon MWAA, mais vous pouvez créer un plugin personnalisé qui utilise le `PythonVirtualenvOperator`. Pour un exemple de code, reportez-vous à [the section called “Plugin personnalisé à patcher PythonVirtualenvOperator”](#).

Combien de temps faut-il à Amazon MWAA pour reconnaître un nouveau fichier DAG ?

DAGs sont régulièrement synchronisés entre le compartiment Amazon S3 et votre environnement. Si vous ajoutez un nouveau fichier DAG, Amazon MWAA met environ 300 secondes pour commencer à utiliser le nouveau fichier. Si vous mettez à jour un DAG existant, Amazon MWAA met environ 30 secondes à reconnaître vos mises à jour.

Ces valeurs, 300 secondes pour les nouvelles DAGs et 30 secondes pour les mises à jour des versions existantes DAGs, correspondent respectivement aux options [dag_dir_list_interval](#) de configuration d'Apache Airflow. [min_file_process_interval](#)

Pourquoi mon fichier DAG n'est-il pas récupéré par Apache Airflow ?

Les solutions possibles à ce problème sont les suivantes :

1. Vérifiez que votre rôle d'exécution dispose d'autorisations suffisantes pour accéder à votre compartiment Amazon S3. Pour en savoir plus, reportez-vous à [Rôle d'exécution Amazon MWAA](#).
2. Vérifiez que le compartiment Amazon S3 est configuré pour bloquer l'accès public et que le contrôle de version est activé. Pour en savoir plus, reportez-vous à [Création d'un compartiment Amazon S3 pour Amazon MWAA](#).
3. Vérifiez le fichier DAG lui-même. Par exemple, assurez-vous que chaque DAG possède un ID DAG unique.

Puis-je supprimer un environnement **plugins.zip** ou le supprimer **requirements.txt** d'un environnement ?

Actuellement, il n'existe aucun moyen de supprimer un fichier `plugins.zip` ou `requirements.txt` d'un environnement une fois qu'ils ont été ajoutés, mais nous travaillons sur le problème. Dans l'intervalle, une solution consiste à pointer vers un fichier texte ou un fichier zip vide, respectivement. Pour en savoir plus, reportez-vous à [Suppression de fichiers sur Amazon S3](#).

Pourquoi mes plugins n'apparaissent-ils pas dans le menu des plugins d'administration d'Apache Airflow v2.0.2 ?

Pour des raisons de sécurité, le serveur Web Apache Airflow sur Amazon MWAA dispose d'une sortie réseau limitée et n'installe pas de plugins ni de dépendances Python directement sur le serveur Web Apache Airflow pour les environnements de version 2.0.2. Le plugin répertorié permet à Amazon MWAA d'authentifier vos utilisateurs d'Apache Airflow Gestion des identités et des accès AWS dans (IAM).

Pour pouvoir installer des plugins et des dépendances Python directement sur le serveur Web, nous vous recommandons de créer un nouvel environnement avec Apache Airflow v2.2 et versions ultérieures. Amazon MWAA installe les dépendances Python et les plug-ins personnalisés directement sur le serveur Web pour Apache Airflow v2.2 et versions ultérieures.

Puis-je utiliser les opérateurs du Service AWS de Migration de Base de Données (DMS) ?

Amazon MWAA prend en charge les opérateurs [DMS](#). Toutefois, cet opérateur ne peut pas être utilisé pour effectuer des actions sur la base de données de métadonnées Amazon Aurora PostgreSQL associée à un environnement Amazon MWAA.

Lorsque j'accède à l'API REST Airflow à l'aide des AWS informations d'identification, puis-je augmenter la limite de régulation à plus de 10 transactions par seconde (TPS) ?

Oui, vous pouvez. Pour augmenter la limite de régulation, veuillez contacter le [AWS Support](#) client.

Où s'exécute le serveur d'API d'exécution des tâches Airflow dans Amazon MWAA ?

Amazon MWAA exécute le serveur d'API d'exécution des tâches Airflow dans le composant du serveur Web. L'exécution APIs des tâches n'est disponible que dans Apache Airflow v3 et versions ultérieures. Pour plus d'informations sur l'architecture Amazon MWAA, reportez-vous à [Architecture](#).

Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow

Ce chapitre décrit les problèmes et erreurs courants que vous pouvez rencontrer lors de l'utilisation d'Apache Airflow sur Amazon Managed Workflows for Apache Airflow et les étapes recommandées pour résoudre ces erreurs.

Table des matières

- [Résolution des problèmes : opérateurs DAGs, connexions et autres problèmes](#)
 - [Connexions](#)
 - [Je n'arrive pas à me connecter à Secrets Manager](#)
 - [Comment configurer les conditions du gestionnaire de secretsmanager:ResourceTag/<tag-key> secrets ou une restriction de ressources dans ma politique de rôle d'exécution ?](#)
 - [Je n'arrive pas à me connecter à Snowflake](#)
 - [Je ne trouve pas ma connexion dans l'interface utilisateur d'Airflow](#)
 - [Serveur Web](#)
 - [Je reçois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Je reçois un The scheduler does not seem to be running message d'erreur](#)
 - [Tâches](#)
 - [Mes tâches sont bloquées ou ne sont pas terminées](#)
 - [J'ai des échecs de tâches sans journaux dans Airflow v3](#)
 - [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je reçois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)
 - [Pourquoi la commande dags backfill Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution de contournement ?](#)
 - [Opérateurs](#)
 - [J'ai reçu une PermissionError: \[Errno 13\] Permission denied erreur lors de l'utilisation de l'opérateur S3Transform](#)
- [Résolution des problèmes : création et mise à jour d'un environnement Amazon MWAA](#)
 - [Mise à jour des requirements.txt](#)

- [J'ai spécifié une nouvelle version de mon environnement requirements.txt et la mise à jour de mon environnement prend plus de 20 minutes](#)
- [Plug-ins](#)
 - [Amazon MWAA prend-il en charge la mise en œuvre d'une interface utilisateur personnalisée ?](#)
- [Créer un bucket](#)
 - [Je ne parviens pas à sélectionner l'option pour les paramètres S3 Block Public Access](#)
- [Créer un environnement](#)
 - [J'ai essayé de créer un environnement et il est resté dans l'Creating état](#)
 - [J'ai essayé de créer un environnement mais il affiche le statut comme Create failed](#)
 - [J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur Network Failure](#)
 - [J'ai essayé de créer un environnement et j'ai reçu une erreur de service, de partition ou de ressource « doit être transmise »](#)
 - [J'ai essayé de créer un environnement et il affiche l'état, Available mais lorsque j'essaie d'accéder à l'interface utilisateur Airflow, une 502 Bad Gateway erreur Empty Reply from Server ou s'affiche](#)
 - [J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires](#)
- [Mettre à jour l'environnement](#)
 - [J'ai essayé de changer la classe d'environnement mais la mise à jour a échoué](#)
- [Environnement d'accès](#)
 - [Je n'arrive pas à accéder à l'interface utilisateur d'Apache Airflow](#)
- [Résolution des problèmes : CloudWatch journaux et CloudTrail erreurs](#)
- [Journaux](#)
 - [Je ne trouve pas mes journaux de tâches ou j'ai reçu un Reading remote log from Cloudwatch log_group message d'erreur](#)
 - [Les tâches échouent sans aucun journal](#)
 - [Je reçois une ResourceAlreadyExistsException erreur dans CloudTrail](#)
 - [Je reçois une Invalid request erreur dans CloudTrail](#)
 - [Je reçois Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory des logs d'Apache Airflow](#)

- [Je reçois « le serveur a fermé la connexion de façon inattendue » sur psycpg2 dans les journaux de mon planificateur](#)
- [J'accède Executor reports task instance %s finished \(%s\) although the task says its %s aux journaux de traitement de mon DAG](#)
- [J'accède Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:*{*DAG_ID}/{*TASK_ID}/{*time}/{*n}.log. à mes journaux de tâches](#)

Résolution des problèmes : opérateurs DAGs, connexions et autres problèmes

Les rubriques de cette page décrivent les solutions aux dépendances Python d'Apache Airflow v2 et v3, aux plugins personnalisés, aux opérateurs, aux connexions DAGs, aux tâches et aux problèmes de serveur Web que vous pouvez rencontrer dans un environnement Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [Connexions](#)
 - [Je n'arrive pas à me connecter à Secrets Manager](#)
 - [Comment configurer les conditions du gestionnaire de secretsmanager:ResourceTag/<tag-key> secrets ou une restriction de ressources dans ma politique de rôle d'exécution ?](#)
 - [Je n'arrive pas à me connecter à Snowflake](#)
 - [Je ne trouve pas ma connexion dans l'interface utilisateur d'Airflow](#)
- [Serveur Web](#)
 - [Je reçois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Je reçois un The scheduler does not seem to be running message d'erreur](#)
- [Tâches](#)
 - [Mes tâches sont bloquées ou ne sont pas terminées](#)
 - [J'ai des échecs de tâches sans journaux dans Airflow v3](#)
- [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je reçois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)
 - [Pourquoi la commande dags backfill Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution de contournement ?](#)

- [Opérateurs](#)

- [J'ai reçu une PermissionError: \[Errno 13\] Permission denied erreur lors de l'utilisation de l'opérateur S3Transform](#)

Connexions

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'utilisation d'une connexion Apache Airflow ou de l'utilisation d'une autre AWS base de données.

Je n'arrive pas à me connecter à Secrets Manager

Nous vous recommandons la procédure suivante :

1. Découvrez comment créer des clés secrètes pour votre connexion Apache Airflow et des variables dans. [the section called "Configuration de Secrets Manager"](#)
2. Apprenez à utiliser la clé secrète d'une variable Apache Airflow (test-variable) dans. [Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow](#)
3. Apprenez à utiliser la clé secrète pour une connexion Apache Airflow (myconn) dans. [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow](#)

Comment configurer les conditions du gestionnaire de **secretsmanager:ResourceTag/<tag-key>** secrets ou une restriction de ressources dans ma politique de rôle d'exécution ?

Note

S'applique à Apache Airflow version 2.0 et antérieures.

Actuellement, vous ne pouvez pas limiter l'accès aux secrets de Secrets Manager en utilisant des clés de condition ou d'autres restrictions de ressources dans le rôle d'exécution de votre environnement, en raison d'un problème connu dans Apache Airflow.

Je n'arrive pas à me connecter à Snowflake

Nous vous recommandons la procédure suivante :

1. Testez vos DAGs plugins personnalisés et vos dépendances Python localement en utilisant [aws-mwaa-docker-images](#) on GitHub.
2. Ajoutez les entrées suivantes au fichier requirements.txt correspondant à votre environnement.

```
apache-airflow-providers-snowflake==1.3.0
```

3. Ajoutez les importations suivantes à votre DAG :

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

Assurez-vous que l'objet de connexion Apache Airflow inclut les paires clé-valeur suivantes :

1. Identifiant du Connecticut : snowflake_conn
2. Type de connecteur : flocon de neige
3. Hôte : <my account>. <my region if not us-west-2>.snowflakecomputing.com
4. Schéma : <my schema>
5. Identifiant : <my user name>
6. Mot de passe : *****
7. Port : <port, if any>
8. Supplémentaire :

```
{
    "account": "<my account>",
    "warehouse": "<my warehouse>",
    "database": "<my database>",
    "region": "<my region if not using us-west-2 otherwise omit this line>"
}
```

Par exemple :

```
>>> import json
>>> from airflow.models.connection import Connection
>>> myconn = Connection(
...     conn_id='snowflake_conn',
...     conn_type='Snowflake',
...     host='123456789012.us-east-1.snowflakecomputing.com',
```

```
...     schema='YOUR_SCHEMA'
...     login='YOUR_USERNAME',
...     password='YOUR_PASSWORD',
...     port='YOUR_PORT'
...     extra=json.dumps(dict(account='123456789012', warehouse='YOUR_WAREHOUSE',
database='YOUR_DB_OPTION', region='us-east-1')),
... )
```

Je ne trouve pas ma connexion dans l'interface utilisateur d'Airflow

Apache Airflow fournit des modèles de connexion dans l'interface utilisateur d'Apache Airflow. Il l'utilise pour générer la chaîne d'URI de connexion, quel que soit le type de connexion. Si aucun modèle de connexion n'est disponible dans l'interface utilisateur d'Apache Airflow, un autre modèle de connexion peut être utilisé pour générer une chaîne d'URI de connexion, par exemple en utilisant le modèle de connexion HTTP.

Nous vous recommandons la procédure suivante :

1. Accédez aux types de connexion fournis par Amazon MWAA dans l'interface utilisateur d'Apache Airflow à l'adresse. [Packages du fournisseur Apache Airflow installés sur les environnements Amazon MWAA](#)
2. Accédez aux commandes pour créer une connexion Apache Airflow dans la CLI à l'adresse. [Référence des commandes de la CLI Apache Airflow](#)
3. Découvrez comment utiliser les modèles de connexion dans l'interface utilisateur d'Apache Airflow de manière interchangeable pour les types de connexion qui ne sont pas disponibles dans l'interface utilisateur d'Apache Airflow sur Amazon MWAA à l'adresse. [Vue d'ensemble des types de connexion](#)

Serveur Web

La rubrique suivante décrit les erreurs que vous pouvez recevoir pour votre serveur Web Apache Airflow sur Amazon MWAA.

Je reçois une erreur 5xx lors de l'accès au serveur Web

Nous vous recommandons la procédure suivante :

1. Vérifiez les options de configuration d'Apache Airflow. Vérifiez que les paires clé-valeur que vous avez spécifiées en tant qu'option de configuration d'Apache Airflow, par exemple, ont

été correctement AWS Secrets Manager configurées. Pour en savoir plus, reportez-vous à [the section called “Je n'arrive pas à me connecter à Secrets Manager”](#).

2. Vérifiez le `requirements.txt`. Vérifiez que le package « extras » Airflow et les autres bibliothèques répertoriées dans le votre `requirements.txt` sont compatibles avec votre version d'Apache Airflow.
3. Découvrez comment spécifier les dépendances Python dans un `requirements.txt` fichier, reportez-vous à [Gestion des dépendances Python dans requirements.txt](#).

Je reçois un **The scheduler does not seem to be running** message d'erreur

Si le planificateur ne semble pas fonctionner ou si le dernier « battement de cœur » a été reçu il y a plusieurs heures, il se peut que vous ne soyez pas répertorié dans Apache Airflow et aucune nouvelle tâche ne sera planifiée.

Nous vous recommandons la procédure suivante :

1. Vérifiez que votre groupe de sécurité VPC autorise l'accès entrant au port. 5432 Ce port est nécessaire pour se connecter à la base de données de métadonnées Amazon Aurora PostgreSQL de votre environnement. Une fois cette règle ajoutée, accordez quelques minutes à Amazon MWAA pour que l'erreur disparaisse. Pour en savoir plus, reportez-vous à [the section called “Sécurité dans votre VPC”](#).

 Note

- La métadatabase Aurora PostgreSQL fait partie de l'architecture du service [Amazon MWAA et n'est pas disponible](#) dans votre. Compte AWS
- Les erreurs liées à la base de données sont généralement le symptôme d'une défaillance du planificateur et n'en sont pas la cause première.

2. Si le planificateur n'est pas en cours d'exécution, cela peut être dû à un certain nombre de facteurs tels que l'[échec de l'installation des dépendances](#) ou un planificateur [surchargé](#). Vérifiez que vos DAGs plug-ins et vos exigences fonctionnent correctement en accédant aux groupes de CloudWatch journaux correspondants dans Logs. Pour en savoir plus, reportez-vous à [Surveillance et mesures](#).

Tâches

La rubrique suivante décrit les erreurs que vous pouvez recevoir pour les tâches Apache Airflow dans un environnement.

Mes tâches sont bloquées ou ne sont pas terminées

Si vos tâches Apache Airflow sont « bloquées » ou ne se terminent pas, nous vous recommandons de suivre les étapes suivantes :

1. Il peut y avoir un grand nombre de DAGs définitions. Réduisez le nombre d'environnements DAGs et mettez-les à jour (par exemple en modifiant le niveau d'un journal) pour forcer une réinitialisation.
 - a. Airflow analyse DAGs s'ils sont activés ou non. Si vous utilisez plus de 50 % de la capacité de votre environnement, vous risquez de surcharger le planificateur Apache Airflow. Cela entraîne un temps d'analyse total important dans les CloudWatch métriques ou de longs délais de traitement des DAG dans CloudWatch les journaux. Il existe d'autres moyens d'optimiser les configurations d'Apache Airflow qui sortent du cadre de ce guide.
 - b. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, reportez-vous à [the section called “Optimisation des performances pour Apache Airflow”](#).
2. La file d'attente peut contenir un grand nombre de tâches. Cela se traduit souvent par un nombre important (et croissant) de tâches dans l'NoneÉtat, ou par un nombre important de tâches dans l'État. Queued Tasks and/or Tasks Pending CloudWatch Cela peut se produire pour les raisons suivantes :
 - a. S'il y a plus de tâches à exécuter que l'environnement n'en a la capacité, and/or un grand nombre de tâches mises en file d'attente avant le dimensionnement automatique ont le temps de les détecter et de déployer du personnel supplémentaire.
 - b. S'il y a plus de tâches à exécuter qu'un environnement n'en a la capacité, nous vous recommandons de réduire le nombre de tâches que vous DAGs exécutez simultanément, en and/or augmentant le nombre minimum de travailleurs Apache Airflow.
 - c. Si un grand nombre de tâches ont été mises en file d'attente avant que le scalage automatique n'ait eu le temps de détecter et de déployer des travailleurs supplémentaires, nous vous recommandons d'échelonner le déploiement des tâches en and/or augmentant le nombre minimum de travailleurs Apache Airflow.

- d. Vous pouvez utiliser la commande [update-environment](#) dans le AWS Command Line Interface (AWS CLI) pour modifier le nombre minimum ou maximum de travailleurs exécutés sur votre environnement.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-workers 10
```

- e. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, reportez-vous à [the section called "Optimisation des performances pour Apache Airflow"](#).
3. Si vos tâches sont bloquées dans l'état « en cours », vous pouvez également les effacer ou les marquer comme réussies ou échouées. Cela permet au composant de mise à l'échelle automatique de votre environnement de réduire le nombre de travailleurs travaillant sur votre environnement. L'image suivante illustre un exemple de tâche bloquée.
 - Choisissez le cercle pour la tâche bloquée, puis sélectionnez Effacer (comme indiqué). Cela permet à Amazon MWAA de réduire le nombre de travailleurs ; dans le cas contraire, Amazon MWAA ne peut pas déterminer quelles tâches DAGs sont activées ou désactivées, et ne peut pas réduire ses effectifs s'il y a encore des tâches en file d'attente.
 4. Pour en savoir plus sur le cycle de vie des tâches d'Apache Airflow, consultez [Concepts](#) dans le guide de référence d'Apache Airflow.

J'ai des échecs de tâches sans journaux dans Airflow v3

Si vos tâches Apache Airflow 3 échouent sans journaux, procédez comme suit :

- Si les journaux de travail présentent une erreur, par exemple au Task handler raised error: `WorkerLostError('Worker exited prematurely: exitcode 15 Job: 12.')` moment de l'échec de la tâche, cela indique que le processus de travail bifurqué affecté à la tâche s'est probablement arrêté de manière inattendue.

Pour résoudre ce problème, pensez à configurer `celery.worker_autoscale` avec les mêmes valeurs minimales et maximales. Par exemple :

```
celery.worker_autoscale=5,5 # for mw1.small
celery.worker_autoscale=10,10 # for mw1.medium
```

```
celery.worker_autoscale=20,20 # for mw1.large
```

Cela garantit que la taille du pool de travailleurs reste fixe, empêchant ainsi les licenciements inattendus de travailleurs.

INTERFACE DE LIGNE DE COMMANDE (CLI)

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'exécution des commandes de la CLI Airflow dans le AWS Command Line Interface.

Je reçois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI

La CLI Airflow s'exécute sur le serveur Web Apache Airflow, dont la simultanéité est limitée. Généralement, un maximum de 4 commandes CLI peuvent être exécutées simultanément.

Pourquoi la commande **dags backfill** Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution de contournement ?

Note

Ce qui suit s'applique uniquement aux environnements Apache Airflow v2.0.2.

La **backfill** commande, comme les autres commandes de la CLI Apache Airflow, analyse toutes les données DAGs localement avant qu'elles ne DAGs soient traitées, quel que soit le DAG auquel s'applique l'opération de la CLI. Dans les environnements Amazon MWAA utilisant Apache Airflow v2.0.2, étant donné que les plug-ins et les exigences ne sont pas encore installés sur le serveur Web au moment de l'exécution de la commande CLI, l'opération d'analyse échoue et l'opération n'est pas invoquée. **backfill** Si vous n'aviez aucune exigence ni aucun plugin dans votre environnement, l'opération réussirait.

Pour pouvoir exécuter la commande **backfill** CLI, nous vous recommandons de l'invoquer dans un opérateur bash. Dans un opérateur bash, **backfill** il est lancé par le travailleur, ce qui permet DAGs à l'analyse de fonctionner correctement car toutes les exigences et tous les plugins nécessaires sont disponibles et installés. Utilisez l'exemple suivant pour créer un DAG avec un **BashOperator** to run **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"
    )
```

Opérateurs

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'utilisation des opérateurs.

J'ai reçu une **PermissionError: [Errno 13] Permission denied** erreur lors de l'utilisation de l'opérateur S3Transform

Nous vous recommandons de suivre les étapes suivantes si vous essayez d'exécuter un script shell avec l'opérateur S3Transform et que vous recevez un **PermissionError: [Errno 13] Permission denied** message d'erreur. Les étapes suivantes supposent que vous disposez d'un fichier `plugins.zip` existant. Si vous créez un nouveau fichier `plugins.zip`, reportez-vous à [Installation de plugins personnalisés](#).

1. Testez vos DAGs plugins personnalisés et vos dépendances Python localement en utilisant [aws-mwaa-docker-images](#) on GitHub.
2. Créez votre script de « transformation ».

```
#!/bin/bash
cp $1 $2
```

3. (facultatif) Les utilisateurs de macOS et Linux devront peut-être exécuter la commande suivante pour s'assurer que le script est exécutable.

```
chmod 777 transform_test.sh
```

4. Ajoutez le script à votre fichier `plugins.zip`.

```
zip plugins.zip transform_test.sh
```

5. Suivez les étapes décrites dans [Importer le fichier plugins.zip sur Amazon S3](#).

6. Suivez les étapes décrites dans [Spécifier la version du fichier plugins.zip sur la console Amazon MWAA](#).
7. Créez le DAG suivant.

```
from airflow import DAG
    from airflow.providers.amazon.aws.operators.s3_file_transform import
    S3FileTransformOperator
    from airflow.utils.dates import days_ago
    import os

    DAG_ID = os.path.basename(__file__).replace(".py", "")

    with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False,
start_date=days_ago(1)) as dag:
        file_transform = S3FileTransformOperator(
            task_id='file_transform',
            transform_script='/usr/local/airflow/plugins/transform_test.sh',
            source_s3_key='s3://amzn-s3-demo-bucket/files/input.txt',
            dest_s3_key='s3://amzn-s3-demo-bucket/files/output.txt'
        )
```

8. Suivez les étapes décrites dans la [section Chargement du code DAG sur Amazon S3](#).

Résolution des problèmes : création et mise à jour d'un environnement Amazon MWAA

Les rubriques de cette page contiennent des erreurs que vous pouvez rencontrer lors de la création et de la mise à jour d'un environnement Amazon Managed Workflows pour Apache Airflow et indiquent comment résoudre ces erreurs.

Table des matières

- [Mise à jour des requirements.txt](#)
 - [J'ai spécifié une nouvelle version de mon environnement requirements.txt et la mise à jour de mon environnement prend plus de 20 minutes](#)
- [Plug-ins](#)
 - [Amazon MWAA prend-il en charge la mise en œuvre d'une interface utilisateur personnalisée ?](#)
- [Créer un bucket](#)
 - [Je ne parviens pas à sélectionner l'option pour les paramètres S3 Block Public Access](#)

- [Créer un environnement](#)
 - [J'ai essayé de créer un environnement et il est resté dans l'Creating état](#)
 - [J'ai essayé de créer un environnement mais il affiche le statut comme Create failed](#)
 - [J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur Network Failure](#)
 - [J'ai essayé de créer un environnement et j'ai reçu une erreur de service, de partition ou de ressource « doit être transmise »](#)
 - [J'ai essayé de créer un environnement et il affiche l'état, Available mais lorsque j'essaie d'accéder à l'interface utilisateur Airflow, une 502 Bad Gateway erreur Empty Reply from Server ou s'affiche](#)
 - [J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires](#)
- [Mettre à jour l'environnement](#)
 - [J'ai essayé de changer la classe d'environnement mais la mise à jour a échoué](#)
- [Environnement d'accès](#)
 - [Je n'arrive pas à accéder à l'interface utilisateur d'Apache Airflow](#)

Mise à jour des **requirements.txt**

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la mise à jour de votre `requirements.txt`.

J'ai spécifié une nouvelle version de mon environnement **requirements.txt** et la mise à jour de mon environnement prend plus de 20 minutes

Si votre environnement met plus de vingt minutes à installer une nouvelle version d'un `requirements.txt` fichier, la mise à jour de l'environnement échoue et Amazon MWAA revient à la dernière version stable de l'image du conteneur.

1. Vérifiez les versions des packages. Nous vous recommandons de toujours spécifier une version spécifique (==) ou une version maximale (<=) pour les dépendances Python de votre `requirements.txt`.
2. Consultez les journaux d'Apache Airflow. Si vous avez activé les journaux Apache Airflow, vérifiez que vos groupes de journaux ont bien été créés sur la [page des groupes de journaux](#) de la CloudWatch console. Si vous obtenez des journaux vides, la raison la plus courante est

- l'absence d'autorisations dans votre rôle d'exécution pour Amazon S3 CloudWatch ou dans l'endroit où les journaux sont écrits. Pour en savoir plus, reportez-vous à [Rôle d'exécution](#).
3. Vérifiez les options de configuration d'Apache Airflow. Si vous utilisez Secrets Manager, vérifiez que les paires clé-valeur que vous avez spécifiées comme option de configuration d'Apache Airflow ont été correctement configurées. Pour en savoir plus, reportez-vous à [the section called "Configuration de Secrets Manager"](#).
 4. Vérifiez la configuration du réseau VPC. Pour en savoir plus, reportez-vous à [the section called "Environnement bloqué"](#).
 5. Vérifiez les autorisations des rôles d'exécution. Un rôle d'exécution est un rôle Gestion des identités et des accès AWS (IAM) doté d'une politique d'autorisations qui autorise Amazon MWAA à invoquer les ressources d'autres AWS services (tels qu'Amazon S3, Amazon SQS CloudWatch, Amazon ECR) en votre nom. L'accès à votre [clé gérée par le client](#) ou à votre [clé AWS détenue par](#) le client doit également être autorisé. Pour en savoir plus, reportez-vous à [Rôle d'exécution](#).
 6. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur GitHub

Plug-ins

La rubrique suivante décrit les problèmes que vous pouvez rencontrer lors de la configuration ou de la mise à jour des plugins Apache Airflow.

Amazon MWAA prend-il en charge la mise en œuvre d'une interface utilisateur personnalisée ?

À partir de la version 2.2.2 d'Apache Airflow, Amazon MWAA prend en charge l'installation de plug-ins sur le serveur Web Apache Airflow et la mise en œuvre d'une interface utilisateur personnalisée. Si votre environnement Amazon MWAA exécute Apache Airflow v2.0.2 ou une version antérieure, vous ne pourrez pas implémenter d'interface utilisateur personnalisée.

Pour plus d'informations sur la gestion des versions et la mise à niveau de vos environnements existants, reportez-vous à [Versions](#).

Créer un bucket

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la création d'un compartiment Amazon S3.

Je ne parviens pas à sélectionner l'option pour les paramètres S3 Block Public Access

Le [rôle d'exécution](#) de votre environnement Amazon MWAA nécessite l'autorisation d'effectuer une `GetBucketPublicAccessBlock` action sur le compartiment Amazon S3 afin de vérifier que celui-ci bloque l'accès public. Nous vous recommandons la procédure suivante :

1. Suivez les étapes pour [associer une politique JSON à votre rôle d'exécution](#).
2. Joignez la politique JSON suivante :

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ]
}
```

Remplacez les exemples d'espaces réservés *amzn-s3-demo-bucket* par le nom de votre compartiment Amazon S3.

3. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur. GitHub

Créer un environnement

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la création d'un environnement.

J'ai essayé de créer un environnement et il est resté dans l'**Creating**état

Nous vous recommandons la procédure suivante :

1. Vérifiez le réseau VPC avec un routage public. Si vous utilisez un Amazon VPC avec accès à Internet, vérifiez les points suivants :
 - Que votre Amazon VPC soit configuré pour autoriser le trafic réseau entre les différentes AWS ressources utilisées par votre environnement Amazon MWAA, comme défini dans. [the section called “À propos du réseau”](#) Par exemple, votre groupe de sécurité VPC doit soit autoriser tout le trafic dans une règle d'autoréférencement, soit éventuellement spécifier la plage de ports pour la plage de ports HTTPS 443 et une plage de ports TCP 5432.
2. Vérifiez le réseau VPC avec un routage privé. Si vous utilisez un Amazon VPC sans accès à Internet, vérifiez les points suivants :
 - Que votre Amazon VPC est configuré pour autoriser le trafic réseau entre les différentes AWS ressources de votre environnement Amazon MWAA, comme défini dans. [the section called “À propos du réseau”](#) Par exemple, vos deux sous-réseaux privés ne doivent pas avoir de table de routage vers une passerelle NAT (ou instance NAT), ni une passerelle Internet.
3. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur. GitHub

J'ai essayé de créer un environnement mais il affiche le statut comme **Create failed**

Nous vous recommandons la procédure suivante :

1. Vérifiez la configuration du réseau VPC. Pour en savoir plus, reportez-vous à [the section called “Environnement bloqué”](#).
2. Vérifiez les autorisations des utilisateurs. Amazon MWAA effectue une analyse à sec des informations d'identification d'un utilisateur avant de créer un environnement. Vous n'êtes Compte AWS peut-être pas autorisé Gestion des identités et des accès AWS (IAM) à créer certaines ressources pour un environnement. Par exemple, si vous avez choisi le mode d'accès Apache Airflow au réseau privé, votre administrateur Compte AWS doit vous avoir

accordé l'accès à la politique de contrôle d'MWAAFullConsoleAccessaccès [Amazon](#) pour votre environnement, qui permet à votre compte de créer des points de terminaison VPC.

3. Vérifiez les autorisations des rôles d'exécution. Un rôle d'exécution est un rôle Gestion des identités et des accès AWS (IAM) doté d'une politique d'autorisations qui autorise Amazon MWAA à invoquer les ressources d'autres AWS services (tels qu'Amazon S3, Amazon SQS CloudWatch, Amazon ECR) en votre nom. L'accès à votre [clé gérée par le client](#) ou à votre [cléAWS détenue par](#) le client doit également être autorisé. Pour en savoir plus, reportez-vous à [Rôle d'exécution](#).
4. Consultez les journaux d'Apache Airflow. Si vous avez activé les journaux Apache Airflow, vérifiez que vos groupes de journaux ont bien été créés sur la [page des groupes de journaux](#) de la CloudWatch console. Si vous obtenez des journaux vides, la raison la plus courante est l'absence d'autorisations dans votre rôle d'exécution pour Amazon S3 CloudWatch ou dans l'endroit où les journaux sont écrits. Pour en savoir plus, reportez-vous à [Rôle d'exécution](#).
5. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur GitHub
6. Si vous utilisez un Amazon VPC sans accès à Internet, assurez-vous d'avoir créé un point de terminaison de passerelle Amazon S3 et d'avoir accordé les autorisations minimales requises à Amazon ECR pour accéder à Amazon S3. Pour en savoir plus sur la création d'un point de terminaison de passerelle Amazon S3, consultez ce qui suit :
 - [Création d'un réseau Amazon VPC sans accès à Internet](#)
 - [Créez le point de terminaison de la passerelle Amazon S3](#) dans le guide de l'utilisateur d'Amazon Elastic Container Registry

J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur **Network Failure**

Nous vous recommandons la procédure suivante :

- Si un Network Failure message d'erreur s'affiche lorsque vous essayez de sélectionner un Amazon VPC lors de la création de votre environnement, désactivez tous les proxys du navigateur en cours d'exécution, puis réessayez.

J'ai essayé de créer un environnement et j'ai reçu une erreur de service, de partition ou de ressource « doit être transmise »

Nous vous recommandons la procédure suivante :

- Vous recevez peut-être cette erreur car l'URI que vous avez spécifié pour votre compartiment Amazon S3 inclut un «/» à la fin de l'URI. Nous vous recommandons de supprimer le «/» dans le chemin. La valeur doit être au format suivant :

```
s3://amzn-s3-demo-bucket
```

J'ai essayé de créer un environnement et il affiche l'état, **Available** mais lorsque j'essaie d'accéder à l'interface utilisateur Airflow, une **502 Bad Gateway** erreur **Empty Reply from Server** ou s'affiche

Nous vous recommandons la procédure suivante :

1. Vérifiez la configuration du groupe de sécurité VPC. Pour en savoir plus, reportez-vous à [the section called “Environnement bloqué”](#).
2. Vérifiez que tous les packages Apache Airflow que vous avez répertoriés `requirements.txt` correspondent à la version d'Apache Airflow que vous utilisez sur Amazon MWAA. Pour en savoir plus, reportez-vous à [Installation des dépendances Python](#).
3. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur GitHub

J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires

- Apache Airflow possède un maximum de 64 caractères pour les noms d'utilisateur. Si votre rôle Gestion des identités et des accès AWS (IAM) dépasse cette longueur, un algorithme de hachage est utilisé pour le réduire, tout en restant unique.

Mettre à jour l'environnement

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la mise à jour d'un environnement.

J'ai essayé de changer la classe d'environnement mais la mise à jour a échoué

Si vous mettez à jour votre environnement vers une autre classe d'environnement (par exemple en remplaçant un par `unmw1.small`) et que la demande de mise à jour de votre environnement échoue, l'`UPDATE_FAILED` état de l'environnement passe à un état et l'environnement est rétabli et facturé conformément à la version stable précédente d'un environnement. `mw1.medium`

Nous vous recommandons la procédure suivante :

1. Testez vos DAGs plugins personnalisés et vos dépendances Python localement en utilisant [aws-mwaa-docker-images](#) sur GitHub.
2. Pour exécuter un script de dépannage qui vérifie la configuration du réseau Amazon VPC et la configuration de votre environnement Amazon MWAA, reportez-vous au script [Verify Environment](#) dans AWS Support Tools sur. GitHub

Environnement d'accès

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'accès à un environnement.

Je n'arrive pas à accéder à l'interface utilisateur d'Apache Airflow

Nous vous recommandons la procédure suivante :

1. Vérifiez les autorisations des utilisateurs. Vous n'avez peut-être pas obtenu l'accès à une politique d'autorisation que vous pouvez utiliser pour accéder à l'interface utilisateur d'Apache Airflow. Pour en savoir plus, reportez-vous à [the section called “Accès à un environnement Amazon MWAA”](#).
2. Vérifiez l'accès au réseau. Cela peut être dû au fait que vous avez sélectionné le mode d'accès au réseau privé. Si l'URL de votre interface utilisateur Apache Airflow est au format suivant `387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpc.c10.us-west-2.airflow.amazonaws.com`, cela signifie que vous utilisez un routage privé pour votre serveur Web Apache Airflow. Vous pouvez soit mettre à jour le mode d'accès Apache Airflow vers le mode d'accès au réseau public, soit créer un mécanisme pour accéder au point

de terminaison VPC pour votre serveur Web Apache Airflow. Pour en savoir plus, reportez-vous [à la section called "Gestion de l'accès aux points de terminaison VPC"](#).

Résolution des problèmes : CloudWatch journaux et CloudTrail erreurs

Les rubriques de cette page présentent les solutions apportées à Amazon CloudWatch Logs et AWS CloudTrail les erreurs que vous pouvez rencontrer dans un environnement Amazon Managed Workflows for Apache Airflow.

Table des matières

- [Journaux](#)
 - [Je ne trouve pas mes journaux de tâches ou j'ai reçu un Reading remote log from Cloudwatch log_group message d'erreur](#)
 - [Les tâches échouent sans aucun journal](#)
 - [Je reçois une ResourceAlreadyExistsException erreur dans CloudTrail](#)
 - [Je reçois une Invalid request erreur dans CloudTrail](#)
 - [Je reçois Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory des logs d'Apache Airflow](#)
 - [Je reçois « le serveur a fermé la connexion de façon inattendue » sur psycopg2 dans les journaux de mon planificateur](#)
 - [J'accède Executor reports task instance %s finished \(%s\) although the task says its %s aux journaux de traitement de mon DAG](#)
 - [J'accède Could not read remote logs from log_group: airflow-{{*environmentName}}-Task log_stream: {{*DAG_ID}}/{{*TASK_ID}}/{{*time}}/{{*n}}.log. à mes journaux de tâches](#)

Journaux

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'accès aux journaux d'Apache Airflow.

Je ne trouve pas mes journaux de tâches ou j'ai reçu un **Reading remote log from Cloudwatch log_group** message d'erreur

Amazon MWAA a configuré Apache Airflow pour lire et écrire des journaux directement depuis et vers Amazon CloudWatch Logs. Si un travailleur ne démarre pas une tâche ou n'écrit aucun journal, vous ferez référence à l'erreur suivante :

```
*** Reading remote log from Cloudwatch log_group: airflow-environmentName-Task
log_stream: DAG_ID/TASK_ID/timestamp/n.log.Could not read remote logs from log_group:
airflow-environmentName-Task log_stream: DAG_ID/TASK_ID/time/n.log.
```

- Nous vous recommandons la procédure suivante :
 - a. Vérifiez que vous avez activé les journaux de tâches au INFO niveau de votre environnement. Pour plus d'informations, consultez [Accès aux journaux Airflow sur Amazon CloudWatch](#).
 - b. Vérifiez que le [rôle d'exécution de l'environnement](#) dispose des politiques d'autorisation appropriées.
 - c. Vérifiez que votre opérateur ou votre tâche fonctionne correctement, qu'il dispose de suffisamment de ressources pour analyser le DAG et que les bibliothèques Python appropriées doivent être chargées. Pour vérifier si vous avez les bonnes dépendances, essayez d'éliminer les importations jusqu'à ce que vous trouviez celle à l'origine du problème. Nous vous recommandons de tester vos dépendances Python à l'aide de [aws-mwaa-docker-images](#).

Les tâches échouent sans aucun journal

Si des tâches échouent dans un flux de travail et que vous ne trouvez aucun journal pour les tâches ayant échoué, vérifiez si vous définissez le queue paramètre dans vos arguments par défaut, comme indiqué ci-dessous.

```
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'] }}"
    )

```

Pour résoudre le problème, supprimez-le queue de votre code et invoquez à nouveau le DAG.

Je reçois une **ResourceAlreadyExistsException** erreur dans CloudTrail

```

"errorCode": "ResourceAlreadyExistsException",
"errorMessage": "The specified log stream already exists",
"requestParameters": {
    "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
    "logStreamName": "scheduler_cross-account-eks.py.log"
}

```

Certaines exigences relatives à Python, `apache-airflow-backport-providers-amazon` telles que le retour à une ancienne version de la `watchtower` bibliothèque avec CloudWatch laquelle Amazon MWAA communique. Nous vous recommandons la procédure suivante :

- Ajoutez la bibliothèque suivante à votre `requirements.txt`

```
watchtower==1.0.6
```

Je reçois une **Invalid request** erreur dans CloudTrail

```
Invalid request provided: Provided role does not have sufficient permissions for s3
location airflow-xxx-xxx/dags
```

Si vous créez un environnement Amazon MWAA et un compartiment Amazon S3 à l'aide du même CloudFormation modèle, vous devez ajouter une `DependsOn` section dans votre CloudFormation modèle. Les deux ressources (MWAA Environment et MWAA Execution Policy) sont dépendantes de. CloudFormation Nous vous recommandons la procédure suivante :

- Ajoutez la **DependsOn** déclaration suivante à votre CloudFormation modèle.

```

...
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:
...

```

Pour un exemple, reportez-vous à [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#).

Je reçois **Cannot locate a 64-bit Oracle Client library:**
"libcintsh.so: cannot open shared object file: No such file or
directory des logs d'Apache Airflow

- Nous vous recommandons la procédure suivante :
 - Si vous utilisez Apache Airflow v2, ajoutez-le en `core.lazy_load_plugins : False` tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, reportez-vous à la section [Utilisation des options de configuration pour charger des plug-ins en 2](#).

Je reçois « le serveur a fermé la connexion de façon inattendue » sur psycopg2 dans les journaux de mon planificateur

Si vous obtenez une erreur similaire à la suivante, votre planificateur Apache Airflow est peut-être à court de ressources.

```
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.
```

Nous vous recommandons la procédure suivante :

- Envisagez de passer à Apache Airflow v2.0.2, que vous pouvez utiliser pour spécifier jusqu'à 5 planificateurs.

J'accède **Executor reports task instance %s finished (%s) although the task says its %s** aux journaux de traitement de mon DAG

Si vous recevez un message d'erreur similaire à ce qui suit, il est possible que vos tâches de longue durée aient atteint la limite de temps sur Amazon MWAA. Amazon MWAA impose une limite de 12 heures pour chaque tâche Airflow, afin d'éviter que les tâches ne restent bloquées dans la file d'attente et ne bloquent des activités telles que le dimensionnement automatique.

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
%s) Was the task killed externally
```

Nous vous recommandons la procédure suivante :

- Envisagez de diviser la tâche en plusieurs tâches plus courtes. Airflow utilise généralement un modèle dans lequel les opérateurs sont asynchrones. Il invoque des activités sur des systèmes externes et les capteurs Apache Airflow interrogent pour vérifier s'il est terminé. Si un capteur tombe en panne, il peut être réessayé en toute sécurité sans affecter les fonctionnalités de l'opérateur.

J'accède **Could not read remote logs from log_group: airflow-
{environmentName}-Task log_stream: {DAG_ID}/{TASK_ID}/
*{time}/{n}.log.** à mes journaux de tâches

Si vous obtenez une erreur similaire à la suivante, il est possible que le rôle d'exécution de votre environnement ne contienne pas de politique d'autorisation permettant de créer des flux de journaux pour les journaux de tâches.

```
Could not read remote logs from log_group: airflow-*{environmentName}-Task  
log_stream:* {DAG_ID}/{TASK_ID}/{time}/{n}.log.
```

Nous vous recommandons la procédure suivante :

- Modifiez le rôle d'exécution de votre environnement à l'aide de l'un des exemples de politiques disponibles sur [the section called “Rôle d'exécution”](#).

Vous avez peut-être également indiqué dans votre `requirements.txt` fichier un package de fournisseur incompatible avec votre version d'Apache Airflow. Par exemple, si vous utilisez Apache Airflow v2.0.2, vous avez peut-être spécifié un package, tel que le [apache-airflow-providers-databricks](#) package, qui n'est compatible qu'avec Airflow 2.1+.

Nous vous recommandons la procédure suivante :

1. Si vous utilisez Apache Airflow v2.0.2, modifiez le `requirements.txt` fichier et ajoutez-le. `apache-airflow[databricks]` Cela installe la version correcte du package Databricks compatible avec Apache Airflow v2.0.2.
2. Testez vos DAGs plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-docker-images](#) on GitHub.

Historique du guide de l'utilisateur Amazon MWAA

Le tableau suivant décrit les ajouts importants à la documentation du service Amazon MWAA, à compter de novembre 2020. Pour recevoir des notifications concernant les mises à jour de cette documentation, abonnez-vous au flux RSS.

Modification	Description	Date
Nouvelle version d'Apache Airflow	Amazon MWAA prend désormais en charge Apache Airflow v2.11.0. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, ainsi que des informations sur l'utilisation d'Apache Airflow v2.11.0 sur Amazon MWAA. • Versions • the section called “Packages de fournisseurs spécifiques à la version”	7 janvier 2026
Ajout du support d'Apache Airflow v3	Documentation mise à jour pour inclure le support d'Apache Airflow v3 • Exemples de code • Versions	1er octobre 2025
IPv6 mises à jour	Ajout d'informations sur le IPv6 support. • the section called “À propos du réseau”	26 août 2025
Mises à jour environnementales	Ajout d'une remarque concernant le workerRep	6 août 2025

lacementStrategy
passage de GRACEFUL à
FORCED si vous effectuez
une mise à jour alors que
votre environnement est en
MAINTENANCE état.

- [the section called “Mise à jour d’un environnement”](#)

[Informations relatives à l'obsolescence des versions](#)

Rubrique mise à jour sur la dépréciation des versions afin d'inclure des notifications et des chronologies d'obsolescence pour Apache Airflow v2.4.3, Apache Airflow v2.5.1 et Apache Airflow v2.6.3.

24 juin 2025

- [the section called “Versions obsolètes d'Apache Airflow”](#)

[Ajout d'une nouvelle classe d'environnement : mw1.micro](#)

Amazon MWAA propose désormais une nouvelle classe d'environnement : mw1.micro.

19 novembre 2024

- [the section called “Configuration de la classe d'environnement”](#)
- [the section called “Optimisation des performances pour Apache Airflow”](#)

[Support d'une méthode plus simple pour accéder à l'API REST Apache Airflow](#)

Amazon MWAA propose désormais une approche simplifiée pour interagir avec l'API AWS REST Apache Airflow à l'aide d'informations d'identification.

23 octobre 2024

- [the section called “Utilisation de l'API REST Apache Airflow”](#)
- [the section called “Accès à l'API Rest d'Apache Airflow”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.10.1. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, ainsi que des informations sur l'utilisation d'Apache Airflow v2.10.1 sur Amazon MWAA.

26 septembre 2024

- [Versions](#)
- [the section called “Packages de fournisseurs spécifiques à la version”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.9.2. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, ainsi que des informations sur l'utilisation d'Apache Airflow v2.9.2 sur Amazon MWAA.

9 juillet 2024

- [Versions](#)
- [the section called “Packages de fournisseurs spécifiques à la version”](#)

[Amazon MWAA prend en charge la configuration de noms de domaine de serveur Web personnalisés](#)

Amazon MWAA prend en charge la configuration de noms de domaine de serveur Web personnalisés pour les environnements privés sans accès à Internet. Cette mise à jour inclut la nouvelle rubrique suivante qui décrit la configuration d'un nouveau domaine personnalisé.

18 juin 2024

- [the section called “Configuration d'un domaine personnalisé”](#)

[Amazon MWAA prend en charge le dimensionnement automatique du serveur Web et l'API REST Apache Airflow](#)

Amazon MWAA prend désormais en charge le dimensionnement automatique des serveurs Web ainsi que la possibilité d'accéder à l'API REST Apache Airflow et de l'utiliser.

16 mai 2024

- [the section called “Configuration de la mise à l'échelle automatique du serveur Web”](#)
- [the section called “Utilisation de l'API REST Apache Airflow”](#)

[Description améliorée du comportement de mise à l'échelle automatique](#)

La rubrique suivante a été mise à jour pour refléter le nouveau comportement de dimensionnement automatique d'Amazon MWAA lorsque les employés prennent en charge de nouvelles tâches alors que les employés de Fargate réduisent leur effectif.

10 mai 2024

- [the section called “Configuration de la mise à l'échelle automatique des travailleurs”](#)

[Support pour des instances de plus grande taille](#)

Amazon MWAA prend désormais en charge deux options de taille d'instance plus importantes pour les charges de travail plus importantes : mw1.xlarge et mw1.2xlarge

16 avril 2024

- [the section called “Capacités environnementales”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.8.1. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, ainsi que des informations sur l'utilisation d'Apache Airflow v2.8.1 sur Amazon MWAA.

22 février 2024

- [Versions](#)
- [the section called “Packages de fournisseurs spécifiques à la version”](#)

[Support pour Amazon VPC partagé](#)

Amazon MWAA prend en charge la création d'environnements multi-comptes pour les organisations qui utilisent Amazon OpenSearch Service pour gérer les ressources Amazon MWAA à l'aide d'un Amazon VPC partagé centralisé dans un compte propriétaire. Dans le cadre de ce lancement, Amazon MWAA vous permet de choisir de créer et de gérer vos propres points de terminaison Amazon VPC.

15 novembre 2023

- [the section called “Gestion de vos propres points de terminaison Amazon VPC”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.7.2. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, ainsi que des informations sur l'utilisation d'Apache Airflow v2.7.2 sur Amazon MWAA.

6 novembre 2023

- [Versions](#)
- [the section called “Packages de fournisseurs spécifiques à la version”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.6.3. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour et des informations sur l'utilisation d'Apache Airflow v2.6.3 sur Amazon MWAA,

9 août 2023

- [Versions](#)

[Informations relatives à l'obsolescence des versions](#)

Rubrique mise à jour sur la dépréciation des versions afin d'inclure les notifications et les chronologies de dépréciation pour Apache Airflow v2.0.2 et Apache Airflow v2.2.2.

31 juillet 2023

- [the section called “Versions obsolètes d'Apache Airflow”](#)

[Nouveaux sujets et cas d'utilisation](#)

Amazon MWAA prend en charge les mises à niveau de versions mineures. Cette mise à jour inclut la nouvelle rubrique suivante qui décrit comment mettre à niveau l'environnement et s'assurer que les ressources de votre flux de travail sont compatibles avec la version d'Apache Airflow vers laquelle vous effectuez la mise à niveau :

5 juin 2023

- [the section called “Modification de la version”](#)

Rubrique mise à jour

Mise à jour des politiques IAM gérées par le client qui accordent à l'utilisateur un accès complet à la console et à l'API Amazon MWAA. La mise à jour explique pourquoi vous devez fournir une autorisation `iam:PassRole` pour autoriser un utilisateur à transmettre des rôles à Amazon MWAA. Amazon MWAA utilise ces autorisations pour effectuer des actions au nom d'un utilisateur.

12 avril 2023

- [the section called “Accès à un environnement Amazon MWAA”](#)

[Nouvelles directives](#)

Rubrique mise à jour

12 avril 2023

sur la configuration

AWS Secrets Manager en tant que backend pour Amazon MWAA afin de fournir des conseils sur l'utilisation des modèles de recherche.

L'utilisation de modèles de recherche permet de limiter les secrets recherchés par Apache Airflow et de réduire le nombre d'appels d'API qu'Amazon MWAA envoie à Secrets Manager pour récupérer des connexions et des variables. Cela réduit les coûts associés à l'utilisation de Secrets Manager en tant que backend.

- [Créez le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.5.1. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour et des informations sur l'utilisation d'Apache Airflow v2.5.1 sur Amazon MWAA,

11 avril 2023

- [Versions](#)

[Nouveaux sujets et cas d'utilisation](#)

Ajout d'une nouvelle rubrique sur l'utilisation d'un script de démarrage dans un environnement Amazon MWAA. Cette rubrique décrit la configuration d'un script de démarrage pour un environnement existant, son utilisation pour installer des environnements d'exécution Linux et la définition de variables d'environnement.

3 avril 2023

- [the section called “Utilisation d'un script de démarrage”](#)

[Section mise à jour sur l'accès aux serveurs Web privés](#)

Mise à jour de la rubrique suivante sur l'accès aux serveurs Web privés. La mise à jour précise que, dans les environnements avec accès privé à un serveur Web, vous devez utiliser une archive Python Wheel (.whl) pour emballer et installer les dépendances.

24 février 2023

- [Mode d'accès au serveur Web privé](#)

[Ajout d'informations sur les versions obsolètes d'Apache Airflow](#)

17 février 2023

La rubrique [Versions](#) a été mise à jour avec de nouvelles informations sur la façon dont Amazon MWAA a géré la désapprobation des versions d'Apache Airflow. Suppression d'une section concernant la mise à niveau vers une version plus récente d'Apache Airflow et d'une section décrivant les modifications entre Apache Airflow v1 et Apache Airflow v2. Pour plus d'informations sur la migration vers une nouvelle version d'Apache Airflow, consultez le guide de migration [Amazon MWAA](#).

- [the section called “Versions obsolètes d'Apache Airflow”](#)
- [the section called “Support des versions d'Apache Airflow et FAQ”](#)

[Correctifs relatifs aux
métriques des conteneurs
Amazon MWAA](#)

20 janvier 2023

Mise à jour de la rubrique relative aux métriques du conteneur et suppression d'un ensemble de métriques erronées qui n'existaient pas dans la `Cluster` dimension. Ajout d'une section supplémentaire qui décrit comment évaluer le nombre de travailleurs supplémentaires utilisés par un environnement à un moment donné en représentant graphiquement le `CPUUtilization` ou la `MemoryUtilization` métrique du `AdditionalWorker` composant et en définissant le type de `Sample Count` statistiques sur.

- [the section called “Évaluation du nombre de conteneurs de travailleurs et de serveurs Web supplémentaires”](#)

[Nouvelle version d'Apache Airflow](#)

Amazon MWAA prend désormais en charge Apache Airflow v2.4.3. Cette mise à jour inclut des informations sur les packages de fournisseurs mis à jour, des informations sur l'utilisation d'Apache Airflow v2.4.3 sur Amazon MWAA et des informations consolidées sur les fonctionnalités prises en charge dans chaque version d'Apache Airflow sur Amazon MWAA.

5 janvier 2023

- [Versions](#)

[Rubrique mise à jour sur le rôle lié à un service](#)

Informations mises à jour sur le rôle lié à un service utilisé par Amazon MWAA pour créer et gérer des AWS ressources en votre nom, notamment des informations sur la manière dont vous pouvez supprimer le rôle lié à un service lorsque vous n'en avez plus besoin. Cela inclut une politique d'autorisation de rôle liée à un service mise à jour qui permet à Amazon MWAA de publier des CloudWatch métriques supplémentaires dans l'espace de noms. AWS/MWAA

18 novembre 2022

- [the section called “Rôle lié à un service”](#)

[Nouveau sujet sur les indicateurs de service](#)

Ajout d'une nouvelle rubrique qui décrit les métriques de service émises par Amazon MWAA dans l'espace de AWS/MWAA noms. Il s'agit notamment des planificateurs, des travailleurs et des serveurs Web des métriques du cluster Amazon ECS, des métriques Amazon SQS pour les files d'attente qui permettent à Amazon MWAA de dissocier les planificateurs des travailleurs, ainsi que des métriques Amazon RDS pour la base de données de métadonnées.

18 novembre 2022

- [the section called “Mesures relatives aux conteneurs, aux files d'attente et aux bases de données”](#)

[Nouvelle rubrique](#)

Ajout de nouvelles instructions sur la modification d'un fichier de contraintes afin de spécifier les nouvelles versions des packages de fournisseurs à utiliser avec votre environnement Amazon MWAA.

18 novembre 2022

- [the section called “Fichier de contraintes”](#)

[Entrée de FAQ mise à jour](#)

Informations mises à jour relatives à l'éligibilité d'Amazon MWAA à la loi HIPAA.

15 novembre 2022

- [the section called “Conformité à la loi américaine HIPAA”](#)

[Nouvelle rubrique](#)

Ajout d'une nouvelle rubrique sur l'utilisation [aws:SourceArn](#) et les clés contextuelles de condition [aws:SourceAccount](#) globale dans une politique de confiance des rôles d'exécution Amazon MWAA, afin d'éviter toute confusion entre les services en tant qu'adjoint.

21 octobre 2022

- [the section called “Prévention du cas de figure de l'adjoint désorienté entre services”](#)

[Nouvel exemple de code](#)

Ajout d'instructions mises à jour et d'un exemple de code DAG qui écrit des métriques personnalisées au niveau du système d'exploitation dans CloudWatch

13 septembre 2022

- [the section called “Utiliser un DAG pour écrire des métriques personnalisées”](#)

Nouvel exemple de code	Ajout d'instructions mises à jour et d'un nouvel exemple de code AWS Lambda Python qui récupère un jeton de CLI Apache Airflow, puis invoque un DAG dans un environnement Amazon MWAA spécifié. • the section called “Invoquer DAGs avec Lambda”	12 septembre 2022
Nouveaux schémas architecturaux	Ajout de nouveaux diagrammes architecturaux illustrant un environnement Amazon MWAA avec un serveur Web public et privé. • the section called “Modes d'accès Apache Airflow”	12 septembre 2022
Nouvel exemple de code	Ajout d'instructions mises à jour et d'un nouvel exemple de code DAG qui récupère un jeton CLI Apache Airflow, puis invoque un autre DAG dans un autre environnement Amazon MWAA. • the section called “Invocation DAGs dans différents environnements”	16 août 2022

[Nouvel exemple de code](#)

Ajout d'instructions mises à jour et d'un nouveau DAG qui interroge Aurora PostgreSQL d'un environnement pour obtenir des informations de métadonnées, écrit le résultat dans des fichiers CSV et stocke les fichiers dans Amazon S3.

12 août 2022

- [the section called “Exportation des métadonnées de l'environnement vers Amazon S3”](#)

[Nouvel exemple de code](#)

Ajout d'instructions mises à jour et d'un nouveau DAG qui actualise un AWS CodeArtifact jeton lors de l'exécution et stocke le résultat dans Amazon S3.

3 août 2022

- [the section called “Actualisation d'un AWS CodeArtifact jeton lors de l'exécution”](#)

[Nouvel exemple de code](#)

Ajout d'instructions mises à jour et d'un exemple de code DAG pour l'utiliser ECSOperator dans Amazon MWAA.

26 juillet 2022

- [the section called “Utilisation de l'ECSOperator ”](#)

Nouvel exemple de code	Ajout d'instructions mises à jour et d'un exemple de code DAG pour l'utiliser SSHOperator dans Amazon MWAA. • the section called “Utilisation de l’SSHOperator ”	15 juillet 2022
Nouvel exemple de code	Ajout de nouvelles instructions et d'un exemple de code DAG pour l'utilisation de dbt Postgres avec Amazon MWAA. • the section called “Utiliser dbt avec Amazon MWAA”	17 juin 2022
Nouveaux sujets et cas d'utilisation	Ajout de nouvelles instructions et d'un exemple de code DAG pour installer des dépendances à l'aide de fichiers de roue Python pour les environnements Amazon MWAA avec accès public et privé. • Gestion des dépendances à l'aide de roues Python	13 mai 2022
Nouveaux sujets et cas d'utilisation	Ajout de nouvelles instructions sur le choix des métriques Apache Airflow auxquelles Amazon MWAA envoie. CloudWatch • Choix des métriques d'Apache Airflow à signaler	19 avril 2022

[Nouveaux guides](#)

Amazon MWAA propose un guide de migration pour la migration des flux de travail Apache Airflow à partir de déploiements autogérés, ainsi que d'environnements Amazon MWAA existants.

7 mars 2022

- [Guide de migration vers Amazon MWAA](#)

[Nouveaux sujets et cas d'utilisation](#)

Ajout de nouvelles bonnes pratiques de sécurité pour l'utilisation d'Apache Airflow, notamment une solution permettant de détecter les modifications apportées aux privilèges des utilisateurs d'Apache Airflow.

18 février 2022

- [the section called “Bonnes pratiques de sécurité dans Apache Airflow”](#)

[Nouvel exemple de code](#)

Ajout d'un nouvel exemple de code pour la création d'une fonction de fuseau horaire à DAGs l'aide de [Pendulum](#), et clarification de la manière d'utiliser un plugin personnalisé pour modifier le fuseau horaire dans lequel les journaux Apache Airflow sont créés.

11 février 2022

- [the section called “Modifier le fuseau horaire d'un DAG”](#)

[Lancement d'Apache Airflow v2.2.2](#)

27 janvier 2022

Amazon Managed Workflows pour Apache Airflow prend désormais en charge Apache Airflow v2.2.2. À partir de la version 2.2, Amazon MWAA installera des packages Python et des plug-ins personnalisés directement sur le serveur Web Apache Airflow, ce qui vous permettra de gérer vos environnements avec une plus grande flexibilité. Pour plus d'informations, reportez-vous à ce qui suit.

- [Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow](#).
- Journal des [modifications d'Apache Airflow v2.2.2](#) sur le site Web de documentation d'Apache Airflow.

[Nouveaux tutoriels](#)

Ajout d'un nouveau didacticiel illustrant la création d'un nouveau rôle Apache Airflow personnalisé et l'attribution de ce rôle à un utilisateur Apache Airflow mappé depuis IAM afin de limiter l'accès de l'utilisateur à un sous-ensemble spécifié. DAGs

8 décembre 2021

- [Tutoriel : Restreindre l'accès d'un utilisateur Amazon MWAA à un sous-ensemble de DAGs](#)

Correctifs

Correction d'une recommandation de bonnes pratiques pour définir la valeur de `scheduler.min_file_process_interval` afin d'optimiser l'utilisation du processeur. Ajout d'un exemple de politique IAM accordant l'accès aux ressources de Secrets Manager dans le rôle d'exécution. Ajout d'une rubrique de résolution des problèmes sur l'utilisation des clés de condition de Secrets Manager.

22 novembre 2021

- [Réglage des performances du mode d'analyse du planificateur DAGs](#)
- [Fournir à Amazon MWAA l'autorisation d'accéder aux clés secrètes de Secrets Manager](#)
- [Configuration des clés de condition dans le rôle d'exécution Amazon MWAA pour Secrets Manager](#)

Nouvel exemple de code

Ajout du nouvel exemple de code suivant pour modifier le fuseau horaire dans lequel les données DAGs sont traitées à l'aide d'un plugin personnalisé, et d'une nouvelle rubrique de résolution des problèmes pour invoquer la commande de la CLI `dags backfill` Apache Airflow depuis un opérateur `bash`.

1er novembre 2021

- [the section called “Modifier le fuseau horaire d'un DAG”](#)
- [Commande de remplissage de la CLI à l'aide d'un opérateur bash](#)

Correctifs

Correction de problèmes dans l'exemple de code d'opérateur Amazon ECS et clarification des autorisations supplémentaires requises dans le rôle d'exécution Amazon MWAA pour permettre à l'environnement d'accéder au groupe de CloudWatch journaux de tâches Amazon ECS dans Logs.

26 octobre 2021

- [Autorisations d'opérateur Amazon ECS.](#)

Nouvel exemple de code

Ajout d'un nouvel exemple de code qui interroge la base de données Aurora PostgreSQL pour obtenir des informations relatives aux exécutions DAG et écrit les résultats CSV dans un fichier stocké sur Amazon S3.

1er octobre 2021

- [the section called “Exportation des métadonnées de l'environnement vers Amazon S3”](#).

Correctifs

Informations corrigées sur la façon dont Amazon MWAA synchronise automatiquement les objets nouveaux et modifiés depuis votre compartiment Amazon S3 cible avec vos planificateurs et vos employés.

1er octobre 2021

- [Fonctionnement du dossier DAG](#).

Désormais pris en charge

Amazon MWAA prend désormais en charge des packages de fournisseurs supplémentaires pour Apache Airflow 2.0+. Pour en savoir plus sur les packages pris en charge, consultez ce qui suit :

24 septembre 2021

[Nouvelles commandes et procédures](#)

Ajout de conseils et d'exemples de AWS CLI commandes supplémentaires pour créer un point de terminaison de passerelle Amazon S3 lors de l'utilisation d'un Amazon VPC sans accès à Internet :

24 septembre 2021

- [Création d'un réseau Amazon VPC sans accès à Internet.](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

19 septembre 2021

- Ajout d'un nouvel exemple de code qui utilise un opérateur Amazon Elastic Container Service dans [the section called “Utilisation de l'ECSOperator”](#).
- Ajout de nouvelles rubriques de résolution des problèmes liés à la configuration des plugins Apache Airflow dans [the section called “Plug-ins”](#)

Nouvelle région prise en charge

Amazon MWAA est désormais disponible dans les régions suivantes :

31 août 2021

- Asie-Pacifique (Mumbai) – ap-south-1
- Asie-Pacifique (Séoul) – ap-northeast-2
- Europe (Londres) – eu-west-2
- Europe (Paris) – eu-west-3
- Canada (Centre) – ca-central-1
- Amérique du Sud (São Paulo) – sa-east-1

Pour plus d'informations sur la disponibilité des régions et les points de terminaison du service, consultez les sites suivants :

- [Points de terminaison et quotas Amazon MWAA](#) dans le. Références générales AWS

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

27 août 2021

- Les exemples de politiques ont été mis à jour pour permettre à Amazon MWAA de récupérer les paramètres `s3:GetAccountPublicAccessBlock` Amazon S3 () au niveau du compte dans. [Rôle d'exécution Amazon MWAA](#)

Correctifs

Les modifications suivantes ont été ajoutées :

27 août 2021

- Correction du CloudFormation modèle qui utilisait une règle d'autoréférencement entrant pour le groupe de sécurité dans. [Création du réseau VPC](#)
- Correction du CloudFormation modèle qui utilisait une règle d'autoréférencement entrant pour le groupe de sécurité dans. [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

20 août 2021

- Le décorateur DAG a été ajouté à la liste des fonctionnalités prises en charge par Apache [Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow](#) Airflow v2.0.2.

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

13 août 2021

- Cas `celery.py` `nc_parallelism` d'utilisation ajouté à [Optimisation des performances pour Apache Airflow sur Amazon MWAA](#).
- Ajout de points de terminaison de service à la page des quotas et changement de nom en [Amazon Managed Workflows pour les points de terminaison et quotas du service Apache Airflow](#).
- Prérequis de mise en réseau clarifiés sur la base des commentaires des utilisateurs sur. [Commencez avec Amazon Managed Workflows for Apache Airflow](#)
- Déplacé `dags list-runs` et `dags next-execution` vers des commandes Airflow CLI non prises en charge dans. [Référence des commandes de la CLI Apache Airflow](#)

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

13 août 2021

- Ajout d'un exemple de bash pour définir, obtenir ou supprimer une variable Apache Airflow v2.0.2 dans. [Référence des commandes de la CLI Apache Airflow](#)
- Ajout de dépendances d'Apache Airflow v2.0.2 et d'un exemple de connexion Airflow à. [Utilisation d'Amazon MWAA avec Amazon RDS pour Microsoft SQL Server](#)

[Correctifs](#)

Les modifications suivantes ont été ajoutées :

13 août 2021

- Correction de l'exemple de code Python basé sur les commentaires des utilisateurs sur [Création d'une connexion SSH à l'aide du SSHOperator](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

6 août 2021

- Déplacé `variables set` vers les commandes Airflow CLI prises en charge dans. [Référence des commandes de la CLI Apache Airflow](#)
- Le résumé des modifications apportées à la version 2.0.2 a été ajouté sur la page des versions d'Airflow en [Installation des dépendances Python](#) fonction des commentaires des utilisateurs.
- Le résumé des modifications apportées à la version 2.0.2 a été ajouté sur la page des versions d'Airflow en [Référence des commandes de la CLI Apache Airflow](#) fonction des commentaires des utilisateurs.
- Le résumé des modifications apportées à la version 2.0.2 a été ajouté sur la page des versions d'Airflow en [Vue d'ensemble des types de connexion](#) fonction des commentaires des utilisateurs.
- Le résumé des modifications apportées à la version 2.0.2 a été ajouté sur la page des versions d'Airflow

en [Installation de plugins personnalisés](#) fonction des commentaires des utilisateurs.

- Le résumé des modifications apportées à la version 2.0.2 a été ajouté sur la page des versions d'Airflow en [Ajouter ou mettre à jour DAGs](#) fonction des commentaires des utilisateurs.

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

6 août 2021

- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Utilisation d'un DAG pour importer des variables dans la CLI](#)
- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Invocation DAGs avec une fonction Lambda](#)

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

29 juillet 2021

- Ajout d'une rubrique de résolution des problèmes pour « Je ne trouve pas ma connexion dans l'interface utilisateur Airflow » à l'adresse. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)
- Ajout d'une liste des supports Amazon VPCs Amazon MWAA à [À propos de la mise en réseau sur Amazon MWAA](#).

Correctifs

Les modifications suivantes
ont été ajoutées :

29 juillet 2021

- Correction de l'exemple de code Python basé sur les commentaires des utilisateurs pour imprimer le jeton de connexion Web sur [Création d'un jeton d'accès au serveur Web Apache Airflow](#).
- Correction du problème de connexion Snowflake basé sur les commentaires des utilisateurs selon lequel il fallait utiliser un devis unique pour le paramètre d'entrepôt à. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)

Sujets supprimés ou déplacés

Les modifications suivantes ont été ajoutées :

23 juillet 2021

- Restructuration de la page existante pour y inclure toutes les pages de documentation sur la surveillance et les [Surveillance et métriques pour Amazon Managed Workflows pour Apache Airflow](#) métriques.
- Déplacé [Métriques de l'environnement Apache Airflow dans CloudWatch](#) vers le menu de navigation relatif à la surveillance et aux métriques.

Nouveaux guides

Les modifications suivantes ont été ajoutées :

23 juillet 2021

- Créé [Packages du fournisseur Apache Airflow installés sur les environnements Amazon MWAA](#).
- Créé [Vue d'ensemble de la surveillance sur Amazon MWAA](#).
- Créé [L'accès aux journaux d'audit se connecte AWS CloudTrail](#).
- Créé [Accès aux journaux Airflow sur Amazon CloudWatch](#).

Correctifs

Les modifications suivantes
ont été ajoutées :

23 juillet 2021

- Correction de l'exemple de code Python basé sur les commentaires des utilisateurs pour générer une chaîne de connexion Airflow dans le bon ordre et y a ajouté le paramètre de port. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- Ajout d'une étape pour installer un package de décompression localement en fonction des commentaires des utilisateurs dans [Création d'un plugin personnalisé avec Oracle](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

16 juillet 2021

- Ajout d'un sujet pour les opérateurs AWS DMS à [Questions fréquemment posées sur Amazon MWAA](#) l'adresse.
- Ajout d'une rubrique de résolution des problèmes pour une erreur de journalisation à distance sur [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Déplacé `variables set` vers des commandes Airflow CLI non prises en charge dans. [Référence des commandes de la CLI Apache Airflow](#)

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

9 juillet 2021

- Ajout d'étapes séquentielles pour créer un fichier requirements.txt en fonction des commentaires des utilisateurs sur. [Installation des dépendances Python](#)
- Ajout d'étapes séquentielles pour créer un fichier plugins.zip en fonction des commentaires des utilisateurs sur. [Installation de plugins personnalisés](#)
- Des liens de référence croisée ont été ajoutés dans le guide de l'utilisateur vers le guide de référence des API disponible dans le guide de référence des [API Amazon Managed Workflows for Apache Airflow](#).
- Ajout d'une rubrique expliquant pourquoi les plugins ne sont pas répertoriés dans le menu Airflow 2.0 Admin > Plugins à l'[Questions fréquemment posées sur Amazon MWAA](#)adresse.

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

9 juillet 2021

- Créé [Suppression de fichiers sur Amazon S3](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

2 juillet 2021

- Ajout d'une liste des valeurs prises en charge sur [Utilisation de clés gérées par le client pour le chiffrement](#).
- Mise à jour et clarification de l'exemple d'URL de dépôt privé en fonction des commentaires des utilisateurs dans [Gestion des dépendances Python dans requirements.txt](#).

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

2 juillet 2021

- Ajout d'un exemple de code Apache Airflow v1.10.12 permettant d'utiliser une clé privée AWS Secrets Manager pour une connexion SSH à. [Création d'une connexion SSH à l'aide du SSHOperator](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

25 juin 2021

- Ajouté StartedTaskInstances et FinishedTaskInstances métriques à [Métriques de l'environnement Apache Airflow dans CloudWatch](#).

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

25 juin 2021

- Ajout d'un exemple de code Apache Airflow v2.0.2 sur. [Utilisation d'Amazon MWAA avec Amazon EKS](#)

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

25 juin 2021

- Créé [Optimisation des performances pour Apache Airflow sur Amazon MWAA](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

18 juin 2021

- Ajout `connections` `add` et `connections delete` aux commandes CLI Apache Airflow v2.0.2 prises en charge à l'adresse. [Référence des commandes de la CLI Apache Airflow](#)
- Il a été ajouté que la dernière version disponible en CloudFormation est Apache Airflow v2.0.2 à l'adresse. [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#)
- Ajout d'une question pour le stockage de données temporaires sur les travaux d'Apache Airflow à. [Questions fréquemment posées sur Amazon MWAA](#)
- Ajout d'un sujet pour l'erreur « L'exécuteur signale que l'instance de tâche %s est terminée » à. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)
- Ajout d'une rubrique pour le journal « Le serveur a fermé la connexion de façon inattendue » à. [Résolution](#)

[des problèmes liés aux flux
de travail gérés par Amazon
pour Apache Airflow](#)

- Ajout d'un exemple pour exécuter des commandes CLI sur un tunnel SSH vers un hôte bastion pour. [Création d'un jeton CLI Apache Airflow](#)
- Ajout d'une rubrique pour les noms d'utilisateur générés de manière aléatoire pour. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)
- Ajout d'une rubrique pour une erreur 503 lors de l'exécution d'un DAG dans la CLI pour [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout d'une rubrique pour les plugins personnalisés dans Apache Airflow v2.0.2, qui nécessitent une option de configuration Airflow permettant de charger les plugins au début de chaque processus Airflow `core.lazy_load_plugins : False` afin de remplacer le paramètre par défaut de la version sur. [Utilisation des options](#)

[de configuration d'Apache Airflow sur Amazon MWAA](#)

- Ajout d'une étape d'options de configuration Airflow pour les plugins Apache Airflow v2.0.2, exemple de code sur. [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#)
- Ajout d'une étape d'options de configuration Airflow pour un exemple de code de plugins Apache Airflow v2.0.2.
- Ajout d'une étape d'options de configuration Airflow pour les plugins Apache Airflow v2.0.2, exemple de code sur. [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#)
- Ajout d'une étape d'options de configuration Airflow pour les plugins Apache Airflow v2.0.2, exemple de code sur. [Création d'un plugin personnalisé avec Oracle](#)

[Nouvel exemple de code](#)

Les modifications suivantes
ont été ajoutées :

18 juin 2021

- Ajout d'un exemple de code pour une connexion Apache Airflow Snowflake à l'adresse. [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow Snowflake](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

2 juin 2021

- Ajout de conseils de chiffrement côté serveur à. [Création d'un compartiment Amazon S3 pour Amazon MWAA](#)
- Le backend secret pour Apache Airflow v2.0.2 a été ajouté à. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- Ajout d'une question pour les demandes d'augmentation du quota des travailleurs d'Apache Airflow à. [Questions fréquemment posées sur Amazon MWAA](#)
- Ajout d'une question concernant les métriques utilisées pour déterminer s'il faut adapter les travailleurs d'Apache Airflow à. [Questions fréquemment posées sur Amazon MWAA](#)
- Ajout d'une question pour créer des métriques personnalisées dans CloudWatch [Questions fréquemment posées sur Amazon MWAA](#).
- Ajout d'étapes pour activer les adresses IP privées pour

un point de terminaison
d'interface Amazon S3 VPC
pour un VPC avec routage
privé. [Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé](#)

- Ajout d'une option pour configurer un tunnel SSH en utilisant la redirection de port locale dans [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#).

[Nouvel exemple de code](#)

Les modifications suivantes
ont été ajoutées :

2 juin 2021

- Ajout d'un exemple de code pour un DAG qui interroge la base de données de métadonnées Amazon Aurora PostgreSQL et publie des métriques personnalisées sur Amazon à l'adresse. CloudWatch [Utilisation d'un DAG pour écrire des métriques personnalisées dans CloudWatch](#)

Nouveaux guides

Les modifications suivantes
ont été ajoutées :

2 juin 2021

- Création d'un guide expliquant comment utiliser les modèles de connexion de manière interchangeable dans l'interface utilisateur d'Apache Airflow dans. [Vue d'ensemble des types de connexion](#)

Correctifs

Les modifications suivantes
ont été ajoutées :

2 juin 2021

- Ajout de points de terminais on VPC Apache Airflow au CloudFormation modèle dans la troisième option : création d'un réseau VPC sans accès Internet à. [Création du réseau VPC](#)

[Lancement d'Apache Airflow v2.0.2](#)

Lancement de la mise à disposition générale d'Apache Airflow v2.0.2.

26 mai 2021

- Créé [Versions Apache Airflow sur Amazon Managed Workflows for Apache Airflow](#).
- Créé [Métriques de l'environnement Apache Airflow dans CloudWatch](#).
- Ajout de liens spécifiques à la version pour Apache Airflow v2.0.2 vers. [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#)
- Ajout de conseils spécifiques à la version 2.0.2 d'Apache Airflow à. [Installation des dépendances Python](#)
- Ajout de conseils spécifiques à la version 2.0.2 d'Apache Airflow à. [Gestion des dépendances Python dans requirements.txt](#)
- Ajout d'exemples de plugins Apache Airflow v2.0.2 à. [Installation de plugins personnalisés](#)
- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Nettoyage de base de données Aurora PostgreSQ](#)

[L dans un environnement Amazon MWAA](#)

- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow](#)
- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#)
- Les commandes Apache Airflow v2.0.2 ont été ajoutées à. [Référence des commandes de la CLI Apache Airflow](#)
- Ajout de scripts Apache Airflow v2.0.2 à. [Création d'un jeton CLI Apache Airflow](#)
- Ajout d'une note indiquant qu'Amazon MWAA utilise par défaut la dernière version d'Apache Airflow pour. [Création d'un environnement Amazon MWAA](#)

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

14 mai 2021

- Ajout de conseils pour le dépannage des tâches Airflow bloquées ou non exécutées. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)

Correctifs

Les modifications suivantes ont été ajoutées :

12 mai 2021

- Nous avons mis à jour l'exemple de code des plug-ins pour utiliser la dernière version de Java dans [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#). Auparavant, c'était le cas de `os.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"`.

Sujets supprimés ou déplacés

Les modifications suivantes ont été ajoutées :

10 mai 2021

- Sujets déplacés [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#) vers de nouvelles pages par catégorie.

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

10 mai 2021

- Présentation du compartiment Amazon S3 ajoutée à [Travailler avec DAGs sur Amazon MWAA](#).

Sujets supprimés ou déplacés

Les modifications suivantes ont été ajoutées :

7 mai 2021

- Déplacé [Accès à Apache Airflow](#) vers la navigation de niveau principal et ajout de pages pour [Création d'un jeton d'accès au serveur Web Apache Airflow](#)[Création d'un jeton CLI Apache Airflow](#), et. [Référence des commandes de la CLI Apache Airflow](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

7 mai 2021

- Ajout de liens spécifiques à la version vers le guide de référence d'Apache Airflow pour toutes les commandes Airflow CLI prises en charge et non prises en charge dans. [Référence des commandes de la CLI Apache Airflow](#)
- Ajout de liens spécifiques à la version vers le guide de référence d'Apache Airflow pour toutes les options de configuration dans. [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#)
- L'utilitaire Amazon MWAA CLI a été ajouté à. [Gestion des dépendances Python dans requirements.txt](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

30 avril 2021

- Ajout d'exemples plats et imbriqués expliquant comment structurer un fichier plugins.zip dans [Installation de plugins personnalisés](#).
- L'utilitaire Amazon MAAA CLI a été ajouté aux [Installation des dépendances Python](#) pages [Ajouter ou mettre à jour DAGs](#) [Installation de plugins personnalisés](#), et.
- Contenu restructuré en une vue d'ensemble, téléchargement sur Amazon S3 et installation sur Amazon MWAA sections en fonction des commentaires des utilisateurs dans et [Installation des dépendances Python](#) sur les [Installation de plugins personnalisés](#) pages.
- Ajout d'un exemple de cas d'utilisation pour créer et associer les points de terminaison VPC requis à un Amazon VPC existant sans accès à Internet. [À propos de la mise en réseau sur Amazon MWAA](#)

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

30 avril 2021

- Ajout d'un exemple de code qui utilise une clé secrète dans Secrets Manager pour une variable Apache Airflow dans. [Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow](#)

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

30 avril 2021

- Créé [Création des points de terminaison de service VPC requis dans un Amazon VPC avec routage privé.](#)

[Correctifs](#)

Les modifications suivantes ont été ajoutées :

30 avril 2021

- Nous l'avons mis `core.default_ui_timezone` à jour webserver `.default_ui_timezone` en [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA.](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

23 avril 2021

- Ajout d'étapes Windows (PuTTY) pour le tunnel SSH vers. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#)
- Ajout d'une rubrique pour `apach-airflow-providers-amazon`, qui est uniquement compatible avec Apache Airflow 2.0 pour. [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

23 avril 2021

- Ajout d'un exemple de code qui utilise une clé secrète dans Secrets Manager pour une connexion Apache Airflow dans. [Utilisation d'une clé secrète AWS Secrets Manager pour une connexion Apache Airflow](#)

[Nouveaux guides](#)

Les modifications suivantes
ont été ajoutées :

23 avril 2021

- Créé [À propos de la mise en réseau sur Amazon MWAA](#).
- Créé [Sécurité de votre VPC sur Amazon MWAA](#).
- Créé [Gestion de l'accès aux points de terminaison Amazon VPC spécifiques à un service sur Amazon MWAA](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

16 avril 2021

- Ajout d'un nouveau CloudFormation modèle pour créer un réseau Amazon VPC sans accès à Internet. [Création du réseau VPC](#)
- Ajout d'un nouveau didacticiel pour créer un AWS Client VPN in [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un AWS Client VPN](#).
- Le nom de la page d'accès au réseau a été remplacé par les modes d'accès Apache Airflow en fonction des commentaires des utilisateurs, et la documentation a été rationalisée dans. [Modes d'accès Apache Airflow](#)
- Documents rationalisés pour inclure uniquement les informations de démarrage d'Amazon VPC et les modèles basés sur les commentaires des utilisateurs dans. [Création du réseau VPC](#)
- Ajout d'une solution BigQuery de contournement pour les opérateurs à. [Résolution des problèmes](#)

[liés aux flux de travail gérés
par Amazon pour Apache
Airflow](#)

- Ajout d'un fichier de contraintes Apache Airflow v1.10.12 (meilleures pratiques) à. [Installation des dépendances Python](#)

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

16 avril 2021

- Ajout d'un exemple de code pour créer un plugin personnalisé à l'aide d'Oracle in [Création d'un plugin personnalisé avec Oracle](#).
- Ajout d'un exemple de code pour créer un plugin personnalisé qui génère des variables d'environnement d'exécution.
-

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

9 avril 2021

- Ajout d'une rubrique concernant l'exigence de règle d'autoréférencement sur un groupe de sécurité VPC à. [Questions fréquemment posées sur Amazon MWAA](#)
- Ajout d'un répertoire de plugins personnalisés et de limites de taille à [Installation de plugins personnalisés](#).
- Ajout du répertoire des exigences et des limites de taille à [Installation des dépendances Python](#).
- Clarification des options de configuration d'Apache Airflow pour `foo.user` et `foo.pass` dans. [Gestion des dépendances Python dans requirements.txt](#)
- Présentation des options de configuration ajoutée à [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

9 avril 2021

- Ajout d'un exemple de code pour créer un plugin personnalisé PythonVirtualenvOperator à l'aide de [Création d'un plugin personnalisé pour Apache Airflow PythonVirtualenvOperator](#).
- Ajout d'un exemple de code pour créer un plugin personnalisé avec Apache Hive et Hadoop. [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#)

[Correctifs](#)

Les modifications suivantes ont été ajoutées :

31 mars 2021

- Nous avons mis à jour le format d'un fichier requirements.txt et ajouté un exemple compatible avec Apache Airflow v1.10.12 dans. [Installation des dépendances Python](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

26 mars 2021

- Ajout d'une solution de contournement pour supprimer un fichier requirements.txt ou plugins.zip dans [Questions fréquemment posées sur Amazon MWAA](#).
- Ajout d'une solution de contournement bash pour SSH dans un environnement à. [Questions fréquemment posées sur Amazon MWAA](#)
- Ajout d'un sujet d' CloudTrail ResourceAlreadyExistsException erreur à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

19 mars 2021

- Ajout de la liste des AWS services utilisés pour [Rôle d'exécution Amazon MWAA](#).
- Ajout de la liste des AWS services utilisés pour [Rôle lié à un service pour Amazon MWAA](#).
- Une question a été ajoutée pour la version Python 3.7 pour Amazon MWAA à [Questions fréquemment posées sur Amazon MWAA](#).
- Ajout d'une question PythonVirtualenvOperator pour [Questions fréquemment posées sur Amazon MWAA](#).
- Le script de dépannage a été ajouté comme prochaines étapes pour toutes les rubriques relatives au VPC et à la configuration de l'environnement à l'adresse [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Clarification de la documentation selon laquelle un bastion Linux doit se trouver dans la même région qu'un

environnement. [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#)

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

19 mars 2021

- Création du guide des connexions Apache Airflow pour AWS Secrets Manager at. [Configuration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager](#)
- J'ai créé un didacticiel de démarrage rapide à l'aide d'un CloudFormation modèle pour créer l'infrastructure Amazon VPC, le compartiment Amazon S3 et l'environnement Amazon MWAA sur. [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#)

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

12 mars 2021

- Ajout de la rubrique relative à la résolution des problèmes liés à la création d'un compartiment Amazon S3 [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout d'étapes pour créer et associer une politique JSON à [Rôle d'exécution Amazon MWAA](#).

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

12 mars 2021

- Ajout d'un exemple de code pour ajouter une configuration lors du déclenchement d'un DAG sur [Accès à Apache Airflow](#).

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

12 mars 2021

- A créé un guide des meilleures pratiques à l'adresse [Gestion des dépendances Python dans requirements.txt](#).

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

5 mars 2021

- Ajout d'une rubrique de Google/GCP/BigQuery résolution des problèmes à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout de la rubrique de résolution des problèmes liés à Cython à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)
- Ajout d'une rubrique de résolution des problèmes MySQL à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout de la rubrique de résolution des erreurs du serveur Web 5xx à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#)

Désormais pris en charge

Les modifications suivantes ont été ajoutées :

4 mars 2021

- Auparavant, `backend_kwargs` il n'était pas pris en charge pour AWS Secrets Manager et vous deviez trouver une solution pour annuler l'appel de fonction Secrets Manager. Maintenant, `backend_kwargs` est pris en charge. Reportez-vous à la rubrique AWS Secrets Manager relative à la résolution des problèmes dans [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).

Correctifs

Les modifications suivantes ont été ajoutées :

4 mars 2021

- Nous avons mis à jour la taille de chaque classe d'environnement pour refléter le nombre réel de Go en Go [Configuration de la classe d'environnement Amazon MWAA](#).

[Nouveaux sujets et cas d'utilisation](#)

Les modifications suivantes ont été ajoutées :

26 février 2021

- Accès au réseau privé ajouté à l'aide d'une politique de point de terminaison VPC pour. [Modes d'accès Apache Airflow](#)
- Des vérifications supplémentaires ont été ajoutées pour la rubrique de résolution des problèmes liés à la création d'un environnement à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout d'étapes pour accéder aux journaux `requirements.txt` de [Installation des dépendances Python](#).

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

25 février 2021

- Ajout du cas d'utilisation d'Apache Hive à [Installation des dépendances Python](#)
- Clarification de la documentation selon laquelle les dépendances requises pour un package Apache Airflow doivent être incluses dans le `requirements.txt` fichier à l'adresse [Installation des dépendances Python](#)
- Ajout de la rubrique de résolution des problèmes de mise à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#) jour de `requirements.txt` à

Nouveaux tutoriels

Les modifications suivantes ont été ajoutées :

22 février 2021

- Ajout d'un didacticiel sur le réseau privé à [Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion](#).

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

22 février 2021

- Des configurations de réseau privé et public ont été ajoutées à [Modes d'accès Apache Airflow](#).
- Ajout de cas d'utilisation de groupes de développement et de scénarios utilisateur à [Rôle d'exécution Amazon MWAA](#).

Nouvel exemple de code

Les modifications suivantes ont été ajoutées :

22 février 2021

- Ajout d'exemples de scripts Python pour le jeton de connexion Web et le jeton CLI à [Accès à Apache Airflow](#).
- Ajout d'un exemple de code pour déclencher le DAG dans un autre environnement pour [Exemples de code pour Amazon Managed Workflows pour Apache Airflow](#).
- Ajout d'un exemple de code pour déclencher le DAG à l'aide d'une fonction Lambda pour. [Invocation DAGs avec une fonction Lambda](#)

[Nouvelles commandes et procédures](#)

Les modifications suivantes ont été ajoutées :

22 février 2021

- Des procédures étape par étape ont été ajoutées à tous les scripts sur [Accès à Apache Airflow](#).

[Nouvel exemple de code](#)

Les modifications suivantes ont été ajoutées :

17 février 2021

- Exemple de curl mis à jour pour le jeton de connexion Web à [Accès à Apache Airflow](#) l'adresse.
- Ajout d'un exemple de code pour se connecter à un Amazon RDS Microsoft SQL Server à [Utilisation d'Amazon MWAA avec Amazon RDS pour Microsoft SQL Server](#).

[Nouvelles commandes et procédures](#)

Les modifications suivantes ont été ajoutées :

17 février 2021

- AWS CLI Commandes ajoutées aux [Travailler avec DAGs sur Amazon MWAA](#) pages.
- Apache Airflow ne prend pas en charge la sérialisation dans les commandes DAGs CLI. Étant donné que la CLI s'exécute sur le serveur Web, qui n'a pas de plugins ni d'exigences pour des raisons de sécurité, aucun environnement MWAA doté d'un fichier plugins.zip ou requirements.txt ne prendra pas en charge ces commandes. Apache Airflow `list_dags` et ses commandes ont été déplacés vers des `backfill` commandes non prises en charge sur. [Accès à Apache Airflow](#)

[GitHub lancement](#)

Les documents du guide de l'utilisateur sont désormais open source sur GitHub. Choisissez Modifier cette page GitHub sur n'importe quelle page.

17 février 2021

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

12 février 2021

- Une question a été ajoutée pour le cas d'utilisation de Step Functions v. Amazon MWAA pour [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Politique d'accès CLI ajoutée à [Accès à un environnement Amazon MWAA](#).
- Clarification de la documentation dans laquelle toute option de configuration Apache Airflow prise en charge peut être spécifiée . [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#)
- Clarification de la documentation selon laquelle en cas de défaillance d'un conteneur Fargate dans une zone de disponibilité, MWAA passe à l'autre conteneur dans une autre zone de disponibilité à. [Création du réseau VPC](#)

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

5 février 2021

- Ajouté [Configuration de la classe d'environnement Amazon MWAA](#).

Sujets supprimés ou déplacés

Les modifications suivantes ont été ajoutées :

4 février 2021

- Suppression de l'obligation pour le nom du compartiment Amazon S3 de commencer par airflow- at [Commencez avec Amazon Managed Workflows for Apache Airflow](#).
- Déplacé [Accès à un environnement Amazon MWAA](#) et [Rôle d'exécution Amazon MWAA](#) à [Gestion de l'accès à un environnement Amazon MWAA](#).

[Amazon MWA CloudFormation](#)

Mettez à jour les paramètres pour créer un environnement sur [Amazon MWAA CloudFormation](#).

4 février 2021

- Supprimer SubnetList.
- Supprimer TagList.
- Ajoutez NetworkConfiguration.
- Ajoutez TagMap.
- Ajoutez des exemples de demandes de création d'environnement.

Nouveaux sujets et cas d'utilisation

Les modifications suivantes ont été ajoutées :

29 janvier 2021

- Un exemple de configuration d'e-mail a été ajouté à [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).
- Ajout d'une rubrique de PostgresHook résolution des problèmes à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout d'une rubrique de AWS Secrets Manager résolution des problèmes à [Résolution des problèmes liés aux flux de travail gérés par Amazon pour Apache Airflow](#).
- Ajout d'un cas d'utilisation haute performance à [Configuration du dimensionnement automatique d'Amazon MWAA Worker](#).

Lancement d'Amazon MWAA

Lancement de la disponibilité générale d'Amazon Managed Workflows pour Apache Airflow.

24 novembre 2020

- Documentation du guide de l'utilisateur
- CloudFormation documentation

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.