



Guide de l'utilisateur

Amazon Managed Workflows for Apache Airflow



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

Copyright © 2024 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
Fonctionnalités	1
Architecture	2
Intégration	4
Versions prises en charge	4
Quelle est la prochaine étape ?	4
Démarrage rapide	5
Dans ce tutoriel	5
Prérequis	6
Étape 1 : enregistrer le AWS CloudFormation modèle localement	7
Deuxième étape : créer la pile à l'aide du AWS CLI	17
Troisième étape : télécharger un DAG sur Amazon S3 et l'exécuter dans l'interface utilisateur d'Apache Airflow	17
Quatrième étape : Afficher les journaux dans CloudWatch Logs	18
Quelle est la prochaine étape ?	19
Mise en route	20
Prérequis	20
A propos de ce manuel	20
Avant de commencer	21
Régions disponibles	22
Créer un compartiment	22
Avant de commencer	23
Créez le compartiment.	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 ?	40
Création d'un environnement	41
Avant de commencer	41
Versions d'Apache Airflow	41
Création d'un environnement	43
Quelle est la prochaine étape ?	25

Gestion des accès	48
Accès à un environnement Amazon MWAA	48
Comment ça marche	49
Accès complet à la console	51
Accès complet à l'API	57
Accès à la console en lecture seule	61
Accès à l'interface utilisateur 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
Afficher et mettre à jour une politique de rôle d'exécution	76
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 à Apache Airflow	86
Modes d'accès à Apache Airflow	86
Vue d'ensemble des modes d'accès	88
Configuration pour les modes d'accès privé et public	89
Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)	91
Accès à Apache Airflow	92
Prérequis	92

Accès	92
AWS CLI	93
Ouvrez l'interface utilisateur d'Apache Airflow	93
Connexion à Apache Airflow	93
Création d'un jeton d'accès au serveur Web	93
Prérequis	94
En utilisant le AWS CLI	95
Utiliser un script bash	95
Utilisation d'une requête d'API POST	96
Utilisation d'un script Python	97
Quelle est la prochaine étape ?	97
Configuration d'un domaine personnalisé	98
Configuration du domaine personnalisé	98
Configuration de l'infrastructure réseau	99
Jeton CLI Apache Airflow	104
Prérequis	105
Utilisation du AWS CLI	105
Utilisation d'un script curl	105
Utiliser un script bash	108
Utiliser un script Python	109
Quelle est la prochaine étape ?	112
Utilisation de l'API REST Apache Airflow	112
Création d'un jeton de session de serveur Web	113
Appellez l'API REST Apache Airflow	115
Référence des commandes de la CLI Apache Airflow	116
Prérequis	117
Ce qui a changé dans la version 2	118
Commandes CLI prises en charge	118
Exemple de code	122
Gestion des connexions	125
Présentation	125
Paquets Apache Airflow	126
Packages de fournisseurs pour les connexions Apache Airflow v2.9.2	126
Packages de fournisseurs pour les connexions Apache Airflow v2.8.1	127
Packages de fournisseurs pour les connexions Apache Airflow v2.7.2	128
Packages de fournisseurs pour les connexions Apache Airflow v2.6.3	129

Packages de fournisseurs pour les connexions Apache Airflow v2.5.1	130
Packages de fournisseurs pour les connexions Apache Airflow v2.4.3	131
Packages de fournisseurs pour les connexions Apache Airflow v2.2.2	131
Packages de fournisseurs pour les connexions Apache Airflow v2.0.2	132
Spécifier les nouveaux packages de fournisseurs	133
Types de connexion	133
Exemple de chaîne d'URI de connexion	134
Exemple de modèle de connexion de connexion	134
Exemple d'utilisation d'un modèle de connexion HTTP pour une connexion Jdbc	136
Configuration de Secrets Manager	138
Étape 1 : donnez à Amazon MWAA l'autorisation d'accéder aux clés secrètes de Secrets Manager	139
Deuxième étape : créer le backend Secrets Manager en tant qu'option de configuration d'Apache Airflow	140
Troisième étape : générer une chaîne d'URI de AWS connexion Apache Airflow	141
Étape 4 : ajouter les variables dans Secrets Manager	144
Étape 5 : ajouter la connexion dans Secrets Manager	145
Exemple de code	147
Ressources	147
Quelle est la prochaine étape ?	147
Gestion des environnements	148
Configuration de la classe d'environnement	148
Capacités environnementales	149
Planificateurs Apache Airflow	150
Configuration de la mise à l'échelle automatique des travailleurs	151
Comment fonctionne Worker Scaling	151
Utilisation de la console Amazon MWAA	152
Exemple de cas d'utilisation à hautes performances	152
Tâches de dépannage bloquées en cours d'exécution	154
Quelle est la prochaine étape ?	154
Configuration de la mise à l'échelle automatique du serveur Web	155
Comment fonctionne le dimensionnement des serveurs Web	155
Utilisation de la console Amazon MWAA	155
Utilisation des options de configuration	156
Prérequis	157
Comment ça marche	157

Utilisation des options de configuration pour charger des plugins dans Apache Airflow v2 ...	158
Vue d'ensemble des options de configuration	158
Référence de configuration	160
Exemples et exemple de code	166
Quelle est la prochaine étape ?	168
Mise à niveau de la version	168
Améliorez les ressources de votre flux de travail	169
Spécifiez la nouvelle version	170
Utilisation d'un script de démarrage	171
Configuration d'un script de démarrage	172
Installation des environnements d'exécution Linux	176
Définir les variables d'environnement	177
Travailler avec les DAG	181
Présentation du bucket Amazon S3	181
Ajout ou mise à jour des DAG	182
Prérequis	182
Comment ça marche	183
Ce qui a changé dans la version 2	184
Tester les DAG à l'aide de l'utilitaire Amazon MWAA CLI	184
Chargement du code DAG dans Amazon S3	184
Spécification du chemin d'accès à un dossier DAG	186
Affichage des modifications sur votre interface utilisateur Apache Airflow	186
Quelle est la prochaine étape ?	186
Installation de plugins personnalisés	187
Prérequis	187
Comment ça marche	188
Ce qui a changé dans la version 2	188
Vue d'ensemble des plugins personnalisés	189
Exemples de plugins personnalisés	190
Création d'un fichier plugins.zip	199
Téléchargement plugins.zip vers Amazon S3	200
Installation de plugins personnalisés sur votre environnement	202
Exemples de cas d'utilisation pour plugins.zip	203
Quelle est la prochaine étape ?	203
Installation des dépendances Python	203
Prérequis	204

Comment ça marche	205
Vue d'ensemble des dépendances Python	205
Création d'un fichier requirements.txt	206
Téléchargement requirements.txt vers Amazon S3	209
Installation de dépendances Python dans votre environnement	211
Afficher les journaux de votre requirements.txt	212
Quelle est la prochaine étape ?	213
Suppression de fichiers sur Amazon S3	213
Prérequis	213
Vue d'ensemble du versionnement	214
Comment ça marche	214
Supprimer un DAG sur Amazon S3	215
Suppression du fichier « actuel » plugins.zip ou requirements.txt	215
Supprimer le fichier « non actuel » plugins.zip ou requirements.txt	215
Supprimer des fichiers avec des cycles de vie	216
Exemple de politique de cycle de vie	216
Quelle est la prochaine étape ?	217
Réseaux	218
À propos du réseau	218
Conditions	219
Ce qui est pris en charge	219
Présentation de l'infrastructure VPC	219
Exemples de cas d'utilisation pour un Amazon VPC et un mode d'accès Apache Airflow	223
Sécurité dans votre VPC	225
Conditions	226
Aperçu de la sécurité	226
Listes de contrôle d'accès réseau (listes ACL)	226
Groupes de sécurité VPC	227
Politiques de point de terminaison VPC (routage privé uniquement)	229
Gestion de l'accès aux points de terminaison VPC	231
Tarification	231
Vue d'ensemble des points de terminaison VPC	232
Autorisation d'utiliser d'autres AWS services	232
Affichage des points de terminaison VPC	233
Accès au point de terminaison VPC de votre serveur Web Apache Airflow (accès réseau privé)	235

Points de terminaison de service VPC dans des VPC Amazon privés	236
Tarification	237
Réseau privé et routage privé	237
Points de terminaison VPC (obligatoires)	238
Connexion des points de terminaison VPC requis	239
(Facultatif) Activez les adresses IP privées pour le point de terminaison de votre interface Amazon S3 VPC	243
Gestion de vos propres points de terminaison Amazon VPC	244
Création d'un environnement dans un Amazon VPC partagé	244
Didacticiels	255
Didacticiel : AWS Client VPN	255
Réseau privé	256
Cas d'utilisation	257
Avant de commencer	257
Objectifs	257
(Facultatif) Première étape : identification de votre VPC, de vos règles CIDR et de vos sécurités VPC	258
Étape 2 : créer les certificats de serveur et de client	259
Troisième étape : enregistrer leAWS CloudFormation modèle localement	260
Quatrième étape : créer laAWS CloudFormation pile Client VPN	262
Étape 5 : associer des sous-réseaux à votre Client VPN	262
Sixième étape : ajouter une règle d'entrée d'autorisation à votre Client VPN	263
Étape 7 : Télécharger le fichier de configuration du point de terminaison Client VPN	264
Huitième étape : Connect auAWS Client VPN	265
Quelle est la prochaine étape ?	266
Tutoriel : Linux Bastion Host	266
Réseau privé	267
Cas d'utilisation	268
Avant de commencer	268
Objectifs	268
Première étape : créer l'instance de bastion	269
Deuxième étape : créer le tunnel SSH	270
Troisième étape : configurer le groupe de sécurité Bastion en tant que règle entrante	271
Quatrième étape : copier l'URL d'Apache Airflow	272
Étape 5 : configurer les paramètres du proxy	272
Sixième étape : ouvrir l'interface utilisateur d'Apache Airflow	275

Quelle est la prochaine étape ?	275
Tutoriel : Restreindre les utilisateurs à un sous-ensemble de DAG	276
Prérequis	276
Étape 1 : fournissez un accès au serveur Web Amazon MWAA à votre principal IAM avec le rôle Public Apache Airflow par défaut.	277
Deuxième étape : créer un nouveau rôle personnalisé pour Apache Airflow	278
Troisième étape : attribuer le rôle que vous avez créé à votre utilisateur Amazon MWAA	279
Étapes suivantes	280
Ressources connexes	280
Tutoriel : Automatisez la gestion des points de terminaison de votre propre environnement	280
Prérequis	281
Création de l'Amazon VPC	282
Créer la fonction Lambda	282
Créez la EventBridge règle	283
Création de l'environnement	284
Exemples de code	286
Importer des variables DAG	287
Version	287
Prérequis	287
Autorisations	287
Dépendances	287
Exemple de code	288
Quelle est la prochaine étape ?	289
Utilisation de l'SSHOperator	289
Version	290
Prérequis	290
Autorisations	291
Prérequis	291
Copiez votre clé secrète sur Amazon S3	291
Création d'une nouvelle connexion Apache Airflow	291
Exemple de code	292
Connexion à Apache Airflow Snowflake dans Secrets Manager	294
Version	294
Prérequis	294
Autorisations	295
Prérequis	295

Exemple de code	295
Quelle est la prochaine étape ?	296
Utilisation d'un DAG pour écrire des métriques personnalisées	296
Version	297
Prérequis	297
Autorisations	297
Dépendances	297
Exemple de code	297
Nettoyage de base de données Aurora PostgreSQL	301
Version	301
Prérequis	301
Dépendances	301
Exemple de code	301
Exportation des métadonnées de l'environnement vers Amazon S3	304
Version	304
Prérequis	304
Autorisations	305
Prérequis	305
Exemple de code	305
Utilisation d'une variable Apache Airflow dans Secrets Manager	308
Version	308
Prérequis	308
Autorisations	309
Prérequis	309
Exemple de code	309
Quelle est la prochaine étape ?	310
Utilisation d'une connexion Apache Airflow dans Secrets Manager	310
Version	311
Prérequis	311
Autorisations	311
Prérequis	309
Exemple de code	312
Quelle est la prochaine étape ?	315
Plug-in personnalisé avec Oracle	315
Version	315
Prérequis	316

Autorisations	316
Prérequis	316
Exemple de code	316
Créez le plugin personnalisé	317
Options de configuration du débit d'air	321
Quelle est la prochaine étape ?	321
Plugin personnalisé avec variables d'environnement	321
Version	322
Prérequis	322
Autorisations	322
Prérequis	322
Plug-in personnalisé	322
Plugins.zip	323
Options de configuration du débit d'air	323
Quelle est la prochaine étape ?	323
Changer le fuseau horaire d'un DAG	324
Version	324
Prérequis	324
Autorisations	324
Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow	325
Création d'un plugins.zip	325
Exemple de code	326
Quelle est la prochaine étape ?	327
RafraîchissantAWS CodeArtifactjeton au moment de l'exécution	327
Version	328
Prérequis	328
Autorisations	328
Exemple de code	329
Quelle est la prochaine étape ?	330
Plugin personnalisé avec Apache Hive et Hadoop	330
Version	331
Prérequis	331
Autorisations	331
Prérequis	309
Dépendances de téléchargement	332
Plugin personnalisé	333

Plugins.zip	334
Exemple de code	334
Options de configuration du débit d'air	335
Quelle est la prochaine étape ?	335
Plugin personnalisé à patcherPythonVirtualenvOperator	335
Version	336
Prérequis	336
Autorisations	336
Prérequis	336
Exemple de code de plugin personnalisé	336
Plugins.zip	338
Exemple de code	338
Options de configuration du débit d'air	341
Quelle est la prochaine étape ?	341
Invoquer des DAG avec Lambda	341
Version	342
Prérequis	342
Autorisations	342
Dépendances	343
Exemple de code	343
Invoquer des DAG dans différents environnements	344
Version	345
Prérequis	345
Autorisations	345
Dépendances	345
Exemple de code	346
Serveur Amazon RDS	348
Version	348
Prérequis	348
Dépendances	301
Connexion Apache Airflow v2	349
Exemple de code	350
Quelle est la prochaine étape ?	352
Intégration avec Amazon EMR	352
Version	353
Exemple de code	353

Amazon EKS (extrait)	355
Version	356
Prérequis	356
Création d'une clé publique pour Amazon EC2	357
Création du cluster	357
Créez un namespace de noms	358
Créez un rôle pour un namespace de noms	358
Création et attachement d'un rôle IAM pour le cluster Amazon EKS	359
Créez le fichier requirements.txt	363
Création d'un mappage d'identité pour Amazon EKS	363
Créer le kubeconfig	363
Création d'un DAG	364
Ajoutez le DAG et kube_config.yaml vers le compartiment Amazon S3	366
Activer et déclencher l'exemple	366
Utilisation du ECSOperator	367
Version	367
Prérequis	367
Autorisations	368
Création d'un cluster Amazon ECS	369
Exemple de code	374
Utiliser dbt avec Amazon MWAA	377
Version	377
Prérequis	378
Dépendances	378
Importer un projet dbt sur Amazon S3	379
Utiliser un DAG pour vérifier l'installation de la dépendance dbt	380
Utiliser un DAG pour exécuter un projet dbt	381
AWS blogs et tutoriels	381
Bonnes pratiques	382
Optimisation des performances pour Apache Airflow	382
Ajout d'une option de configuration Apache Airflow	383
Planificateur Apache Airflow	383
Dossiers DAG	388
fichiers DAG	390
Tâches	395
Gestion des dépendances en Python	400

Test des DAG à l'aide de l'utilitaire Amazon MAWA 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	408
Afficher les journaux sur la console CloudWatch Logs	409
Affichage des erreurs dans l'interface utilisateur d'Apache Airflow	410
Exemples de requirements.txt scénarios	411
Surveillance et mesures	412
Présentation	412
CloudWatch Présentation d'Amazon	413
AWS CloudTrail vue d'ensemble	413
Consultation des journaux d'audit	413
Création d'un parcours dans CloudTrail	414
Afficher les événements avec l'historique des CloudTrail événements	414
Exemple de parcours pour CreateEnvironment	414
Quelle est la prochaine étape ?	416
Affichage des journaux de flux d'air	416
Tarification	417
Avant de commencer	417
Types de journaux	417
Activation des journaux Apache Airflow	418
Afficher les 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	427
Quelle est la prochaine étape ?	427
Métriques de l'environnement Apache Airflow v2	427
Conditions	428
Dimensions	429
Accès aux métriques dans la CloudWatch console	430
Les métriques Apache Airflow sont disponibles dans CloudWatch	430
Choix des indicateurs à signaler	447
Quelle est la prochaine étape ?	447

Mesures relatives aux conteneurs, aux files d'attente et aux bases de données	448
Conditions	449
Dimensions	449
Accès aux métriques	450
Liste des métriques	450
Sécurité	454
Protection des données	455
Chiffrement	456
Utilisation de clés gérées par le client	458
AWS Identity and Access Management	462
Public ciblé	462
Authentification avec des identités	463
Gestion des accès à l'aide de politiques	466
Autoriser des utilisateurs à afficher leurs propres autorisations	469
Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow	470
Comment Amazon MWAA fonctionne avec IAM	472
Validation de la conformité	477
Résilience	479
Sécurité de l'infrastructure	479
Configuration et analyse des vulnérabilités	480
Bonnes pratiques	480
Bonnes pratiques de sécurité dans Apache Airflow	481
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
Versions obsolètes d'Apache Airflow	486
Support des versions d'Apache Airflow et FAQ	487
Questions fréquentes (FAQ)	487
Points de terminaison et quotas	489
Points de terminaison de service	489

Quotas de service	489
Augmenter les quotas	490
FAQ	491
Versions prises en charge	492
Prise en charge d'Apache Airflow	492
Versions d'Apache Airflow	492
Version Python	492
Version Pip	494
Cas d'utilisation	494
Quand dois-je utiliser AWS Step Functions vs. Amazon MWAA ?	494
Spécifications relatives à l'environnement	495
Quelle est la capacité de stockage des tâches disponible pour chaque environnement ?	495
Système d'exploitation par défaut	495
Images personnalisées	495
Conformité à la loi américaine HIPAA	495
Amazon MWAA prend-il en charge les instances ponctuelles ?	495
Domaine personnalisé	496
Accès SSH	496
Règle d'autoréférencement	497
Métriques personnalisées	497
Stocker les données	497
Quota de travailleurs	497
VPC Amazon partagés	497
Métriques	498
Indicateurs relatifs aux travailleurs	498
Métriques personnalisées	498
DAG, opérateurs, connexions et autres questions	498
PythonVirtualenvOperator	498
Combien de temps faut-il à Amazon MWAA pour reconnaître un nouveau fichier DAG ?	498
Pourquoi mon fichier DAG n'est-il pas récupéré par Apache Airflow ?	499
Supprimer plugins.zip ou requirements.txt	499
Supprimer plugins.zip ou requirements.txt	499
Puis-je utiliser les opérateurs du Service AWS de Migration de Base de Données (DMS) ?	500
Résolution des problèmes	501
Apache Airflow v2	504
Connexions	505

Serveur web	507
Tâches	508
INTERFACE DE LIGNE DE COMMANDE (CLI)	511
Opérateurs	512
Apache Airflow v1	514
Mise à jour de requirements.txt	515
DAG cassé	515
Opérateurs	518
Connexions	518
Serveur web	520
Tâches	522
INTERFACE DE LIGNE DE COMMANDE (CLI)	525
Création/mise à jour Amazon MWAA	526
Mise à jour des requirements.txt	527
Plugins	528
Créer un compartiment.	528
Créer un environnement	529
Environnement de mise à jour	532
Environnement d'accès	532
CloudWatch Logs et CloudTrail	533
Journaux	534
Historique du document	539
.....	dcxv

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

Amazon Managed Workflows for Apache Airflow est un service d'orchestration géré pour [Apache Airflow](#) que vous pouvez utiliser pour configurer et exploiter des pipelines de données dans le cloud à grande échelle. Apache Airflow est un outil open source utilisé pour créer, planifier et surveiller par programmation des séquences de processus et de tâches appelées flux de travail. Avec Amazon MWAA, vous pouvez utiliser Apache Airflow et Python pour créer des flux de travail sans avoir à gérer l'infrastructure sous-jacente en termes d'évolutivité, de disponibilité et de sécurité. Amazon MWAA adapte automatiquement sa capacité d'exécution de flux de travail en fonction de vos besoins. Amazon MWAA s'intègre aux services de AWS sécurité pour vous fournir un accès rapide et sécurisé à vos données.

Contenu

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

Fonctionnalités

- 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 vous pouvez télécharger sur Internet.
- Mise à l'échelle automatique : dimensionnez automatiquement les travailleurs Apache Airflow en définissant le nombre minimum et maximum de travailleurs exécutés dans votre environnement. 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, jusqu'à ce que le nombre maximum de travailleurs que vous avez défini soit atteint.
- Authentification intégrée : activez l'authentification et l'autorisation basées sur les rôles pour votre serveur Web Apache Airflow en définissant les [politiques de contrôle d'accès](#) dans AWS Identity

and Access Management (IAM). Les Apache Airflow Workers adoptent ces politiques pour un accès sécurisé aux AWS services.

- Sécurité intégrée : les serveurs et les planificateurs 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 AWS Identity and Access Management (IAM) et AWS par le SSO.
- 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 : consultez les journaux Apache Airflow et les [métriques d'Apache Airflow](#) sur Amazon CloudWatch 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 AWS Batch, CloudWatch Amazon, Amazon DynamoDB, Amazon EMR, AWS DataSync Amazon EKS, Amazon Data Firehose AWS Fargate, Amazon AWS Lambda Redshift, Amazon SQS AWS Glue, Amazon SNS, Amazon et Amazon S3, ainsi que des centaines d'opérateurs intégrés et créés par la SageMaker communauté 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 ci-dessous) apparaissent sous la forme d'un seul environnement Amazon MWAA dans votre compte. L'Apache Airflow Scheduler et Workers sont AWS Fargate (Fargate) des 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 les conteneurs Scheduler et Workers Fargate peuvent accéder via un point de terminaison VPC sécurisé de manière privée.

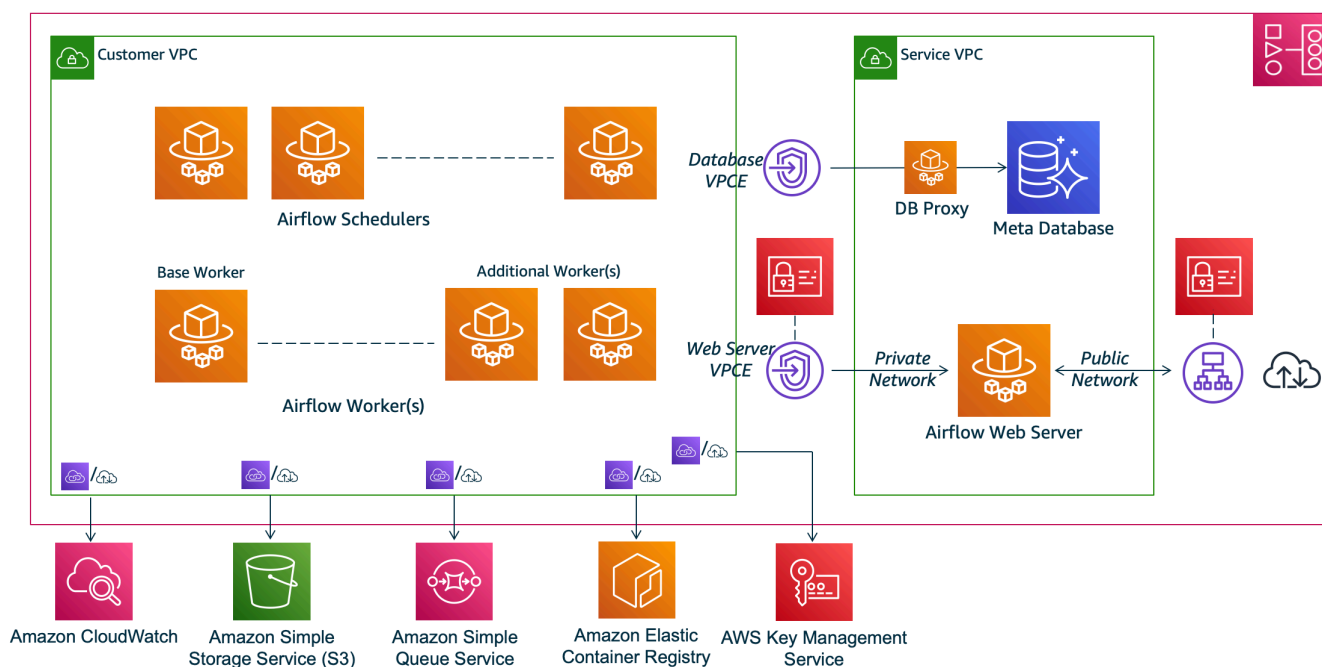
Amazon CloudWatch, Amazon S3, Amazon SQS, Amazon ECR AWS KMS sont distincts d'Amazon MWAA et doivent être accessibles depuis le ou les planificateurs Apache Airflow et les conteneurs Workers in the Fargate.

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 AWS Identity and Access Management (IAM).

Note

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.

Amazon MWAA Architecture



Integration

La communauté open source Apache Airflow active et croissante 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, 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 Amazon MWAA, et Amazon MWAA a l'intention de fournir des intégrations fiables aux AWS services, de les mettre à la disposition de la communauté et de participer au développement de fonctionnalités communautaires.

Pour un exemple de code, consultez [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, consultez. [Versions d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow](#)

Quelle est la prochaine étape ?

- Commencez avec un AWS CloudFormation modèle unique qui crée un compartiment Amazon S3 pour vos DAG Airflow 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 vos DAG Airflow 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. [Démarez 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 simultanément 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 AWS 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 : Afficher les journaux dans CloudWatch Logs](#)
- [Quelle est la prochaine étape ?](#)

Dans ce tutoriel

Ce didacticiel vous présente trois AWS Command Line Interface (AWS CLI) commandes pour télécharger un DAG sur Amazon S3, exécuter le DAG dans Apache Airflow et afficher les connexions. CloudWatch Il se termine en vous expliquant les étapes de création d'une politique IAM pour une équipe de développement d'Apache Airflow.

Note

Le AWS CloudFormation modèle de cette page crée un environnement Amazon Managed Workflows pour Apache Airflow pour la dernière version d'Apache Airflow disponible dans. AWS CloudFormation La dernière version disponible est Apache Airflow v2.8.1.

Le AWS CloudFormation modèle de cette page 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 [Créez un compartiment Amazon S3 pour Amazon S3 pour Amazon Amazon S3 pour Amazon S3](#).
- 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 l'environnement par défaut pour le chiffrement à l'aide d'une [cléAWS détenue](#), comme défini dans. [Création d'un environnement Amazon MWAA](#)
- CloudWatch Journaux. Le modèle permet à Apache Airflow de se connecter CloudWatch au niveau « INFO » et au-delà 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. [Afficher les 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. [Ajout ou mise à jour des DAG](#)
- Afficher les journaux. Affichez le groupe de journaux du serveur Web Airflow dans CloudWatch Logs, tel que défini dans [Afficher les 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](#)

Prérequis

The AWS Command Line Interface (AWS CLI) est un outil open source qui vous permet d'interagir avec les AWS services à l'aide de commandes dans votre shell 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 AWS CloudFormation modèle localement

- Copiez le contenu du modèle suivant et enregistrez-le localement sous `mwaas-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-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 MAAA  
#####  
  
MwaaEnvironment:  
  Type: AWS::Mwaa::Environment  
  DependsOn: MwaaExecutionPolicy  
  Properties:  
    Name: !Sub "${AWS::StackName}-MwaaEnvironment"  
    SourceBucketArn: !GetAtt EnvironmentBucket.Arn  
    ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn  
    DagS3Path: dags  
    NetworkConfiguration:  
      SecurityGroupIds:  
        - !GetAtt SecurityGroup.GroupId  
      SubnetIds:  
        - !Ref PrivateSubnet1  
        - !Ref PrivateSubnet2  
    WebserverAccessMode: PUBLIC_ONLY  
    MaxWorkers: !Ref MaxWorkerNodes  
    LoggingConfiguration:  
      DagProcessingLogs:  
        LogLevel: !Ref DagProcessingLogs  
        Enabled: true  
      SchedulerLogs:  
        LogLevel: !Ref SchedulerLogsLevel  
        Enabled: true  
      TaskLogs:  
        LogLevel: !Ref TaskLogsLevel  
        Enabled: true  
      WorkerLogs:  
        LogLevel: !Ref WorkerLogsLevel  
        Enabled: true  
      WebserverLogs:  
        LogLevel: !Ref WebserverLogsLevel  
        Enabled: true  
  SecurityGroup:  
    Type: AWS::EC2::SecurityGroup  
    Properties:  
      VpcId: !Ref VPC  
      GroupDescription: !Sub "Security Group for Amazon MAAA Environment  
${AWS::StackName}-MwaaEnvironment"
```

```
GroupName: !Sub "airflow-security-group-${AWS::StackName}-MwaaEnvironment"
```

SecurityGroupIngress:

```
Type: AWS::EC2::SecurityGroupIngress
```

Properties:

```
GroupId: !Ref SecurityGroup
```

```
IpProtocol: "-1"
```

```
SourceSecurityGroupId: !Ref SecurityGroup
```

SecurityGroupEgress:

```
Type: AWS::EC2::SecurityGroupEgress
```

Properties:

```
GroupId: !Ref SecurityGroup
```

```
IpProtocol: "-1"
```

```
CidrIp: "0.0.0.0/0"
```

MwaaExecutionRole:

```
Type: AWS::IAM::Role
```

Properties:**AssumeRolePolicyDocument:**

```
Version: 2012-10-17
```

Statement:

```
- Effect: Allow
```

Principal:**Service:**

```
- airflow-env.amazonaws.com
```

```
- airflow.amazonaws.com
```

Action:

```
- "sts:AssumeRole"
```

```
Path: "/service-role/"
```

MwaaExecutionPolicy:

```
DependsOn: EnvironmentBucket
```

```
Type: AWS::IAM::ManagedPolicy
```

Properties:**Roles:**

```
- !Ref MwaaExecutionRole
```

PolicyDocument:

```
Version: 2012-10-17
```

Statement:

```
- Effect: Allow
```

```
Action: airflow:PublishMetrics
```

```
Resource:
```



```

    - !Sub "arn:aws:airflow:${AWS::Region}:${AWS::AccountId}:environment/
    ${EnvironmentName}"
  - Effect: Deny
    Action: s3:ListAllMyBuckets
    Resource:
      - !Sub "${EnvironmentBucket.Arn}"
      - !Sub "${EnvironmentBucket.Arn}/*"

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

  - Effect: Allow
    Action:
      - logs:DescribeLogGroups
    Resource: "*"

  - Effect: Allow
    Action:
      - logs:CreateLogStream
      - logs:CreateLogGroup
      - logs:PutLogEvents
      - logs:GetLogEvents
      - logs:GetLogRecord
      - logs:GetLogGroupFields
      - logs:GetQueryResults
      - logs:DescribeLogGroups
    Resource:
      - !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:airflow-${AWS::StackName}*"
  - Effect: Allow
    Action: cloudwatch:PutMetricData
    Resource: "*"

  - Effect: Allow
    Action:
      - sqs:ChangeMessageVisibility
      - sqs>DeleteMessage
      - sqs:GetQueueAttributes
      - sqs:GetQueueUrl
      - sqs:ReceiveMessage

```

```

    - sqs:SendMessage
  Resource:
    - !Sub "arn:aws:sqs:${AWS::Region}:*:airflow-celery-*"
- Effect: Allow
  Action:
    - kms:Decrypt
    - kms:DescribeKey
    - "kms:GenerateDataKey*"
    - kms:Encrypt
  NotResource: !Sub "arn:aws:kms:*:${AWS::AccountId}:key/*"
  Condition:
    StringLike:
      "kms:ViaService":
        - !Sub "sqs.${AWS::Region}.amazonaws.com"

```

Outputs:**VPC:**

Description: A reference to the created VPC

Value: !Ref VPC

PublicSubnets:

Description: A list of the public subnets

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

PrivateSubnets:

Description: A list of the private subnets

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

PublicSubnet1:

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

Value: !Ref PublicSubnet1

PublicSubnet2:

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

Value: !Ref PublicSubnet2

PrivateSubnet1:

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

Value: !Ref PrivateSubnet1

PrivateSubnet2:

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

Value: !Ref PrivateSubnet2

SecurityGroupIngress:

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

```
MwaaApacheAirflowUI:
```

```
Description: MWA 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 `mwa-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 mwa-environment-public-network --  
template-body file://mwa-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
```

- 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
```

- Utilisez le script suivant pour télécharger le `tutorial.py` fichier dags dans votre dossier. Remplacez la valeur d'échantillon par `YOUR_S3_BUCKET_NAME`.

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

- Ouvrez la [page Environnements](#) sur la console Amazon MWAA.
- Choisissez un environnement.
- Choisissez Open Airflow UI.
- Dans l'interface utilisateur d'Apache Airflow, dans la liste des DAG disponibles, choisissez le DAG du didacticiel.
- 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.
- Choisissez Trigger DAG.

Quatrième étape : Afficher les journaux dans CloudWatch Logs

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

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

Quelle est la prochaine étape ?

- En savoir plus sur la façon de télécharger des DAG, de spécifier les dépendances Python dans un fichier `requirements.txt` et les plugins personnalisés `plugins.zip` dans [Utilisation des DAG sur Amazon MWAA](#) un fichier d'entrée.
- 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](#).

Démarrez avec Amazon Managed Workflows for Apache Airflow

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

Rubriques

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

Prérequis

Pour créer un environnement Amazon MWAA, vous souhaitez peut-être prendre des mesures supplémentaires pour vous assurer que vous êtes autorisé à accéder aux AWS ressources que vous devez créer.

- **AWS compte** : AWS compte autorisé à utiliser Amazon MWAA ainsi que les AWS services et ressources utilisés par votre environnement.

A propos de ce manuel

Cette section décrit l'AWS infrastructure et les ressources que vous allez créer dans ce guide.

- **Amazon VPC** : 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, comme défini dans [the section called "Création du réseau VPC"](#).

- **Compartiment Amazon S3** : compartiment Amazon S3 pour stocker vos DAG et les fichiers associés, tels que `plugins.zip` et `requirements.txt`. Votre compartiment Amazon S3 doit être configuré pour bloquer tout accès public, le contrôle de version des compartiments étant activé, comme défini dans [Créez un compartiment Amazon S3 pour Amazon S3 pour Amazon Amazon S3 pour Amazon S3](#).
- **Environnement Amazon MWAA** : environnement Amazon MWAA configuré avec l'emplacement de votre compartiment Amazon S3, le chemin d'accès à votre code DAG et tous les plug-ins 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, besoin des éléments suivants suivants suivants suivants suivants suivants suivants suivants suivants suivants suivants suivants suivants suivants 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 propriétaire](#) lorsque vous créez un environnement, ou spécifier une [clé gérée par le client](#) existante avec des autorisations d'accès à d'autres AWS services utilisés par votre environnement configurées (avancé). Pour en savoir plus, consultez [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, consultez [Rôle d'exécution Amazon MWAA](#).
- **Groupe de sécurité VPC** : groupe de sécurité VPC qui permet à Amazon MWAA d'accéder à d'autres AWS ressources 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, consultez [Sécurité de votre VPC sur Amazon MWAA](#).

Régions disponibles

Amazon MWAA est disponible dans les AWS régions suivantes suivantes suivantes suivantes suivantes suivantes suivantes suivantes suivantes

- Europe (Stockholm) - eu-north-1
- Europe (Francfort) - eu-central-1
- Europe (Irlande) - eu-west-1
- Europe (Londres) - eu-west-2
- Europe (Paris) - eu-west-3
- Asie-Pacifique (Mumbai) - ap-south-1
- Asie-Pacifique (Singapour) - ap-southeast-1
- Asie-Pacifique (Sydney) - ap-southeast-2
- Asie-Pacifique (Tokyo) - ap-northeast-1
- Asie-Pacifique (Séoul) - ap-northeast-2
- USA Est (Virginie du Nord) - us-east-1
- USA Est (Ohio) - us-east-2
- USA Ouest (Oregon) - us-west-2
- Canada (Centre) - ca-central-1
- Amérique du Sud (São Paulo) - sa-east-1

Créez un compartiment Amazon S3 pour Amazon S3 pour Amazon S3 pour Amazon S3

Ce guide décrit les étapes à suivre pour créer un bucket Amazon S3 afin de stocker vos graphes acycliques dirigés (DAG) d'Apache Airflow, vos plug-ins personnalisés dans un `plugins.zip` fichier et vos dépendances Python dans un fichier `requirements.txt`

Table des matières

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

Avant de commencer

- Vous ne pouvez modifier le nom du compartiment Amazon S3 après avoir créé le compartiment Amazon S3. 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, la gestion des versions des compartiments étant activée.
- Un compartiment Amazon S3 utilisé pour un environnement Amazon MWAA doit être situé dans la même AWS région qu'un environnement Amazon MWAA. Pour consulter la liste des AWS régions d'Amazon MWAA, consultez la section [Points de terminaison et quotas Amazon MWAA](#) dans le. Références générales AWS

Créez le compartiment.

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

Créer un compartiment

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

Les 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 de compartiment est visible dans les URL qui pointent vers les objets du compartiment.

4. Choisissez une AWS région dans Region. Il doit s'agir de la même AWS région que votre environnement Amazon MWAA.
 - Nous vous recommandons de choisir une région proche de vous afin de limiter la latence et les coûts, et répondre aux exigences légales.
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 Amazon S3 key (SSE-S3) (Clé Amazon S3 (SSE-S3)).
 - b. Choisissez AWS Key Management Service la 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 [clé AWS détenue](#) gérée par 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 dans 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 autorisera le verrouillage permanent des objets de ce bucket. Pour de plus amples informations, [consultez la Amazon S3 de l'utilisateur d'Amazon Simple Storage Service](#).

b. Choisissez l'accusé de réception.

10. Choisissez Create bucket (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 la section [Comment définir les autorisations des compartiments ACL ?](#)
- Découvrez comment supprimer un compartiment de stockage. .

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.

i 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 AWS PrivateLinkpoints de terminaison pour Amazon SQS, CloudWatch, Amazon S3, AWS KMS, et Amazon ECR. Pour plus d'informations sur la configuration AWS PrivateLink pour Amazon MWAA, voir [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

L'AWS Command Line Interface (AWS CLI) est un outil à code source libre qui vous permet d'interagir avec les services AWS à l'aide des commandes du terminal 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 le routage privé ou public pour votre Amazon VPC et Apache Airflow serveur Web. Pour consulter la liste des options, voir [the section called “Exemples de cas d'utilisation pour un Amazon VPC et un mode d'accès Apache Airflow”](#).

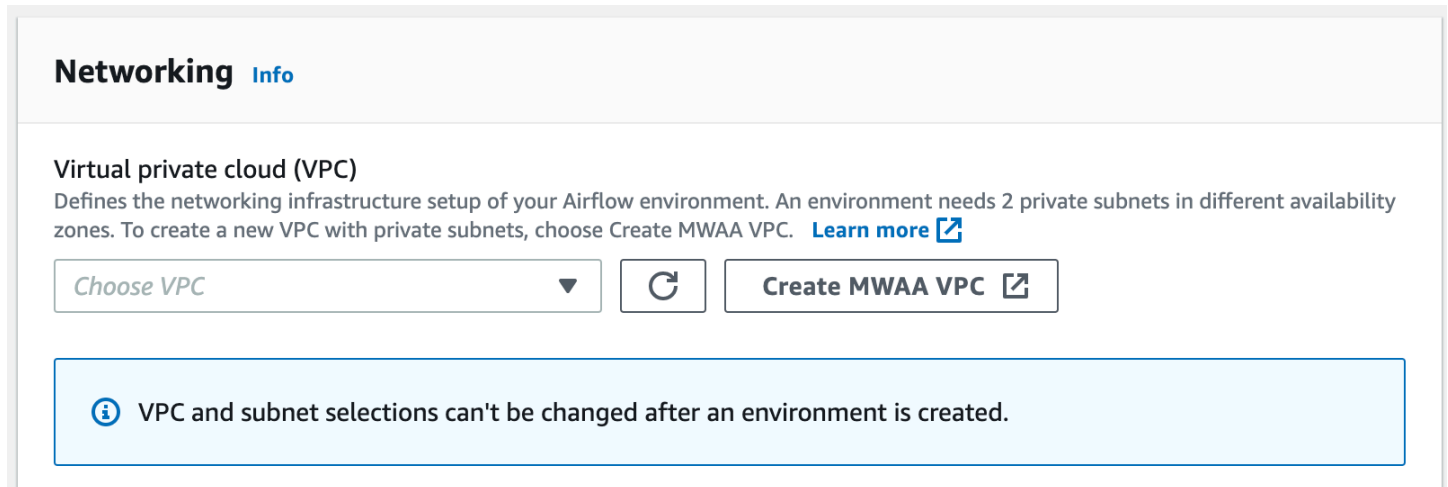
Options pour créer le réseau Amazon VPC

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

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 Apache Airflow serveur Web avec le Réseau privé ou Réseau public modes d'accès.

L'image suivante montre où vous pouvez trouver la création d'un VPC MWAABouton sur la console Amazon MWAA.



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

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

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: mwa-

VpcCIDR:

Description: Please enter the IP range (CIDR notation) for this VPC

Type: String

Default: 10.192.0.0/16

PublicSubnet1CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone

Type: String

Default: 10.192.10.0/24

PublicSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone

Type: String

Default: 10.192.11.0/24

PrivateSubnet1CIDR:

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

Type: String

Default: 10.192.20.0/24

PrivateSubnet2CIDR:

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

Type: String

Default: 10.192.21.0/24

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGatewayAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

```
InternetGatewayId: !Ref InternetGateway
VpcId: !Ref VPC

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

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

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

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

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

NatGateway1EIP:

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

NatGateway2EIP:

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

NatGateway1:

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

NatGateway2:

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

PublicRouteTable:

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

DefaultPublicRoute:

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

PublicSubnet1RouteTableAssociation:


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

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

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

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

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

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

DefaultPrivateRoute2:
Type: AWS::EC2::Route
Properties:
```

```
RouteTableId: !Ref PrivateRouteTable2
DestinationCidrBlock: 0.0.0.0/0
NatGatewayId: !Ref NatGateway2

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

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

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

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

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

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

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

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

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

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

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

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

```
cd mwaaproject
```

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

```
aws cloudformation create-stack --stack-name mwa-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

Ce qui suit AWS CloudFormation le modèle crée un réseau Amazon VPC sans accès à Internet dans votre option par défaut AWS région.

Important

Lorsque vous utilisez un Amazon VPC sans accès à Internet, vous devez autoriser Amazon ECR à accéder à Amazon S3 via un point de terminaison de passerelle. Vous pouvez créer un point de terminaison de passerelle en procédant comme suit :

1. Copiez ce qui suit JSON Politique IAM, et enregistrez-la localement sous `s3-gw-endpoint-policy.json`. La politique accorde l'autorisation minimale requise à Amazon ECR pour accéder aux ressources Amazon S3.

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

2. Créez le point de terminaison en utilisant ce qui suit AWS CLI commande. Remplacez les valeurs pour `--vpc-id` et `--route-table-ids` avec les informations relatives à votre Amazon VPC. Remplacer `--service-name` avec le nom correspondant à votre région.

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

Pour plus d'informations sur la création de points de terminaison de passerelle Amazon S3 pour Amazon ECR, consultez [Création du point de terminaison de la passerelle Amazon S3](#) dans le Guide de l'utilisateur d'Amazon Elastic Container Registry.

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

1. Copiez le contenu du modèle suivant et enregistrez-le localement sous `cf-n-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 MWA Environment to access VPC endpoints
```

```
GroupName: !Sub "${AWS::StackName}-mwa-vpc-endpoints"
```

```
SecurityGroupIngress:
```

```
Type: AWS::EC2::SecurityGroupIngress
```

```
Properties:
```

```
GroupId: !Ref SecurityGroup
```

```
IpProtocol: "-1"
```

```
SourceSecurityGroupId: !Ref SecurityGroup
```

```
SqsVpcEndpoint:
```

```
Type: AWS::EC2::VPCEndpoint
```

```
Properties:
```

```
ServiceName: !Sub "com.amazonaws.${AWS::Region}.sqs"
```

```
VpcEndpointType: Interface
```

```
VpcId: !Ref VPC
```

```
PrivateDnsEnabled: true
```

```
SubnetIds:
```

```
- !Ref PrivateSubnet1
```

```
- !Ref PrivateSubnet2
```

```
SecurityGroupIds:
```

```
- !Ref SecurityGroup
```

```
CloudWatchLogsVpcEndpoint:
```

```
Type: AWS::EC2::VPCEndpoint
```

```
Properties:
```

```
ServiceName: !Sub "com.amazonaws.${AWS::Region}.logs"
```

```
VpcEndpointType: Interface
```

```
VpcId: !Ref VPC
```

```
PrivateDnsEnabled: true
```

```
SubnetIds:
```

```
- !Ref PrivateSubnet1
```

```
- !Ref PrivateSubnet2
```

```
SecurityGroupIds:
```

```
- !Ref SecurityGroup
```

```
CloudWatchMonitoringVpcEndpoint:
```

```
Type: AWS::EC2::VPCEndpoint
```

```
Properties:
```

```
ServiceName: !Sub "com.amazonaws.${AWS::Region}.monitoring"
```

```
VpcEndpointType: Interface
```

```
VpcId: !Ref VPC
```

```
PrivateDnsEnabled: true
```

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

KmsVpcEndpoint:

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

EcrApiVpcEndpoint:

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

EcrDkrVpcEndpoint:

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


AirflowApiVpcEndpoint:

Type: AWS::EC2::VPCEndpoint

Properties:

ServiceName: !Sub "com.amazonaws.\${AWS::Region}.airflow.api"

VpcEndpointType: Interface

VpcId: !Ref VPC

PrivateDnsEnabled: true

SubnetIds:

- !Ref PrivateSubnet1

- !Ref PrivateSubnet2

SecurityGroupIds:

- !Ref SecurityGroup

AirflowEnvVpcEndpoint:

Type: AWS::EC2::VPCEndpoint

Properties:

ServiceName: !Sub "com.amazonaws.\${AWS::Region}.airflow.env"

VpcEndpointType: Interface

VpcId: !Ref VPC

PrivateDnsEnabled: true

SubnetIds:

- !Ref PrivateSubnet1

- !Ref PrivateSubnet2

SecurityGroupIds:

- !Ref SecurityGroup

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 où `cfn-vpc-private.yml` est stocké. Par exemple :

```
cd mwaaproject
```

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

```
aws cloudformation create-stack --stack-name maa-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, consultez [Gestion de l'accès aux points de terminaison Amazon VPC spécifiques à un service sur Amazon MWA](#).

Note

Vous pouvez restreindre davantage l'accès sortant dans le CIDR de votre groupe de sécurité Amazon MWA. Par exemple, vous pouvez vous limiter à elle-même en ajoutant une règle sortante autoréférencée, 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 MWA dans [Création d'un environnement Amazon MWA](#).
- Découvrez comment créer un tunnel VPN entre votre ordinateur et votre Amazon VPC avec un routage privé dans [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 devez disposer d'un AWS compte [autorisé pour utiliser Amazon MWAA](#) et d'un droit d'accès AWS Identity and Access Management (IAM) pour créer des rôles IAM. Si vous choisissez le mode d'accès 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.

Versions d'Apache Airflow

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

Note

- À partir d'Apache Airflow v2.2.2, Amazon MWAA prend en charge l'installation des exigences 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 instruction. `--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, consultez [Installation des dépendances Python](#).

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.9.2	Guide de référence d'Apache Airflow v2.9.2	Fichier de contraintes Apache Airflow v2.9.2	Python 3.11
v2.8.1	Guide de référence d'Apache Airflow v2.8.1	Fichier de contraintes Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guide de référence d'Apache Airflow v2.7.2	Fichier de contraintes Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guide de référence d'Apache Airflow v2.6.3	Fichier de contraintes Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guide de référence d'Apache Airflow v2.5.1	Fichier de contraintes Apache Airflow v2.5.1	Python 3.10

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.4.3	Guide de référence d'Apache Airflow v2.4.3	Fichier de contraintes Apache Airflow v2.4.3	Python 3.10
v2.2.2	Guide de référence d'Apache Airflow v2.2.2	Fichier de contraintes Apache Airflow v2.2.2	Python 3.7
v2.0.2	Guide de référence d'Apache Airflow v2.0.2	Fichier de contraintes Apache Airflow v2.0.2	Python 3.7

[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. Utilisez le sélecteur de AWS région pour sélectionner votre région.
3. Choisissez Create environment.
4. Sur la page Spécifier les détails, sous Détails de l'environnement :
 - a. Tapez 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'Airflow est utilisée par défaut. La dernière version disponible est Apache Airflow v2.8.1.

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. dossier DAGS. 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 votre interface utilisateur Apache Airflow est uniquement accessible 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 Groupe (s) 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, veuillez consulter la section [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 Nombre maximal de travailleurs, spécifiez le nombre maximal 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, consultez [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 AWS clé propre 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 CloudWatch Logs :
 - 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 CloudWatch Logs in Log.
 - d. Journaux des travailleurs de Airflow. Choisissez le type de journaux de travail 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, 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 [Créez un compartiment Amazon S3 pour Amazon S3 pour Amazon Amazon S3 pour Amazon S3](#) Neo-compilé,

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 AWS Identity and Access Management (IAM). Cette section décrit le rôle d'exécution utilisé pour accorder l'accès aux AWS ressources de votre environnement et comment ajouter des autorisations, ainsi que les autorisations de AWS compte dont vous avez besoin pour accéder à votre environnement Amazon MWAA et à l'interface utilisateur Apache Airflow.

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 à 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 page 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 et de définir 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 la rubrique [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. Par exemple, voir [Tutoriel : Déléguer l'accès à l' Comptes AWS aide de rôles IAM](#) dans le Guide de l'utilisateur IAM.

Sections

- [Comment ça marche](#)
- [Politique d'accès complète à la console : AmazonMWAA FullConsoleAccess](#)
- [Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess](#)
- [Politique d'accès à la console en lecture seule : AmazonMWAA ReadOnlyAccess](#)
- [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#)
- [Politique de la CLI Apache Airflow : AmazonMWAA AirflowCliAccess](#)
- [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 AWS Identity and Access Management (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 les API utilisées par un environnement.

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.

- [AmazonMWAA FullConsoleAccess](#) — Utilisez cette politique pour autoriser la configuration d'un environnement sur la console Amazon MWAA.
- [AmazonMWAA FullApiAccess](#) — Utilisez cette politique pour accorder l'accès à toutes les API Amazon MWAA utilisées pour gérer un environnement.
- [AmazonMWAA ReadOnlyAccess](#) — Utilisez cette politique pour autoriser l'accès à l'affichage des ressources utilisées par un environnement sur la console Amazon MWAA.
- [AmazonMWAA WebServerAccess](#) — Utilisez cette politique pour accorder l'accès au serveur Web Apache Airflow.
- [AmazonMWAA AirflowCliAccess](#) — 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. Pour plus d'informations, voir la rubrique [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) du 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](#) du 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 : AmazonMWAAConsoleFullAccess

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

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 MWAAConsole. Amazon MWAAConsole assume chaque rôle afin d'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 MWAAConsole (`airflow.amazonaws.com`) en tant que service auquel un rôle peut être transmis. Pour plus d'informations `iam:PassRole`, consultez la section [Accorder à un utilisateur l'autorisation de 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 MWAAConsole à l'aide d'un système [Clé détenue par AWS](#) de [chiffrement au repos](#).

À l'aide d'un Clé détenue par AWS

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

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

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

```

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

```

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

```

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

```

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

```

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

```

Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess

Un utilisateur peut avoir besoin d'accéder à la politique AmazonMWAAFullApiAccess d'autorisations s'il a besoin d'accéder à toutes les API Amazon MWAA utilisées 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 afin d'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`, consultez la section [Accorder à un utilisateur l'autorisation de 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

```
{
  "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

```

{
  "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/
AWSServiceRoleForAmazonMWSAA"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcs",
      "ec2:DescribeRouteTables"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:ListGrants",
      "kms:CreateGrant",
      "kms:RevokeGrant",
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey*",
      "kms:ReEncrypt*"
    ],
    "Resource": "arn:aws:kms::*:YOUR_ACCOUNT_ID:key/YOUR_KMS_ID"
  }

```

```

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

```

Politique d'accès à la console en lecture seule : AmazonMWAAReadOnlyAccess

Un utilisateur peut avoir besoin d'accéder à la politique d'AmazonMWAAReadOnlyAccess autorisations s'il doit consulter les ressources utilisées par un environnement sur la page de détails de l'environnement de la console Amazon MWAAR. Il n'autorise pas un utilisateur à créer de nouveaux environnements, à modifier des environnements existants ou à consulter l'interface utilisateur d'Apache Airflow.

```

{
  "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 : AmazonMWAA WebServerAccess

Un utilisateur peut avoir besoin d'accéder à la politique d'AmazonMWAAWebServerAccess autorisations s'il a besoin d'accéder à l'interface utilisateur d'Apache Airflow. Il ne permet pas à l'utilisateur de visualiser les environnements sur la console Amazon MWAA ou d'utiliser les API Amazon MWAA 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, consultez la section [Rôles par défaut](#) dans le guide de référence d'Apache Airflow.

```

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

```


Note

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, consultez [the section called "Tutoriel : Restreindre les utilisateurs à un sous-ensemble de DAG"](#).

Politique de la CLI Apache Airflow : AmazonMWAA AirflowCliAccess

Un utilisateur peut avoir besoin d'accéder à la politique AmazonMWAAAirflowCliAccess d'autorisations s'il doit exécuter des commandes de la CLI Apache Airflow (telles que `trigger_dag`). Il ne permet pas à l'utilisateur de visualiser les environnements sur la console Amazon MWAA ou d'utiliser les API Amazon MWAA pour effectuer des actions.

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

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. Sélectionnez 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 nommer la politique `AmazonMWAAReadOnlyAccess`.
7. Choisissez 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 `AmazonMWAFullConsoleAccess`, `AmazonMWAAirflowCliAccess`, et `AmazonMWAWebServerAccess` 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 [clé que vous AWS possédez](#).

Pour créer la politique `AmazonMWA FullConsoleAccess`

1. Téléchargez la politique d'[FullConsoleAccess accès d'Amazon MWA](#).
2. Ouvrez la [page Politiques](#) sur la console IAM.
3. Sélectionnez Créer une politique.
4. Choisissez l'onglet JSON.
5. Collez la politique JSON pour `AmazonMWAFullConsoleAccess`.
6. Remplacez les valeurs suivantes :
 - a. `{your-account-id}` — Votre identifiant de AWS compte (tel que `0123456789`)
 - b. `{your-kms-id}` — 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 `AmazonMWAFullConsoleAccess` le nom.
9. Choisissez Créer une politique.

Pour créer la politique AmazonMWA WebServerAccess

1. Téléchargez la politique d'[WebServerAccess accès d'Amazon MWAA](#).
2. Ouvrez la [page Politiques](#) sur la console IAM.
3. Sélectionnez Créer une politique.
4. Choisissez l'onglet JSON.
5. Collez la politique JSON pourAmazonMWAWebServerAccess.
6. Remplacez les valeurs suivantes :
 - a. *{your-region}* : région de votre environnement Amazon MWAA (telle que us-east-1)
 - b. *{your-account-id}* — votre identifiant de AWS compte (tel que0123456789)
 - c. *{your-environment-name}* : le nom de votre environnement Amazon MWAA (tel queMyAirflowEnvironment)
 - d. *{airflow-role}* – le rôle [par défaut d'AdminApache Airflow](#)
7. Choisissez Examiner une politique.
8. Tapez AmazonMWAWebServerAccess le nom.
9. Choisissez Créer une politique.

Pour créer la politique AmazonMWA AirflowCliAccess

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

Pour créer le groupe

1. Ouvrez la [page Groupes](#) sur la console IAM.
2. Entrez le nom deAirflowDevelopmentGroup.

3. Choisissez Étape suivante.
4. Tapez AmazonMWAAs 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 un 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 MWAAs

Amazon Managed Workflows pour Apache Airflow utilise des rôles liés à un [service AWS Identity and Access Management](#) (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon MWAAs. Les rôles liés à un service sont prédéfinis par Amazon MWAAs 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 MWAAs, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon MWAAs définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon MWAAs peut assumer ses rôles. Les autorisations définies comprennent la politique d'approbation 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 prenant en charge les rôles liés à un service, consultez [Services AWS qui fonctionnent avec IAM](#) et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Sélectionnez un Oui ayant un lien pour consulter la documentation du rôle lié à un service, pour 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.
 - [Des interfaces réseau élastiques \(ENI\)](#) dans votre Amazon VPC pour permettre l'accès réseau AWS aux 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.

```
{
  "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 MWAAS d'effectuer les actions suivantes sur les ressources spécifiées :

```

{
  "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, consultez [Autorisations de 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 ne vous permet pas de modifier le rôle lié au `AWSServiceRoleForAmazonMWAA` service. Une fois que vous avez 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, consultez [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 à un service à l'aide d'IAM

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonMWAA` service. Pour plus d'informations, consultez [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

Modification	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

Modification	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'autorisation des rôles liés aux services AWS gérés.	18 novembre 2022

Rôle d'exécution Amazon MWAA

Un rôle d'exécution est un rôle AWS Identity and Access Management (IAM) doté d'une politique d'autorisations qui autorise Amazon Managed Workflows for Apache Airflow à invoquer les ressources d'autres AWS 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 page décrit comment utiliser et configurer le rôle d'exécution pour votre environnement afin de permettre à Amazon MWAA d'accéder aux 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](#)
- [Afficher et mettre à jour une politique de rôle d'exécution](#)
 - [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](#)
- [Quelle est la prochaine étape ?](#)

Vue d'ensemble des rôles d'exécution

L'autorisation permettant à Amazon MWAA d'utiliser d'autres AWS services utilisés par votre environnement est obtenue à partir 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 de votre [clé gérée par le client](#)).

Note

Si vous avez choisi qu'Amazon MWAA utilise une clé KMS AWS gérée 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. Afin d'atténuer les risques associés à l'accès aux ressources entre comptes, nous vous recommandons de revoir le code placé dans vos DAG afin de vous assurer que vos flux de travail n'accèdent pas 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 `arn:aws:logs:your-region:your-account-id: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 ajouter les autorisations 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'attachant 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 DAG 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 propre](#) 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.

Permissions [Info](#)

Execution role
The IAM role used by your environment to access your DAG code, write logs, and perform other actions.

Create a new role  

Role name

AmazonMWAA-MyAirflowEnvironment-rdfjhHm

Use alphanumeric and '+,=, @, -, _' characters. Maximum 64 characters.

Afficher et mettre à jour une politique de rôle d'exécution

Vous pouvez consulter le 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. Sélectionnez 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

```
{
  "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 Attach Politiques (Attacher des politiques).
5. Choisissez 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 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.

```
{  
  "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, veuillez consulter la section [Gestion des connexions à Apache Airflow](#).

Exemples de politiques JSON pour un rôle d'exécution

Les exemples de politiques d'autorisation présentés dans cette section présentent deux politiques que vous pouvez utiliser 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 MWA](#).

Nous vous recommandons de copier l'exemple de politique, de remplacer les exemples d'ARN ou d'espaces réservés, puis d'utiliser la politique JSON pour créer ou mettre à jour un rôle d'exécution. Par exemple, remplacer `{your-region}` par `us-east-1`.

Exemple de politique pour une clé gérée par le client

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

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



```

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

```

```

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

```

Ensuite, vous devez autoriser Amazon MWA à assumer ce rôle afin d'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 Vous trouverez ci-dessous un exemple de document relatif à la politique d'acceptation des rôles :

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

Exemple de politique pour une clé AWS détenue

L'exemple suivant montre une politique de rôle d'exécution que vous pouvez utiliser pour une [clé AWS détenue](#).

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

```

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

```

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

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 un problème de confusion chez les 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 appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé d'accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services avec des principaux de service qui ont eu 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.

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

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

Modes d'accès à 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 à 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 \(accès réseau privé\)](#)

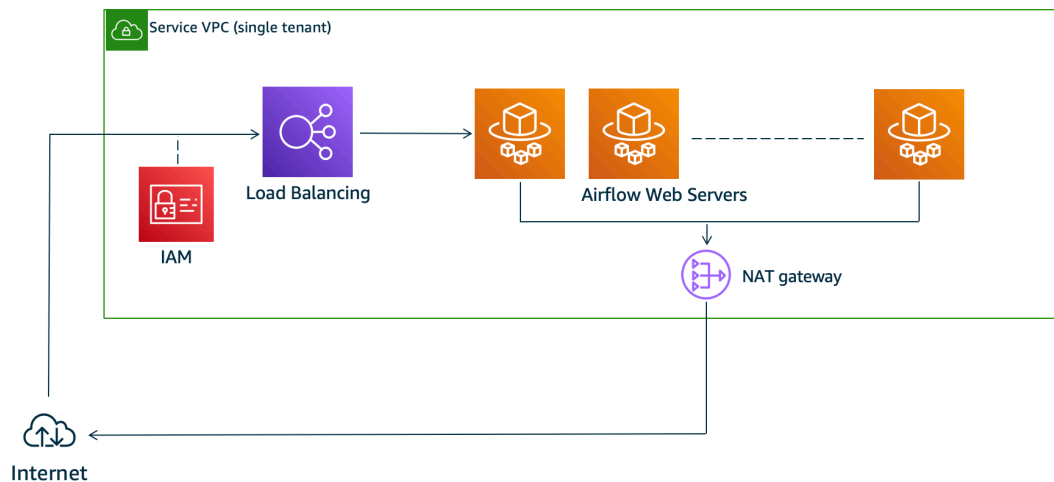
Modes d'accès à Apache Airflow

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 montre un environnement Amazon MWAA avec un serveur Web public.

Public Web Server Option



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

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

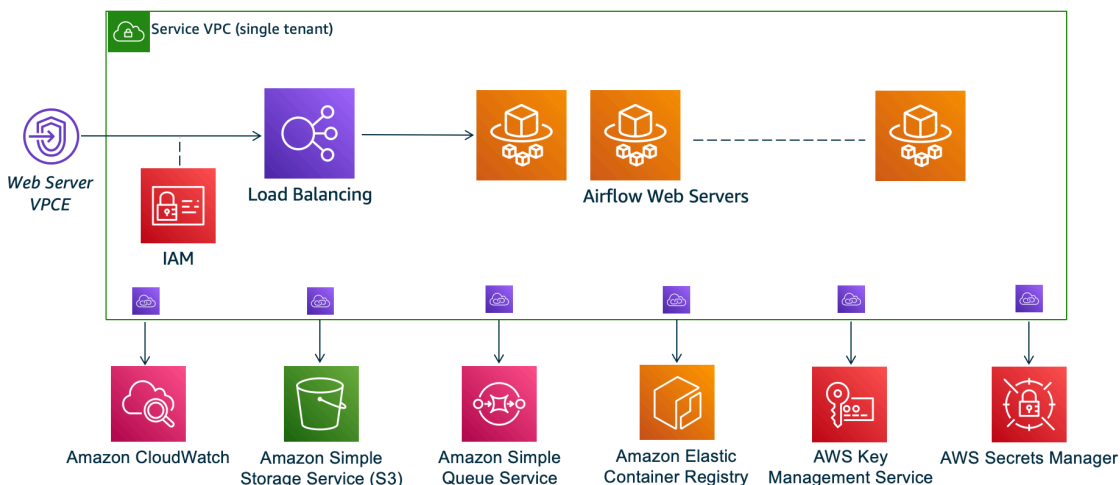
Web server access

- Private network (Recommended)
Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.
- Public network (No additional setup)
Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.

Réseau privé

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

Private Web Server Option



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 à un serveur Web privé, vous devez emballer toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans votre `requirements.txt`. Pour obtenir des instructions sur l'emballage et l'installation de vos dépendances à l'aide de wheel, consultez [la section Gestion des dépendances à l'aide de Python Wheel](#).

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

Web server access

Private network (Recommended)

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

Public network (No additional setup)

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

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 AWS comptes.
- 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.

Pour en savoir plus, veuillez consulter la section [the section called “Exemples de cas d'utilisation pour un Amazon VPC et un 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, veuillez consulter la section [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, veuillez consulter la section [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, veuillez consulter la section [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, veuillez consulter la section [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, consultez 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 afficher et appeler un DAG dans votre interface utilisateur Apache Airflow, ou utiliser les API Amazon MWAA pour obtenir un jeton et appeler 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

- [Prérequis](#)
- [Ouvrez l'interface utilisateur d'Apache Airflow](#)
- [Connexion 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](#)

Prérequis

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

Accès

- AWS accès au compte dans AWS Identity and Access Management (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#)
- AWS accès au compte AWS Identity and Access Management (IAM) conformément à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess](#)

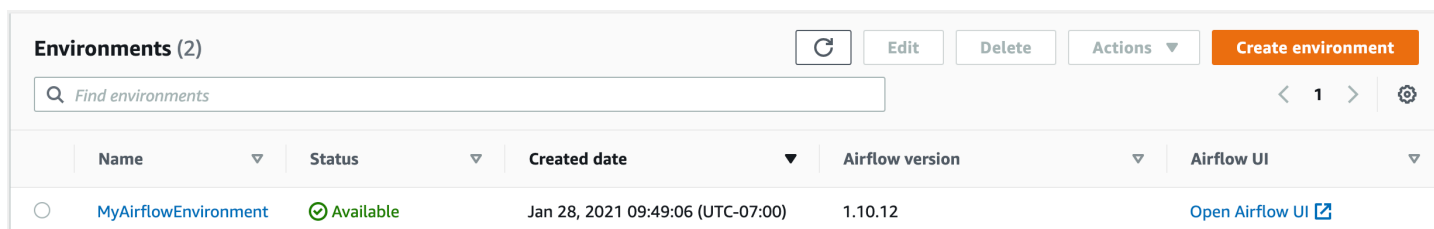
AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source qui vous permet d'interagir avec les AWS services à l'aide de commandes dans votre shell 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 montre le lien vers votre interface utilisateur Apache Airflow sur la console Amazon MWAA.



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

Connexion à Apache Airflow

Vous devez [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#) disposer d'autorisations pour accéder à votre AWS compte dans AWS Identity and Access Management (IAM) pour 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 vous permet d'accéder à votre environnement Amazon MWAA. Par exemple, vous pouvez obtenir un jeton, puis déployer des DAG par programmation à l'aide des API Amazon MWAA.

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.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [En utilisant le AWS CLI](#)
- [Utiliser un script bash](#)
- [Utilisation d'une requête d'API POST](#)
- [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

- AWS accès au compte dans AWS Identity and Access Management (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#)
- AWS accès au compte AWS Identity and Access Management (IAM) conformément à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source qui vous permet d'interagir avec les AWS services à l'aide de commandes dans votre shell 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-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://$HOST/aws_mwa/aws-console-ssso?login=true#
WEB_TOKEN=$(aws mwa create-web-login-token --name YOUR_ENVIRONMENT_NAME --query
  WebToken --output text)
echo $YOUR_URL$WEB_TOKEN
```

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

```
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
```

5. Vous devriez voir ce qui suit dans votre invite de commande :

```
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'une requête d'API POST

L'exemple suivant utilise une requête d'API POST pour créer un jeton de connexion Web Apache Airflow.

1. Copiez l'URL suivante et collez-la dans le champ URL de votre client d'API REST.

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

2. Remplacez les espaces réservés en *rouge par*. *YOUR_HOST_NAME* Par exemple, le nom d'hôte d'un réseau public peut ressembler à ceci (sans le https ://) :

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

3. Copiez le code JSON suivant et collez-le dans le champ body de votre client d'API REST.

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

4. Remplacez l'espace réservé en *rouge par*. *YOUR_ENVIRONMENT_NAME*
5. Ajoutez des paires clé-valeur dans le champ d'autorisation. Par exemple, si vous utilisez Postman, choisissez AWS Signature, puis entrez votre :

- *AWS_ACCESS_KEY_ID* dans AccessKey
- *AWS_SECRET_ACCESS_KEY* dans SecretKey

6. Vous devriez voir la réponse suivante :

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

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

2. Remplacez l'espace réservé en *rouge par* `YOUR_ENVIRONMENT_NAME`
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 Apache Airflow géré. 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.

L'exemple suivant montre comment utiliser le AWS CLI pour créer un environnement avec un nom de domaine de serveur Web personnalisé.

```
$ aws mwa create-environment \
  --name my-mwaa-env \
  --source-bucket-arn arn:aws:s3:::my-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 AWS compte 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 AWS compte.

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 réussite, 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 commande Amazon EC2 `describe-vpc-endpoints` suivante. `--filters Name=service-name,Values=web-server-vpc-endpoint-service-id` dans la commande suivante.

- 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 les identifiants d'interface réseau associés 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
```

- 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 [AWS 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, `commesg-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 commande Amazon EC2 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 commande Amazon EC2 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 “Didacticiel : AWS Client VPN”](#)

Création d'un jeton CLI Apache Airflow

Vous pouvez utiliser les commandes de cette page pour générer un jeton CLI, puis effectuer des appels d'API Amazon Managed Workflows pour Apache Airflow directement dans votre interface de commande. Par exemple, vous pouvez obtenir un jeton, puis déployer des DAG par programmation à l'aide des API Amazon MWAA. La section suivante inclut les étapes permettant de créer un jeton CLI Apache Airflow à l'aide d'un script curl, d'un script Python ou d'un script bash. Le jeton renvoyé dans la réponse est valide pendant 60 secondes.

Note

Le AWS CLI jeton est destiné à remplacer les actions synchrones du shell, et non les commandes d'API asynchrones. En tant que telle, la simultanéité disponible est limitée. Pour garantir que le serveur Web reste réactif pour les utilisateurs, il est recommandé de ne pas ouvrir de nouvelle AWS CLI demande tant que la précédente ne s'est pas terminée correctement.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [Utilisation du AWS CLI](#)
- [Utilisation d'un script curl](#)
- [Utiliser un script bash](#)
- [Utiliser 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

- AWSaccès du compteAWS Identity and Access Management (IAM) à la politique d'autorisations Amazon MWAA dans [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#).
- AWSaccès au compteAWS Identity and Access Management (IAM) à la politique d'autorisations Amazon MWAA [Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess](#).

AWS CLI

L'AWS Command Line Interface (AWS CLI) est un outil à code source libre qui vous permet d'interagir avec les services AWS à l'aide des commandes du terminal de ligne de commande. Pour effectuer les étapes de cette page, vous avez besoin des éléments suivants :

- [AWS CLI— Installer la version 2](#).
- [AWS CLI— Configuration rapide avecaws configure](#).

Utilisation du AWS CLI

L'exemple suivant utilise la [create-cli-token](#)commande duAWS CLI pour créer un jeton CLI 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 leAWS CLI afin d'appeler la CLI Apache Airflow via un point de terminaison sur le serveur Web Apache Airflow.

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 bloc-notes, vous devrez peut-être utiliser **Édition > Coller** dans le menu de votre coque.

```
CLI_JSON=$(aws mwaas --region YOUR_REGION create-cli-token --
name YOUR_ENVIRONMENT_NAME) \
  && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
  && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
  && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaas/
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 *YOUR_REGION* par la **AWS** région pour votre environnement *YOUR_DAG_NAME*, et *YOUR_ENVIRONMENT_NAME*. Par exemple, le nom d'hôte d'un réseau public peut ressembler à ceci (sans le `https://`) :

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

3. Vous devriez voir ce qui suit dans votre invite de commande :

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

Apache Airflow v1

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

Note

Après l'avoir copié dans votre bloc-notes, vous devrez peut-être utiliser **Édition > Coller** dans le menu de votre coque.

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

2. Remplacez les espaces réservés *YOUR_REGION* par la **AWS** région pour votre environnement *YOUR_DAG_NAME*, et *YOUR_HOST_NAME*. Par exemple, le nom d'hôte d'un réseau public peut ressembler à ceci (sans le `https://`) :

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

3. Vous devriez voir ce qui suit dans votre invite de commande :

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

4. Remplacez les espaces réservés par *YOUR_ENVIRONMENT_NAME* et *YOUR_DAG_NAME*.

Utiliser un script bash

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

Apache Airflow v2

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le 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 en *rouge* par `YOUR_ENVIRONMENT_NAME`, `YOUR_HOST_NAME`, et `YOUR_DAG_NAME`. Par exemple, le nom d'hôte d'un réseau public peut ressembler à ceci (sans le `https://`) :

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

3. (facultatif) Les utilisateurs macOS et Linux peuvent avoir besoin d'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 CLI Apache Airflow.

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

Apache Airflow v1

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le 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 "trigger_dag YOUR_DAG_NAME"
```

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

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

3. (facultatif) Les utilisateurs macOS et Linux peuvent avoir besoin d'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 CLI Apache Airflow.

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

Utiliser 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. Il ne vous reste qu'à 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 v2

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le 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 CLI Apache Airflow.

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

Apache Airflow v1

1. Copiez le contenu de l'exemple de code suivant et enregistrez-le sous `create-cli-token.py`.

```
import boto3
import json
import requests
import base64

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

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

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

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

```
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 CLI 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 à l'adresse [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, en fournissant un moyen standardisé d'invoquer des flux de travail d'orchestration de données, de gérer vos DAG et de surveiller l'état des différents composants d'Apache Airflow tels que la base de données de métadonnées, le déclencheur et le planificateur.

Afin de prendre en charge directement l'utilisation de l'API REST Apache Airflow, Amazon MWAA vous offre la possibilité de dimensionner horizontalement la capacité du serveur Web pour faire face à une demande accrue, qu'elle soit due à des demandes d'API REST, à l'utilisation de l'interface de ligne de commande (CLI) ou à 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 d'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 : la prise en charge des API REST vous permet de 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 le statut de vos DAG dans plusieurs environnements Amazon MWAA, permettant ainsi une surveillance et une gestion centralisées.

Les rubriques suivantes montrent comment obtenir un jeton d'accès au serveur Web, puis comment utiliser ce jeton pour effectuer des appels d'API vers l'API REST d'Apache Airflow. Dans l'exemple suivant, vous allez appeler l'API pour démarrer une nouvelle exécution du DAG.

Pour plus d'informations sur l'API REST Apache Airflow, consultez le manuel de référence [de l'API REST Apache Airflow](#).

Rubriques

- [Création d'un jeton de session de serveur Web](#)
- [Appelez l'API REST Apache Airflow](#)

Création d'un jeton de session de serveur Web

Pour créer un jeton d'accès au serveur Web, utilisez la fonction Python suivante. Cette fonction appelle d'abord l'API Amazon MWAA 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.

Note

Le jeton de session expire au bout de 12 heures.

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

    try:
        # Initialize MAAA client and request a web login token
        maaa = boto3.client('maaa', region_name=region)
        response = maaa.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_maaa/login"
        login_payload = {"token": web_token}

        # Make a POST request to the MAAA 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 MAAA 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
```

Appelez l'API REST Apache Airflow

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 ci-dessous, utilisez le point de terminaison `/dags/dag_id/dag`.

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

    Args:
    region (str): AWS region where the MAAA environment is hosted.
    env_name (str): Name of the MAAA 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 page décrit les commandes de la CLI Apache Airflow prises en charge et non prises en charge sur Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [Prérequis](#)
 - [Accès](#)
 - [AWS CLI](#)
- [Ce qui a changé dans la version 2](#)

- [Commandes CLI prises en charge](#)
 - [Commandes prises en charge](#)
 - [Utilisation de commandes qui analysent les DAG](#)
- [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](#)
 - [Exemples GitHub et AWS didacticiels](#)

Prérequis

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

Accès

- AWS accès au compte dans AWS Identity and Access Management (IAM) à la politique d'autorisation Amazon MWAA dans. [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#)
- AWS accès au compte AWS Identity and Access Management (IAM) conformément à la politique d'autorisation Amazon MWAA. [Politique complète d'accès à l'API et à la console : AmazonMWAA FullApiAccess](#)

AWS CLI

The AWS Command Line Interface (AWS CLI) est un outil open source qui vous permet d'interagir avec les AWS services à l'aide de commandes dans votre shell 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`.](#)

Ce qui a changé dans la version 2

- Nouveau : 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 maintenant `dags unpause` dans Apache Airflow v2. Pour en savoir plus, consultez les [modifications apportées à la CLI Airflow en 2 dans](#) le guide de référence d'Apache Airflow.

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 v2

Versions mineures	Command	
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	
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	

Versions mineures	Command	
v2.6+	jours list-import-errors	
v2.2+ (remarque)	listes de jours	
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	
version 2.0 et versions ultérieures	jours d'inpause	
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	

Versions mineures	Command	
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+	secrets des fournisseurs	
v2.7+	déclencheur des fournisseurs	
version 2.0 et versions ultérieures	widgets des fournisseurs	
v2.6+	rôles add perms	
v2.6+	rôles del-perms	
v2.6+	les rôles créent	
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	

Versions mineures	Command	
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 les DAG

Si votre environnement exécute Apache Airflow v1.10.12 ou v2.0.2, les commandes CLI qui analysent les DAG 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 vos DAG n'utilisent 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 de `<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_mwa/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 v1 et Apache Airflow v2 pour ajouter une configuration lors du déclenchement d'un DAG, par exemple. `airflow trigger_dag 'dag_name' -conf '{"key":"value"}`

```
import boto3
import json
import requests
import base64

mwa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
key = "YOUR_KEY"
value = "YOUR_VALUE"
conf = "{\\" + key + \":\\" + value + \"}"
client = boto3.client('mwa')

mwa_cli_token = client.create_cli_token(
    Name=mwa_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 = "trigger_dag {0} -c '{1}'".format(dag_name, conf)

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)
```

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

L'exemple suivant montre comment 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
```

Exemples GitHub et AWS didacticiels

- [Utilisation des paramètres et variables d'Apache Airflow v2.0.2 dans Amazon Managed Workflows pour Apache Airflow](#)
- [Interaction avec Apache Airflow v1.10.12 sur Amazon MWAA via la ligne de commande](#)
- [Commandes interactives avec Apache Airflow v1.10.12 sur Amazon MWAA et Bash Operator](#) sur GitHub

Gestion des connexions à Apache Airflow

Cette section décrit les différentes manières de configurer une connexion Airflow pour un environnement Amazon Managed Workflows pour Apache Airflow.

Rubriques

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

Présentation des variables et des connexions Apache Airflow

Dans certains cas, vous souhaitez peut-être spécifier des connexions ou des variables supplémentaires pour un environnement, comme 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.

- Self-managed Apache AirFlow 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 d'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.

Package du fournisseur Apache Airflow installés sur les environnements Amazon MWAA

Amazon MWAA installe des [fournisseurs supplémentaires](#) pour les types de connexion Apache Airflow v2 et supérieures lorsque vous créez un nouvel environnement. L'installation des packages du fournisseur vous permet de visualiser 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 v2.

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.

Table des matières

- [Package de fournisseurs pour les connexions Apache Airflow v2.9.2](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.8.1](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.7.2](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.6.3](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.5.1](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.4.3](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.2.2](#)
- [Package de fournisseurs pour les connexions Apache Airflow v2.0.2](#)
- [Spécifier les nouveaux packages de fournisseurs](#)

Package de fournisseurs pour les connexions Apache Airflow v2.9.2

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.9.2, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Note

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur. Pour plus d'informations sur la spécification de nouvelles versions, consultez [the section called "Spécifier les nouveaux packages de fournisseurs"](#).

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon[aiobotocore]==8,24.0
Connexion Postgres	apache-airflow-providers-postgres==5,11
Connexion FTP	apache-airflow-providers-ftp==3,9.1
Fab Connection	apache-airflow-providers-fab==1.1.1
Connexion au céleri	apache-airflow-providers-celery==3.7.2
Connexion HTTP	apache-airflow-providers-http==4.11.1
Connexion IMAP	apache-airflow-providers-imap==3,6.1
SQL commun	apache-airflow-providers-common-sql==1,14.0
Connexion SQLite	apache-airflow-providers-sqlite==3,8.1
Connexion SMTP	apache-airflow-providers-smtp==1,7.1

Packages de fournisseurs pour les connexions Apache Airflow v2.8.1

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.8.1, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Note

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur. Pour plus d'informations sur la spécification de nouvelles versions, consultez [the section called “Spécifier les nouveaux packages de fournisseurs”](#).

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon[aiobotocore]==8.16.0
Connexion Postgres	apache-airflow-providers-postgres==5,1,0
Connexion FTP	apache-airflow-providers-ftp==3,7,0
Connexion au céleri	apache-airflow-providers-celery==3.5.1
Connexion HTTP	apache-airflow-providers-http==4,8,0
Connexion IMAP	apache-airflow-providers-imap==3,5,0
SQL commun	apache-airflow-providers-common-sql==1,1,0
Connexion SQLite	apache-airflow-providers-sqlite==3,7,0

Packages de fournisseurs pour les connexions Apache Airflow v2.7.2

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.7.2, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Note

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur. Pour plus d'informations sur la spécification de nouvelles versions, consultez [the section called “Spécifier les nouveaux packages de fournisseurs”](#).

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

Packages de fournisseurs pour les connexions Apache Airflow v2.6.3

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.6.3, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Note

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur. Pour plus d'informations sur la spécification de nouvelles versions, consultez [the section called "Spécifier les nouveaux packages de fournisseurs"](#).

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon[aiobotocore]==8.2.0
Connexion Postgres	apache-airflow-providers-postgres==5,5.1

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

Packages de fournisseurs pour les connexions Apache Airflow v2.5.1

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.5.1, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Note

Vous pouvez spécifier la dernière version prise en charge de `apache-airflow-providers-amazon` pour mettre à niveau ce fournisseur. Pour plus d'informations sur la spécification de nouvelles versions, consultez [the section called “Spécifier les nouveaux packages de fournisseurs”](#).

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon==7,10
Connexion Postgres	apache-airflow-providers-postgres==5,4.0
Connexion FTP	apache-airflow-providers-ftp==3,3,0
Connexion au céleri	apache-airflow-providers-celery==3,10
Connexion HTTP	apache-airflow-providers-http==4.1.1

Type de connexion	Package
Connexion IMAP	apache-airflow-providers-imap==3.1.1
SQL commun	apache-airflow-providers-common-sql==1.3.3
Connexion SQLite	apache-airflow-providers-sqlite==3.3.1

Packages de fournisseurs pour les connexions Apache Airflow v2.4.3

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.4.3, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

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
Connexion SQLite	apache-airflow-providers-sqlite==3.2.1

Packages de fournisseurs pour les connexions Apache Airflow v2.2.2

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.2.2, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Type de connexion	Package
AWS Raccordement	apache-airflow-providers-amazon==2,4,0
Connexion Postgres	apache-airflow-providers-postgres==2,3,0
Connexion FTP	apache-airflow-providers-ftp==2,0,1
Connexion au céleri	apache-airflow-providers-celery==2,10
Connexion HTTP	apache-airflow-providers-http==2,0,1
Connexion IMAP	apache-airflow-providers-imap==2,0,1
Connexion SQLite	apache-airflow-providers-sqlite==2,0,1

Packages de fournisseurs pour les connexions Apache Airflow v2.0.2

Lorsque vous créez un environnement Amazon MWAA dans Apache Airflow v2.0.2, Amazon MWAA installe les packages de fournisseurs suivants utilisés pour les connexions Apache Airflow.

Type de connexion	Package
Connexion Tableau	apache-airflow-providers-tableau==10,0
Connexion Databricks	apache-airflow-providers-databricks==1,0,1
Connexion SSH	apache-airflow-providers-ssh==1,3,0
Connexion Postgres	apache-airflow-providers-postgres==1,0,2
Connexion Docker	apache-airflow-providers-docker==1,2,0
Connexion Oracle	apache-airflow-providers-oracle==1,10
Connexion Presto	apache-airflow-providers-presto==1,0,2
Connexion SFTP	apache-airflow-providers-sftp==1,2,0

Spécifier les nouveaux packages de fournisseurs

À partir de la version 2.7.2 d'Apache Airflow, votre fichier d'exigences doit inclure une instruction. `--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, toutefois, 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 spécifiques à la version depuis `https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt`](https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt) »
2. Modifiez la `apache-airflow-providers-amazon` version du fichier de contraintes selon la version que vous souhaitez utiliser.
3. Enregistrez le fichier de contraintes modifié dans le dossier Amazon S3 dags de votre environnement Amazon MWAA, par exemple sous `constraints-3.11-updated.txt`
4. 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 à l'aide du local-runner Amazon MWAA.](#)

Présentation des types de connexion des types de connexion

Apache Airflow enregistre les connexions sous la forme d'une 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 à l'aide du modèle de connexion HTTP. La principale différence réside dans le préfixe 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 des pertes de données et des perturbations de la disponibilité de votre environnement.

Rubriques

- [Exemple de chaîne d'URI de connexion](#)
- [Exemple de modèle de connexion 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 montre 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 de connexion

L'exemple suivant présente le modèle de connexion HTTP de l'interface d'Apache Airflow.

Apache Airflow v2

L'exemple suivant montre le modèle de connexion HTTP pour Apache Airflow v2 dans l'interface utilisateur d'Apache Airflow.

Add Connection

Conn Id *	<input type="text"/>
Conn Type *	<input type="text" value="HTTP"/> <small>Conn Type missing? Make sure you've installed the corresponding Airflow Provider Package.</small>
Description	<input type="text"/>
Host	<input type="text"/>
Schema	<input type="text"/>
Login	<input type="text"/>
Password	<input type="text"/>
Port	<input type="text"/>
Extra	<input type="text"/>

Apache Airflow v1

L'exemple suivant montre le modèle de connexion HTTP pour Apache Airflow v1 dans l'interface utilisateur d'Apache Airflow.

Add Connection	
Conn Id *	<input type="text"/>
Conn Type	<input type="text" value="HTTP"/>
Host	<input type="text"/>
Schema	<input type="text"/>
Login	<input type="text"/>
Password	<input type="text"/>
Port	<input type="text"/>
Extra	<input type="text"/>

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

L'exemple suivant montre comment utiliser le modèle de connexion HTTP pour un type de connexion Jdbc dans Apache Airflow v2.0.2, et les mêmes valeurs dans le modèle de connexion Jdbc pour Apache Airflow v1.10.12 dans l'interface utilisateur d'Apache Airflow.

Apache Airflow v2

L'exemple suivant montre la chaîne d'URI de connexion générée par Apache Airflow pour l'exemple de 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
```

L'exemple suivant montre comment utiliser le modèle de connexion HTTP pour une connexion Jdbc pour Apache Airflow v2 dans l'interface utilisateur d'Apache Airflow.

Add Connection

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

[Save](#) [←](#)

Apache Airflow v1

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

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

L'exemple suivant montre le modèle de connexion Jdbc pour Apache Airflow v1.10.12 dans l'interface utilisateur d'Apache Airflow.

Add Connection	
Conn Id *	<input type="text" value="my_jdbc_conn"/>
Conn Type	<input type="text" value="Jdbc Connection"/>
Connection URL	<input type="text" value="myconnectionrurl/some/path"/>
Login	<input type="text" value="mylogin"/>
Password	<input type="password"/>
Driver Path	<input type="text" value="/usr/local/airflow/dags/classpath/redshift-jdbc42-2.0.0.1.jar"/>
Driver Class	<input type="text" value="redshift-jdbc42-2.0.0.1"/>

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. Ce guide 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 seront facturés. Pour plus d'informations sur les tarifs de Secrets Manager, consultez la section [AWS Tarification](#).

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 stratégie IAM suivante autorise l'accès en lecture-écriture à l'aide de la AWS stratégie gérée. [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 Attach Policies (Attacher des politiques).
5. Tapez `SecretsManagerReadWrite` dans le champ de texte Politiques de filtrage.
6. Choisissez Attach policy (Attacher une 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 une AWS région spécifique dans Secrets Manager.

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

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

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 AWS Secrets Manager backend. Si vous utilisez un paramètre de configuration du même nom dans `airflow.cfg`, la configuration que vous créez dans les étapes suivantes aura priorité et remplacera 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"}` Ceci configure Apache Airflow pour rechercher les chaînes de connexion et les variables dans les chemins et les chemins `airflow/connections/*`. `airflow/variables/*`

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, consultez [the section called "Spécifier les nouveaux packages de fournisseurs"](#)

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'exemple dans YOUR_EXECUTION_ROLE_ARN par l'ARN du rôle d'exécution et par la région dans YOUR_REGION (telle que). us-east-1*

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

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 *du rouge*.

- 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 MWA :

```
... 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 MWA :

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

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

```
... 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 URI de connexion :

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

Vous devriez voir 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%2FAmazonMWA-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 `sousmwa_connection.py`.

```
import urllib.parse

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

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

2. Remplacez les espaces réservés par du *rouge*.
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 Spécifiez les paires clé/valeur à stocker dans ce volet secret, choisissez Texte en clair.
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 du nom 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 Spécifiez les paires clé/valeur à stocker dans ce volet secret, choisissez Texte en clair.
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
%2FAmazonMWA-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 la connexion sera analysée sous la forme d'une chaîne unique.

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 du nom secret au format suivant.

```
airflow/connections/YOUR_CONNECTION_NAME
```

Par exemple :

```
airflow/connections/myconn
```

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.

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 [Utilisation d'une clé secrète dansAWS Secrets Managerpour une connexion Apache Airflow](#) l'adresse.
- 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 [Utilisation d'une clé secrète dansAWS Secrets Managerpour une variable Apache Airflow](#) l'adresse.

Ressources

- Pour plus d'informations sur la configuration des secrets de Secrets Manager à l'aide de la console et du AWS CLI, consultez 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 qui vous permettent de remplacer les configurations d'Apache Airflow qui ne sont normalement accessibles que dans `airflow.cfg`. Ce guide 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](#)
- [Mise à niveau 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 page décrit chaque classe d'environnement Amazon MWAA et les étapes à suivre pour 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 (vCPU) pour chaque classe d'environnement. Les tâches simultanées répertoriées supposent que la simultanéité des tâches ne dépasse pas la capacité d'Apache Airflow Worker dans l'environnement.

Dans le tableau suivant, la capacité du DAG fait référence aux définitions du DAG, et non aux exécutions, et suppose que vos DAG sont [dynamiques](#) dans un seul fichier Python et écrits 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.small

- Capacité jusqu'à 50 DAG
- 5 tâches simultanées (par défaut)
- 1 vCPU
- 2 GO DE RAM

mw1.medium

- Capacité jusqu'à 200 DAG
- 10 tâches simultanées (par défaut)
- 2 vCPU
- 4 GO DE RAM

mw1.large

- Capacité jusqu'à 1000 DAG
- 20 tâches simultanées (par défaut)
- 4 vCPU
- 8 Go de RAM

mw1.xlarge

- Capacité jusqu'à 2 000 DAG
- 40 tâches simultanées (par défaut)
- 8 vCPU
- 24 GO DE RAM

mw1.2xlarge

- Capacité jusqu'à 4000 DAG
- 80 tâches simultanées (par défaut)
- 16 vCPU
- 48 GO DE RAM

Vous pouvez l'utiliser `celery.worker.autoscale` pour augmenter le nombre de tâches par travailleur. Pour plus d'informations, consultez le [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 v2

- v2 - Accepte les valeurs comprises entre et 25. La valeur par défaut est 2.

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 page 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 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 les DAG (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 pourrez toujours exécuter un total de `14` tâches simultanément que si vous disposez de `2` DAG. Cela étant dit, chaque DAG est configuré pour exécuter uniquement sept tâches simultanément (in `core.dag_concurrency`), même si le parallélisme global est défini sur `100` (in `core.parallelism`

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

Cela signifie que vous pouvez exécuter 50 tâches simultanément dans votre environnement. Toutes les tâches supérieures à 50 seront mises en file d'attente et attendront 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, augmenter le nombre maximum de travailleurs de 10 à 20 doublerait 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 mwaas update-environment --max-workers 10 --min-workers 10 --  
name YOUR_ENVIRONMENT_NAME
```

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

Tâches de dépannage 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 votre interface utilisateur Apache Airflow. Pour plus d'informations, consultez la rubrique relative à la [Je constate que mes tâches sont bloquées ou ne sont pas terminées](#) 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 pendant les pics de charge. En redimensionnant 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 le dimensionnement des serveurs Web](#)
- [Utilisation de la console Amazon MWAA](#)

Comment fonctionne le dimensionnement des serveurs 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 CPUUtilization et que ActiveConnectionCount les valeurs et diminuent, Amazon MWAA demande à Fargate de réduire les 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 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 page décrit les options de configuration d'Apache Airflow disponibles et explique comment les utiliser 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 plug-ins dans Apache Airflow v2](#)
- [Vue d'ensemble des options de configuration](#)
 - [Options de configuration d'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 AWS compte doit avoir été autorisé par votre administrateur à accéder à la politique de contrôle d'[FullConsoleaccès d'AmazonMWAA](#) 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 à Apache Airflow"](#).
- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAG, vos 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 `webserver.expose_config` pour exposer les configurations.

Utilisation des options de configuration pour charger des plugins dans Apache Airflow v2

Par défaut, dans Apache Airflow v2, les plugins sont configurés pour être chargés « paresseusement » à l'option `core.lazy_load_plugins : True` aide de ce paramètre. Si vous utilisez des plugins personnalisés dans Apache Airflow v2, vous devez les ajouter en `core.lazy_load_plugins : False` 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 des options de configuration 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 d'Apache Airflow

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

Airflow configuration options - *optional* [Info](#)

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

All Airflow configuration options are using default values.

Add custom configuration value

Référence Apache Airflow

Pour obtenir la liste des options de configuration prises en charge par Apache Airflow, consultez la section [Référence de configuration](#) du guide de référence Apache Airflow. Pour consulter les 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 saisissez 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 présente les options de configuration des notifications par e-mail Airflow disponibles sur Amazon MWAA.

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](#).

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	email.email_backend	L'utilitaire Apache Airflow utilisé pour les notifications par e-mail dans email_backend .	airflow.utils.email.send_email_smtp
v2	smtp.smtp_host	Nom du serveur sortant utilisé pour l'adresse e-mail dans smtp_host .	localhost
v2	smtp.smtp_starttls	Le protocole TLS (Transport Layer Security) est utilisé pour chiffrer le courrier électronique sur Internet dans smtp_starttls .	False

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	smtp.smtp_ssl	Le protocole SSL (Secure Sockets Layer) est utilisé pour connecter le serveur et le client de messagerie dans smtp_ssl .	True
v2	smtp.smtp_port	Le port TCP (Transmission Control Protocol) désigné pour le serveur dans smtp_port .	587
v2	smtp.smtp_mail_from	Adresse e-mail sortante dans smtp_mail_from .	myemail@domain.com

Configurations des tâches

La liste suivante présente les configurations disponibles dans la liste déroulante pour les tâches Airflow sur Amazon MWAA.

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	core.default_task_retries	Nombre de tentatives d'exécution d'une tâche Apache Airflow dans default_task_retries .	3

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	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 présente les configurations du planificateur Apache Airflow disponibles dans la liste déroulante sur Amazon MWAA.

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	scheduler.catchup_by_default	Indique au planificateur de créer un DAG exécuté pour « rattraper » l'intervalle de temps spécifique indiqué dans catchup_by_default.	False
v2	scheduler.scheduler_zombie_task_threshold	Indique au planificateur s'il convient de marquer l'instance de tâche comme ayant échoué et de replanifier la tâche dans	300

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
		scheduler_zombie_task_threshold.	

Configurations des travailleurs

La liste suivante présente les configurations de travail Airflow disponibles dans la liste déroulante d'Amazon MWAA.

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	celery.worker_autoscale	<p>Nombre maximum et minimum de tâches pouvant être exécutées simultanément sur n'importe quel travailleur utilisant le Celery Executor dans worker_autoscale.</p> <p>Les valeurs doivent être séparées par des virgules dans l'ordre suivant : max_concurrency, min_concurrency</p>	16,12

Configurations du serveur Web

La liste suivante présente les configurations du serveur Web Airflow disponibles dans la liste déroulante d'Amazon MWAA.

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2	webserver.default_ui_timezone	<p>Le paramètre de date/heure par défaut de l'interface utilisateur d'Apache Airflow dans default_ui_timezone.</p> <div data-bbox="852 888 1162 1885"><p>Note</p><p>La définition de default_ui_timezone de cette option ne modifie pas le fuseau horaire dans lequel vos DAG sont programmés pour s'exécuter. Pour modifier le fuseau horaire de vos DAG, vous pouvez</p></div>	Amérique/New_York

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
		utiliser un plugin personnalisé. Pour plus d'informations, consultez the section called "Changer le fuseau horaire d'un DAG" .	

Configurations du déclencheur

La liste suivante présente les configurations du [déclencheur](#) Apache Airflow disponibles sur Amazon MWAA.

Apache Airflow v2

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
v2.7	<code>mwa.triggerer_enabled</code>	Utilisé pour activer et désactiver le déclencheur sur Amazon MWAA. Par défaut, cette valeur indique True. Si ce paramètre est défini sur False, Amazon MWAA ne lancera aucun processus	True

Version Airflow	Option de configuration du flux d'air	Description	Exemple de valeur
		déclencheur sur les planificateurs.	
v2.7	triggerer.default_capacity	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 planificateur est définie sur 60, 125, 250, 500, et 1000 pour les petites, moyennes et grandes instances, xlarge et 2xlarge, respectivement.	125

Exemples et exemple de code

Exemple de DAG

Vous pouvez utiliser le DAG suivant pour imprimer vos options de configuration `email_backend` Apache Airflow. Pour l'exécuter en réponse aux événements Amazon MWAA, copiez le code dans le dossier DAGS 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 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, consultez la section [Se connecter à l'aide des mots de passe des applications](#) dans le guide de référence de l'aide Gmail.

Airflow configuration options - optional [Info](#)

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

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

Quelle est la prochaine étape ?

- Découvrez comment télécharger votre dossier DAG dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

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 de la version `x.y.z` à `x.y.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, consultez la section [Migration vers un nouvel environnement Amazon MWAA](#) dans le guide de migration Amazon MWAA.

Au cours du processus de mise à niveau, Amazon MWAA capture un instantané des métadonnées de votre environnement, met à niveau 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.

Note

Vous ne pouvez pas rétrograder la version d'Apache Airflow pour votre environnement.

Avant de procéder à la mise à niveau, assurez-vous que vos DAG et autres ressources de flux de travail 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 les ressources de votre flux de travail](#)
- [Spécifiez la nouvelle version](#)

Améliorez les ressources de votre flux de travail

Chaque fois que vous modifiez la version d'Apache Airflow, assurez-vous de [référencer 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 peut entraîner un long processus de restauration vers la version précédente d'Apache Airflow par rapport à la version précédente des exigences.

Pour migrer les ressources de votre flux de travail

1. Créez un fork du [aws-mwaa-local-runner](#) référentiel et clonez une copie du runner local Amazon MWAA.
2. Accédez à la branche du `aws-mwaa-local-runner` référentiel qui correspond à la version vers laquelle vous effectuez la mise à niveau.

3. Utilisez l'outil CLI Amazon MWAA local Runner pour créer l'image Docker et exécuter Apache Airflow localement. Pour plus d'informations, consultez le [fichier README](#) du lanceur local dans le GitHub référentiel.
4. 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.
5. (Facultatif) Pour accélérer le processus de mise à niveau, [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.
6. Après avoir testé avec succès les ressources de votre flux de travail, copiez vos DAG et vos plug-ins dans le compartiment Amazon S3 de votre environnement. `requirements.txt`

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, 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.

Pour spécifier une nouvelle version à l'aide de la console

1. Ouvrez la [page Environnements](#) sur la console Amazon MWAA.
2. Dans la liste Environnements, choisissez l'environnement que vous souhaitez mettre à niveau.
3. Sur la page de l'environnement, choisissez Modifier pour modifier l'environnement.

4. Dans la section Détails de l'environnement, pour la version d'Airflow, choisissez le nouveau numéro de version d'Apache Airflow vers lequel vous souhaitez mettre à niveau 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. 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 réussi, le statut s'affichera `UPDATING`, `CREATING_SNAPSHOT` alors qu'Amazon MWAA capture une sauvegarde de vos métadonnées. Enfin, le statut reviendra d'abord à `UPDATING`, puis à une `AVAILABLE` fois la procédure terminée.

Si la mise à niveau de l'environnement échoue, l'état de votre environnement s'affichera `ROLLING_BACK`. Si la restauration est réussie, le statut s'affichera d'abord `UPDATE_FAILED`, indiquant que la mise à jour a échoué mais que l'environnement est disponible. Si la restauration échoue, le statut s'affichera `UNAVAILABLE`, indiquant que 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, similaire à vos DAG, 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 :

- Installer 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`, et `LD_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://mwa-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 identifiants de version sont des chaînes opaques Unicode, codées en UTF-8, prêtes pour les URL, dont la longueur ne dépasse pas 1 024 octets, par exemple. `3sL4kqtJ1cpXroDTDmJ+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 AWS Management Console et ouvrez la console Amazon S3 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. Sélectionnez Charger.

Le script apparaît 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 sous 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 *your-s3-bucket* par vos informations.

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

En cas de succès, Amazon S3 affiche le chemin URL de l'objet :

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

3. Utilisez la commande suivante pour récupérer le dernier ID de version du script.

```
$ aws s3api list-object-versions --bucket your-s3-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://*your-mwaa-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, consultez la section [Types de journaux Apache Airflow](#).

AWS Management Console

Pour consulter 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 afficher les 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 verrez le résultat de la commande imprimant la valeur de `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 consulter les journaux des utilisateurs et des serveurs 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, utilisez les instructions `MWAA_AIRFLOW_COMPONENT if` et `fi` 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 afin d'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 vérifier 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 peut 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'exemples de DAG.
- `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 intégrer d'autres AWS services dans.
- `AWS_DEFAULT_REGION`— Définit la AWS région par défaut 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 simultanément.
- `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 pour planifier les DAG.
- `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 la AWS région pour 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.

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 vérifie les répertoires PATH afin de trouver et d'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 non incluses dans la bibliothèque standard ou installées dans les répertoires du système. Utilisez cette variable pour ajouter vos modules et vos packages Python personnalisés et utilisez-les avec vos DAG.
- **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.

Utilisation des DAG sur Amazon MWAA

Pour exécuter des graphes acycliques dirigés (DAG) dans un environnement Amazon Managed Workflows pour 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 vos DAG et les fichiers de support sur la console Amazon MWAA. Amazon MWAA se charge de synchroniser les DAG entre les travailleurs, les planificateurs et le serveur Web. Ce guide explique comment ajouter ou mettre à jour vos DAG, et comment installer des plugins personnalisés et des dépendances Python dans un environnement Amazon MWAA.

Rubriques

- [Présentation du bucket Amazon S3](#)
- [Ajout ou mise à jour des DAG](#)
- [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, le AWS compte 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 des DAG, 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 [AmazonMWAA. FullConsoleAccess](#)

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 dans votre compartiment Amazon S3.

Ajout ou mise à jour des DAG

Les graphes acycliques dirigés (DAG) sont définis dans un fichier Python qui définit la structure du DAG sous forme de code. Vous pouvez utiliser laAWS CLI console Amazon S3 pour charger les DAG dans votre environnement. Cette page décrit les étapes à suivre pour ajouter ou mettre à jour des DAG Apache Airflow dans votre environnement Amazon Managed Workflows pour Apache Airflow à l'aide du `dags` dossier de votre compartiment Amazon S3.

Sections

- [Prérequis](#)
- [Fonctionnement](#)
- [Ce qui a changé dans la version 2](#)
- [Tester les DAG à l'aide de l'utilitaire Amazon MWAA CLI](#)
- [Chargement du code DAG dans Amazon S3](#)
- [Spécifier le chemin d'accès à votre dossier DAGS sur la console Amazon MWAA \(première fois\)](#)
- [Affichage des modifications sur votre interface utilisateur Apache Airflow](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

Vous aurez besoin des informations suivantes avant de pouvoir suivre les étapes de cette page.

- **Autorisations** : votreAWS compte doit avoir été autorisé par votre administrateur à accéder à la politique de contrôleFullConsoleAccess d'accès [AmazonMWAA](#) pour votre environnement. En outre, votre environnement Amazon MWAA doit être autorisé par votre [rôle d'exécution](#) à accéder auxAWS 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, veuillez consulter [the section called "Modes d'accès à Apache Airflow"](#).

- Configuration Amazon S3 : le compartiment [Amazon S3](#) utilisé pour stocker vos DAG, vos plug-ins personnalisés et vos dépendances `Pythonrequirements.txt` doit être configuré avec l'accès public bloqué et le contrôle de version activé. `plugins.zip`

Fonctionnement

Un graphe acyclique dirigé (DAG) est défini dans un seul fichier Python qui définit la structure du DAG sous forme de code. Il se compose des éléments suivants :

- Définition d'un [DAG](#).
- [Opérateurs](#) qui décrivent comment exécuter le DAG et les [tâches](#) à exécuter.
- [Relations avec 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 votre définition DAG dans le `dags` dossier de votre compartiment de stockage. Par exemple, le dossier DAG de votre compartiment de stockage peut ressembler à ceci :

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. Le temps nécessaire à l'apparition des nouveaux DAG dans l'interface utilisateur d'Apache Airflow est contrôlé par [`scheduler.dag_dir_list_interval`](#). Les modifications apportées aux DAG existants seront prises en compte lors de la prochaine [boucle de traitement des 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, veuillez consulter [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).

Ce qui a changé dans la version 2

- Nouveau : opérateurs, crochets et exécuteurs. Les instructions d'importation dans vos DAG et les plugins personnalisés que vous spécifiez dans un `MWAAplugins.zip` sur Amazon ont changé entre Apache Airflow v1 et Apache Airflow v2. Par exemple, `from airflow.contrib.hooks.aws_hook import AwsHook` dans Apache Airflow v1 est devenu `from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook` dans Apache Airflow v2. Pour en savoir plus, consultez la section [Référence de l'API Python](#) dans le guide de référence d'Apache Airflow.

Tester les DAG à l'aide de l'utilitaire Amazon MWAA CLI

- L'outil d'interface de ligne de commande (interface de commande) permet de répliquer 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. Cela vous permet d'exécuter un environnement Apache Airflow local pour développer et tester des DAG, des extensions personnalisées et des dépendances avant de les déployer sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-local-runner](#) on GitHub.

Chargement du code DAG dans Amazon S3

Vous pouvez utiliser la console Amazon S3 ou le AWS Command Line Interface (AWS CLI) pour charger le code DAG dans votre compartiment Amazon S3. Les étapes suivantes supposent que vous chargez le code (`.py`) vers un dossier nommé `dags` dans votre compartiment Amazon S3.

Utilisation du AWS CLI

L'AWS Command Line Interface (AWS CLI) est un outil à code source libre qui vous permet d'interagir avec les services AWS à l'aide des commandes du terminal de ligne de commande. Pour effectuer les étapes de cette page, vous avez besoin des éléments suivants :

- [AWS CLI— Installer la version 2](#).

- [AWS CLI— Configuration rapide avecaws configure](#).

Pour télécharger à l'aide duAWS 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 dossiers du compartiment Amazon S3 pour votre environnement.

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

3. La commande suivante permet de charger undag_def .py fichierdags dans un dossier.

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

Si aucun dossier nommédags n'existe déjà dans votre compartiment Amazon S3, cette commande crée ledags dossier et charge le fichier nommédag_def .py dans le nouveau dossier.

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur basée sur le Web vous permettant de créer et gérer les ressources Amazon S3. Les étapes suivantes supposent que vous avez un dossier DAGs nommédags.

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 sur la console Amazon S3.
4. Choisissez le dossier dags.
5. Sélectionnez Charger.
6. Choisissez Ajouter un fichier.
7. Sélectionnez la copie locale de votre fichierdag_def .py, puis choisissez Charger.

Spécifier le chemin d'accès à votre dossier DAGS sur la console Amazon MWAA (première fois)

Les étapes suivantes supposent que vous spécifiez le chemin d'accès à un dossier nommé dans votre compartiment Amazon S3dags.

1. Ouvrez la [page Environnements](#) sur la console Amazon MWAA.
2. Choisissez l'environnement dans lequel vous souhaitez exécuter les DAG.
3. Choisissez Edit (Modifier).
4. Dans le code DAG du volet Amazon S3, choisissez Browse S3 à côté du champ du dossier DAG.
5. Sélectionnez votredags dossier.
6. Choisissez Choisir.
7. Choisissez Suivant, Mettre à jour l'environnement.

Affichage des modifications sur votre interface utilisateur Apache Airflow

Connexion à Apache Airflow

Vous devez [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#) disposer des autorisations nécessaires pour accéder à votre AWS compte dans AWS Identity and Access Management (IAM) pour accéder à l'interface utilisateur d'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 DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on 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 `plugins.zip` fichier.

Table des matières

- [Prérequis](#)
- [Fonctionnement](#)
- [Ce qui a changé dans la version 2](#)
- [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](#)
 - [Utilisation de la 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 AWS compte doit avoir été autorisé par votre administrateur à accéder à la politique de contrôle d'FullConsoleAccess d'[AmazonMWAA](#) 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 à Apache Airflow”](#).
- **Configuration Amazon S3** — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAG, vos plugins personnalisés et vos dépendances Python `requirements.txt` doit être configuré avec l'accès public bloqué et le versionnage activé. `plugins.zip`

Fonctionnement

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.

Ce qui a changé dans la version 2

- **Nouveau** : opérateurs, crochets et exécuteurs. Les instructions d'importation dans vos DAG et les plugins personnalisés que vous spécifiez dans un MWAA `plugins.zip` sur Amazon ont changé entre Apache Airflow v1 et Apache Airflow v2. Par exemple, `from airflow.contrib.hooks.aws_hook import AwsHook` dans Apache Airflow v1, il a été remplacé par Apache Airflow v2. `from airflow.providers.amazon.aws.hooks.base_aws`

`import AwsBaseHook` Pour en savoir plus, consultez le manuel de [référence de l'API Python](#) dans le guide de référence d'Apache Airflow.

- Nouveau : Importations dans des plugins. L'importation d'opérateurs, de capteurs, de crochets ajoutés dans les plugins à l'aide de plugins n'airflow. `{operators, sensors, hooks}`. `<plugin_name>` est plus prise en charge. Ces extensions doivent être importées en tant que modules Python classiques. Dans les versions 2 et supérieures, l'approche recommandée consiste à les placer dans le répertoire DAG et à créer et utiliser un fichier `.airflowignore` pour les empêcher d'être analysés en tant que DAG. Pour en savoir plus, consultez les sections [Gestion des modules](#) et [Création d'un opérateur personnalisé](#) dans le guide de référence d'Apache Airflow.

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. Il vous permet d'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 Workers recherchent des plugins personnalisés lors du démarrage sur le conteneur AWS Fargate géré pour votre environnement sur `/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` incluez le `operators` répertoire en tant que répertoire de premier niveau, le répertoire sera extrait dans votre environnement `/usr/local/airflow/plugins/operators`
- 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 v1.10.12 ou Apache Airflow v2.0.2, Amazon MWAA limite le trafic sortant sur le serveur Web Apache Airflow et ne vous permet pas d'installer des plugins ni des dépendances Python directement sur le serveur Web. À partir d'Apache Airflow v2.2.2, 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 montrer comment structurer votre environnement de développement local.

Exemple d'utilisation d'une structure de répertoire plate dans plugins.zip

Apache Airflow v2

L'exemple suivant montre un `plugins.zip` fichier avec une structure de répertoire plate pour Apache Airflow v2.

Exemple répertoire plat avec PythonVirtualenvOperator plugins.zip

L'exemple suivant montre l'arborescence de niveau supérieur d'un fichier `plugins.zip` pour le plugin PythonVirtualenvOperator personnalisé dans [Création d'un plugin personnalisé pour Apache AirflowPythonVirtualenvOperator](#).

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

Exemple `plugins/virtual_python_plugin.py`

L'exemple suivant montre 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 v1

L'exemple suivant montre un `plugins.zip` fichier avec une structure de répertoire plate pour Apache Airflow v1.

Exemple répertoire plat avec PythonVirtualenvOperator `plugins.zip`

L'exemple suivant montre l'arborescence de niveau supérieur d'un fichier `plugins.zip` pour le plugin PythonVirtualenvOperator personnalisé dans [Création d'un plugin personnalisé pour Apache AirflowPythonVirtualenvOperator](#).

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

Exemple `plugins/virtual_python_plugin.py`

L'exemple suivant montre le plugin PythonVirtualenvOperator personnalisé.

```

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

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

class EnvVarPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'

```

Exemple d'utilisation d'une structure de répertoire imbriquée dans plugins.zip

Apache Airflow v2

L'exemple suivant montre un `plugins.zip` fichier avec des répertoires distincts pour `hooks`, `operators`, et un `sensors` répertoire pour Apache Airflow v2.

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 montre les instructions d'importation dans le DAG ([dossier DAG](#)) 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 montrent 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`.

Apache Airflow v1

L'exemple suivant montre un `plugins.zip` fichier avec des répertoires distincts pour `hooksoperators`, et un `sensors` répertoire pour Apache Airflow v1.10.12.

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 montre les instructions d'importation dans le DAG ([dossier DAG](#)) qui utilise les plugins personnalisés.

Exemple dags/your_dag.py

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

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

```
'retry_delay': timedelta(minutes=5),
}

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

    sens = MySensor(
        task_id='taskA'
    )

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

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

sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

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

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

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

Les exemples suivants montrent chacune des instructions d'importation nécessaires dans les fichiers de plug-in personnalisés.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base_hook import BaseHook

class MyHook(BaseHook):

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

Example sensors/my_airflow_sensor.py

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

class MySensor(BaseSensorOperator):

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

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

Example operators/my_airflow_operator.py

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

class MyOperator(BaseOperator):

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



```
self.my_field = my_field

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

Example operators/hello_operator.py

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

class HelloOperator(BaseOperator):

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

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

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. Cela vous permet d'exécuter un environnement Apache Airflow local pour développer et tester des DAG, des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-local-runner](#) on 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.

Utilisation de la AWS CLI

L'AWS Command Line Interface (AWS CLI) est un outil à code source libre qui vous permet d'interagir avec les services AWS à l'aide des commandes du terminal 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://YOUR_S3_BUCKET_NAME/plugins.zip
```

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur Web qui vous permet de créer et de 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 sur la console Amazon S3.
4. Sélectionnez 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 à côté du champ facultatif « Fichier de plugins ».
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 Python Virtualenv Operator](#).
- Découvrez comment créer un plugin personnalisé dans [Plug-in personnalisé avec Oracle](#).
- Découvrez comment créer un plugin personnalisé dans [the section called "Changer le fuseau horaire d'un DAG"](#).

Quelle est la prochaine étape ?

- Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de l'option [aws-mwaa-local-runner](#) on GitHub.

Installation des dépendances Python

Une dépendance Python est un package ou une distribution qui n'est pas 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 page 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

- [Prérequis](#)
- [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](#)
- [Afficher les journaux de votre requirements.txt](#)
- [Quelle est la prochaine étape ?](#)

Prérequis

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

- Autorisations — Votre AWS compte doit avoir été autorisé par votre administrateur à accéder à la politique de contrôle d'[FullConsoleaccès d'AmazonMWAA](#) 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 à Apache Airflow"](#).
- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAG, vos 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 de votre environnement.

Emplacement et limites de taille des dépendances Python

Le planificateur Apache Airflow et les Workers recherchent des plugins personnalisés lors du démarrage sur le conteneur AWS Fargate géré pour votre environnement sur `/usr/local/airflow/plugins`

- 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. Cela vous permet d'exécuter un environnement Apache Airflow local pour développer et tester des DAG, des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, consultez le [aws-mwaa-local-runner](#) on. 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 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, consultez le [aws-mwaa-local-runner](#) sur. GitHub
2. Consultez les suppléments du package Apache Airflow. Pour consulter la liste des packages installés pour Apache Airflow v2 sur Amazon MWAA, consultez Amazon MWAA [local runner 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 instruction. `--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, consultez 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 consulter les 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, consultez [the section called "Affichage des journaux de flux d'air"](#).

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 votre `requirements.txt`.

Apache Airflow v1

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, consultez le [aws-mwaa-local-runner](#) sur GitHub
2. Consultez les extras du package Airflow. [Consultez la liste des packages disponibles pour Apache Airflow v1.10.12 à l'adresse https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt](https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt).
3. Ajoutez le fichier de contraintes. Ajoutez le fichier de contraintes pour Apache Airflow v1.10.12 en haut de votre fichier `requirements.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.

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

4. Paquets Apache Airflow v1.10.12. Ajoutez les [extras du package Airflow](#) et la version Apache Airflow v1.10.12 (). `==` 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]==1.10.12
```

Exemple Shell sécurisé (SSH)

Le `requirements.txt` fichier d'exemple suivant installe SSH pour Apache Airflow v1.10.12.

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

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

Le `requirements.txt` fichier d'exemple suivant installe la bibliothèque Boto3 pour Apache Airflow v1.10.12.

```
boto3 == 1.17.4
```

[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 qui vous permet d'interagir avec les AWS services à l'aide de commandes dans votre shell 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 `requirements.txt` fichier dans un compartiment Amazon S3.

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

Utilisation de la console Amazon S3

La console Amazon S3 est une interface utilisateur Web qui vous permet de créer et de 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 sur la console Amazon S3.
4. Sélectionnez 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 Browse S3 à côté du champ « Fichier d'exigences » (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.

Afficher les journaux de votre **requirements.txt**

Vous pouvez consulter les journaux 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 afficher les journaux Apache Airflow sur la console Logs. CloudWatch

Pour consulter les 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. Vous devriez voir 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, un message d'erreur similaire au suivant peut s'afficher :

```
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 DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide du [aws-mwaa-local-runner](#) activé. 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 `requirements.txt` un `plugins.zip` fichier.

Table des matières

- [Prérequis](#)
- [Vue d'ensemble du versionnement](#)
- [Comment ça marche](#)
- [Supprimer un DAG sur Amazon S3](#)
- [Supprimer un fichier requirements.txt ou plugins.zip « actuel » d'un environnement](#)
- [Supprimer 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 AWS compte doit avoir été autorisé par votre administrateur à accéder à la politique de contrôle d'FullConsoleAccess d'[AmazonMWAA](#) 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, veuillez consulter [the section called “Modes d'accès à Apache Airflow”](#).

- Configuration Amazon S3 — Le compartiment [Amazon S3](#) utilisé pour stocker vos DAG, vos 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 (objet introuvable) /0k indiquant « Je ne suis pas là ». 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 « non actuelles » (précédentes) des fichiers, vous devez supprimer les versions du ou des fichiers, puis le marqueur de suppression correspondant à la version.

Comment ça marche

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 avec les plugins personnalisés et les 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 « Impossible de lire {version} la version du fichier {file} ».

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 sur la console Amazon S3.
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 du DAG. Une fois qu'un DAG a été exécuté dans Apache Airflow, il reste dans la liste des DAG 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.

Supprimer 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.

Supprimer 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 correspondant à la ou aux versions du fichier.

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, consultez la section [Récupération des versions d'objet depuis un compartiment activé pour la gestion](#) des versions dans le guide Amazon S3.
- Pour supprimer la version de l'objet, consultez 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, consultez [la section Gestion des marqueurs de suppression](#) dans le 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 votre 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

L'exemple suivant montre comment 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](#).
- En savoir plus sur les cycles de vie d'Amazon S3 dans [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 lié à votre AWS compte. Il garantit 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 pour 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 un 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 VPC Amazon pris en charge par 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 dans lequel plusieurs AWS comptes 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 apparaissent sous forme d'interfaces réseau élastiques (ENI) avec des adresses IP privées dans votre Amazon VPC. Une fois ces points de terminaison créés, tout trafic destiné à ces adresses IP 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 sortie pour l'ensemble du trafic (0.0.0.0/0).
 - 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.
- 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 ($0.0.0.0/0$).
 - La NACL doit avoir une règle sortante qui refuse tout le trafic ($0.0.0.0/0$).
 - Par exemple, [Exemples d'ACL \(recommandés\)](#).
- 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 renvoie la réponse aux instances.
 - Le périphérique NAT doit être attaché à un sous-réseau public. (Un périphérique NAT par sous-réseau public.)
 - Le périphérique NAT doit disposer d'une adresse IPv4 élastique (EIP) attachée à chaque sous-réseau public.
- 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 sortie pour l'ensemble du trafic ($0.0.0.0/0$).
 - 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 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`.
 - La NACL doit avoir une règle sortante qui refuse tout le trafic `()0.0.0.0/0`.
 - Par exemple, [Exemples d'ACL \(recommandés\)](#).
- 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 situés dans la même région et sur le même Amazon VPC que votre environnement AWS Amazon 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 bucket. 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 un 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 que vous devez choisir 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 AWS CloudFormation démarrage rapide qui crée simultanément l'infrastructure Amazon VPC, un compartiment Amazon S3 et un environnement Amazon MWAA. Pour en savoir plus, veuillez consulter la section [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 AWS 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 AWS 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 pour Apache Airflow et les configurations nécessaires pour ces composants.

Table des matières

- [Conditions](#)
- [Aperçu de la sécurité](#)
- [Listes de contrôle d'accès réseau \(listes ACL\)](#)
 - [Exemples d'ACL \(recommandés\)](#)
- [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 (ACL) 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 (ACL). 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 AWS Apache Airflow.](#)

Listes de contrôle d'accès réseau (listes ACL)

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 apatriote, 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.

Exemples d'ACL (recommandés)

L'exemple suivant montre les règles ACL entrantes et sortantes qui peuvent être utilisées pour un Amazon VPC pour un Amazon VPC avec routage public ou routage privé.

Numéro de règle	Type	Protocole	Plage de ports	Source	Autoriser/ Refuser
100	Tout le trafic IPv4	Tous	Tous	0.0.0.0/0	Autorisation
*	Tout le trafic IPv4	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 montre les règles du groupe de sécurité entrant qui autorisent tout le trafic d'un Amazon VPC 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 montre 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 montre 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 montre une politique de point de terminaison d'interface VPC pour un Amazon VPC avec 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 montre 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 sont stockés vos DAG et les fichiers de support.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-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) vous permet de connecter votre VPC en privé à des services hébergés sur celui-ci 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](#)
- [Affichage des points de terminaison VPC](#)
 - [Affichage des 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 apparaissent sous forme d'interfaces réseau élastiques (ENI) avec des adresses IP privées dans votre Amazon VPC. Une fois ces points de terminaison créés, tout trafic destiné à ces adresses IP 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 AWS comptes.
- 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 AWS Identity and Access Management (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 vous demandera de configurer l'autorisation à l'aide du rôle d'exécution de votre environnement. Pour ajouter des autorisations, consultez [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, veuillez consulter la section [the section called "Politiques de point de terminaison VPC \(routage privé uniquement\)"](#).

Affichage des points de terminaison VPC

Cette section explique comment afficher les points de terminaison VPC créés par Amazon MWAA et comment identifier les adresses IP privées de votre point de terminaison VPC Apache Airflow.

Affichage des points de terminaison VPC sur la console Amazon VPC

La section suivante décrit les étapes à suivre pour afficher le ou les points de terminaison VPC créés par Amazon MWAA, ainsi que tous les points de terminaison VPC que vous avez créés si vous utilisez un routage privé pour votre Amazon VPC.

Pour afficher le ou les points de terminaison du VPC

1. Ouvrez la [page Endpoints](#) sur la console Amazon VPC.
2. Utilisez le sélecteur de AWS région pour sélectionner votre région.
3. Vous devriez voir le ou les points de terminaison de l'interface VPC créés par Amazon MWAA, ainsi que tous les points de terminaison VPC que vous avez 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, consultez [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 devriez voir 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 devriez voir 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 devriez voir 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 votre nom d'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 devriez voir 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 qui vous permet d'accéder en toute sécurité à vos AWS ressources et aux ressources 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'instance Amazon EC2 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, consultez [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 peuvent 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 certificat SSL/TLS 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 certificat SSL/TLS AWS Certificate Manager, vous devez le télécharger dans la même AWS région que 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 (AWS PrivateLink) pour 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

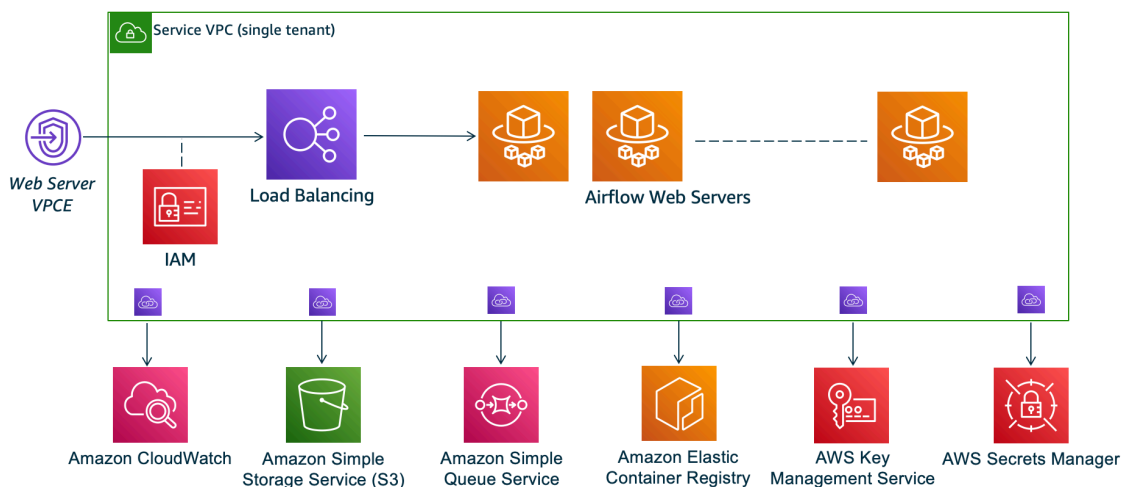
- [Tarification](#)
- [Réseau privé et routage privé](#)
- [Points de terminaison VPC \(obligatoires\)](#)
- [Connexion des points de terminaison VPC requis](#)
 - [Points de terminaison VPC requis pour les services AWS](#)
 - [Points de terminaison VPC requis pour Apache Airflow](#)
- [\(Facultatif\) Activez les adresses IP privées pour le point de terminaison de votre interface Amazon S3 VPC](#)
 - [Utilisation de la Route 53](#)
 - [VPC avec DNS personnalisé](#)

Tarification

- [AWS PrivateLink Tarification](#)

Réseau privé et routage privé

Private Web Server Option



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 à un serveur Web privé, vous devez emballer toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans votre `requirements.txt`. Pour obtenir des instructions sur l'emballage et l'installation de vos dépendances à l'aide de wheel, consultez [la section Gestion des dépendances à l'aide de Python Wheel](#).

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

Web server access

Private network (Recommended)

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

Public network (No additional setup)

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

- Routage privé. Un [Amazon VPC sans accès à Internet](#) limite le trafic réseau au sein du VPC. Cette page suppose 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 la même région et Amazon VPC que votre environnement AWS Amazon MWAA.

Points de terminaison VPC (obligatoires)

La section suivante indique 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.YOUR_REGION.s3
com.amazonaws.YOUR_REGION.monitoring
com.amazonaws.YOUR_REGION.ecr.dkr
com.amazonaws.YOUR_REGION.ecr.api
com.amazonaws.YOUR_REGION.logs
com.amazonaws.YOUR_REGION.sqs
com.amazonaws.YOUR_REGION.kms
com.amazonaws.YOUR_REGION.airflow.api
com.amazonaws.YOUR_REGION.airflow.env
com.amazonaws.YOUR_REGION.airflow.ops
```


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 avec routage privé.

Points de terminaison VPC requis pour les services AWS

La section suivante décrit les étapes à suivre pour 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. Utilisez le sélecteur de AWS région pour sélectionner votre région.
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 le ou 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 Amazon ECR :
 - a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.ecr.dkr**, 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 le ou 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 Amazon ECR :
- a. Choisissez Créer un point de terminaison.
 - b. Dans le champ de texte Filtrer par attributs ou rechercher par mot clé, tapez : **.ecr.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 le ou 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 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 le ou 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 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 le ou les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
8. 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 le ou les groupes de sécurité Amazon VPC de votre environnement.
 - g. Choisissez Accès complet dans la politique.
 - h. Choisissez Créer un point de terminaison.
9. 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 le ou 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 décrit 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. Utilisez le sélecteur de AWS région pour sélectionner votre région.
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 le ou 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 le ou 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 le ou 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.

VPC 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 instance EC2 exécutant un serveur DNS) en créant un enregistrement CNAME. Par exemple :

```
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 terminaux comporte deux principaux cas d'utilisation :

1. Cela signifie que vous pouvez créer des environnements Apache Airflow dans un Amazon VPC partagé lorsque vous utilisez [AWS Organizations](#) pour gérer AWS plusieurs comptes 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 la création de vos propres points de terminaison pour l'environnement, la base de données RDS pour PostgreSQL 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](#).

Création d'un environnement dans un Amazon VPC partagé

Si vous gérez plusieurs AWS comptes 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. [AWS Organizations](#)

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.

Prérequis

Avant de créer un environnement Amazon MWAA dans un VPC partagé, vous avez besoin des ressources suivantes :

- Un AWS compte, `Owner` à utiliser comme compte propriétaire de l'Amazon VPC.
- Unité [AWS Organizations](#) organisationnelle `MyOrganization` créée en tant que racine.
- Un deuxième AWS compte `Participant`, sous lequel `MyOrganization` sera utilisé 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

Créez d'abord un nouvel Amazon VPC que les comptes propriétaire et participant partageront :

1. Connectez-vous à la console en utilisant Owner, puis AWS 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
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. 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, consultez 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 ConsoleutilisateurOwner, 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 environnemen-tale
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 àCREATING, puis enfin àAVAILABLE. Lorsque c'est le casAVAILABLE, 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.

L'environnement dans le **CREATE_FAILED PENDING** post-statut

- 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 *account-number/security-group-id* :

Type	Protocole	Source type (Type de source)	Source
Tout le trafic	Tous	Tous	<i>123456789012/ sg-0909e8e8191 9</i>

Pour plus d'informations, voir [Responsabilités et autorisations pour les propriétaires et les participants](#)

PENDINGÉtat 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 bloquéPENDING, cela peut indiquer que le point de terminaison du serveur Web privé est manquant.

The Vpc Endpoint Service '*vpce-service-name*' does not existErreur reçue

Si le message d'erreur suivant s'affiche, 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'unAWS 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 DAG](#)
- [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'unAWS 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 unAWS Client VPN point de terminaison. Une fois configuré, un point de terminaison Client VPN 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 l'application [AWS Client VPNpour ordinateur de bureau](#).

Sections

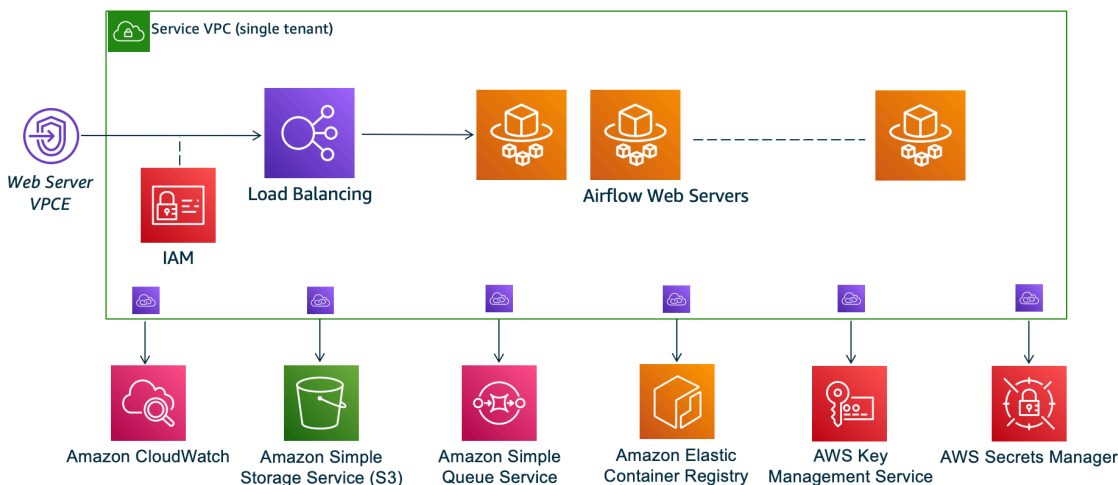
- [Réseau privé](#)
- [Cas d'utilisation](#)
- [Avant de commencer](#)
- [Objectifs](#)
- [\(Facultatif\) Première étape : identification de votre VPC, de vos règles CIDR et de vos sécurités VPC](#)
- [Étape 2 : créer les certificats de serveur et de client](#)
- [Troisième étape : enregistrer leAWS CloudFormation modèle localement](#)

- [Quatrième étape : créer laAWS CloudFormation pile Client VPN](#)
- [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 : Télécharger le fichier de configuration du point de terminaison Client VPN](#)
- [Huitième étape : Connect auAWS Client VPN](#)
- [Quelle est la prochaine étape ?](#)

Réseau privé

Ce didacticiel suppose que vous avez choisi le mode d'accès au réseau privé pour votre serveur Web Apache Airflow.

Private Web Server Option



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 été autorisés à accéder à la [politique IAM de votre environnement](#).

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

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

Web server access

Private network (Recommended)

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

Public network (No additional setup)

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

Cas d'utilisation

Vous pouvez utiliser ce didacticiel avant ou après avoir créé un environnement Amazon MWAA. Vous devez utiliser le même 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 modifier le mode d'accès de votre serveur Web Apache Airflow sur Réseau privé.

Avant de commencer

1. Vérifiez les autorisations des utilisateurs. Assurez-vous que votre compte dans AWS Identity and Access Management (IAM) dispose des autorisations suffisantes pour créer et gérer des ressources VPC.
2. Utilisez votre Amazon MWAA VPC. Ce didacticiel suppose que vous associez le Client VPN à un VPC existant. L'Amazon VPC doit se trouver dans la même AWS région qu'un environnement Amazon MWAA et disposer de deux sous-réseaux privés. Si vous n'avez pas créé d'Amazon VPC, utilisez le AWS 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 AWS CloudFormation modèle pour un Amazon VPC existant.
2. Générez les certificats et les clés de serveur et de client, puis charge le certificat et la clé de serveur AWS Certificate Manager dans la même AWS région qu'un environnement Amazon MWAA.

3. Téléchargez et modifiez un fichier de configuration du point de terminaison Client VPN pour votre Client VPN, et utilisez-le pour créer un profil VPN afin de vous connecter à l'aide du Client VPN pour ordinateur de bureau.

(Facultatif) Première étape : identification de votre VPC, de vos règles CIDR et de vos sécurités VPC

La section suivante explique comment trouver des identifiants pour 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 [page Vos Amazon VPC](#) sur la console Amazon VPC.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir la même AWS région qu'un environnement Amazon MWAA.
3. Choisissez votre Amazon VPC.
4. En supposant que les CIDR de vos 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 l'adresse CIDR de votre Amazon VPC est 10.192.0.0/16, l'adresse CIDR IPv4 du client que vous devez spécifier pour votre Client VPN est 10.192.0.0/22.

5. Enregistrez cette valeur CIDR et la valeur de votre identifiant 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 de vos groupes de sécurité, dont vous aurez besoin pour créer votre Client VPN.

Note

Vous utilisez peut-être plus d'un groupe de sécurité. Vous devrez spécifier tous les groupes de sécurité de votre VPC lors des étapes suivantes.

Pour identifier le ou les groupes de sécurité

1. Ouvrez la [page Groupes de sécurité](#) sur la console Amazon VPC.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir laAWS région.
3. Recherchez l'Amazon VPC dans l'ID 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.

Étape 2 : créer les certificats de serveur et de 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 easy-rsa OpenVPN pour générer les certificats et les clés de serveur et de client, puis charge les certificats dans ACM à l'aide de laAWS Command Line Interface (AWS CLI).

Pour créer les certificats clients

1. Suivez ces étapes rapides pour créer et télécharger les certificats vers ACM via l'AWS CLI[authenticatation et l'autorisation du client : authentification mutuelle](#).
2. Au cours de ces étapes, vous devez spécifier la mêmeAWS région qu'un environnement Amazon MWAA dans laAWS CLI commande lorsque vous chargez 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 de 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 laAWS CLI réponse pour les ARN du certificat serveur et du certificat client. Vous allez spécifier ces ARN dans votreAWS 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 de l'endroit où trouver ces informations d'identification :
 - a. Exemple sur macOS

Sur macOS, le contenu est enregistré dans `/Users/youruser/custom_folder`. Si vous listez tout le contenu (`ls -a`) de ce répertoire, vous devriez voir 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 et de la clé privée dans `client1.domain.tld.key`. `client1.domain.tld.crt` Vous allez ajouter ces valeurs au fichier de configuration de votre Client VPN.

Troisième étape : enregistrer leAWS CloudFormation modèle localement

La section suivante contient leAWS 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 sous `mwaa_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 CIDR IPv4 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.
- Vos identifiants de groupe de sécurité VPC sont 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 laAWS CloudFormation pile Client VPN

Pour créer le AWS Client VPN

1. Ouvrez la [console AWS CloudFormation](#).
2. Choisissez « Le modèle est prêt », puis « Importez un fichier modèle ».
3. Choisissez Choisir un fichier, puis sélectionnez votremwaa_vpn_client.yaml fichier.
- 4.
5. Choisissez Next, Next.
6. Sélectionnez l'accusé de réception, puis choisissez Créer une pile.

Étape 5 : associer des sous-réseaux à votre Client VPN

Pour associer des sous-réseaux privés àAWS Client VPN

1. Ouvrez la [console VPC Amazon](#).
2. Choisissez la page Client VPN Endpoints.
3. Sélectionnez votre Client VPN, puis choisissez l'onglet Associations, Associer.
4. Choisissez les options suivantes dans la liste déroulante :
 - Votre Amazon VPC dans VPC.
 - L'un de vos sous-réseaux privés dans Choisissez un sous-réseau à associer.
5. Choisissez Associate.

Note

L'association du VPC et le sous-réseau au VPN de client prend plusieurs minutes.

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, consultez les [règles d'autorisation](#) dans le guide du Client VPN.

Pour ajouter le CIDR au AWS Client VPN

1. Ouvrez la [console VPC Amazon](#).
2. Choisissez la page Client VPN Endpoints.
3. Sélectionnez votre Client VPN, puis choisissez 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. Par exemple :
5. Choisissez Ajouter une règle d'autorisation.

```
10.192.0.0/16
```

- Choisissez Autoriser l'accès à tous les utilisateurs dans Accorder l'accès à.
- Saisissez un nom descriptif dans Description.

Note

En fonction des composants réseau de votre Amazon VPC, vous devrez peut-être également intégrer cette règle d'autorisation à votre liste de contrôle d'accès réseau (NACL).

Étape 7 : Télécharger le fichier de configuration du point de terminaison Client VPN

Pour télécharger le fichier de configuration :

1. Suivez ces étapes rapides pour télécharger le fichier de configuration du Client VPN sur [Télécharger le fichier de configuration du point de terminaison Client VPN](#).
2. Au cours de ces étapes, vous êtes invité à ajouter une chaîne au nom DNS de votre point de terminaison Client VPN. Voici un exemple:

- Exemple nom DNS du point de terminaison

Si le nom DNS de votre point de terminaison Client VPN ressemble à ceci :

```
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 de 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 modifiez les répertoires en fonction de 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 dans `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 dans `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 : Connect au AWS Client VPN

Le formulaire de client AWS Client VPN est fourni gratuitement. Vous pouvez y connecter votre ordinateur directement au AWS Client VPN pour une expérience VPN de bout en bout.

Pour vous connecter au Client VPN

1. Téléchargez et installez le [AWS Client VPN pour ordinateur de bureau](#).
2. Ouvrez la AWS Client VPN.
3. Choisissez Fichier, Profils gérés dans le menu du client VPN.
4. Choisissez Ajouter un profil, puis choisissez `downloaded-client-config.ovpn`.
5. Entrez un nom descriptif dans Nom d'affichage.

6. Choisissez Ajouter un profil, Terminé.
7. Sélectionnez Connect (Connexion).

Une fois connecté au Client VPN, vous devez vous déconnecter des autres VPN pour consulter les 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 ?

- Apprenez à créer un environnement Amazon MWAA dans [Démarez avec Amazon Managed Workflows for Apache Airflow](#). Vous devez créer un environnement dans la même AWS région que le Client VPN et utiliser le même VPC, les mêmes sous-réseaux privés et le même groupe de sécurité que le Client VPN.

Tutoriel : Configuration de l'accès au réseau privé à l'aide d'un hôte Linux Bastion

Ce didacticiel explique les étapes de création d'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

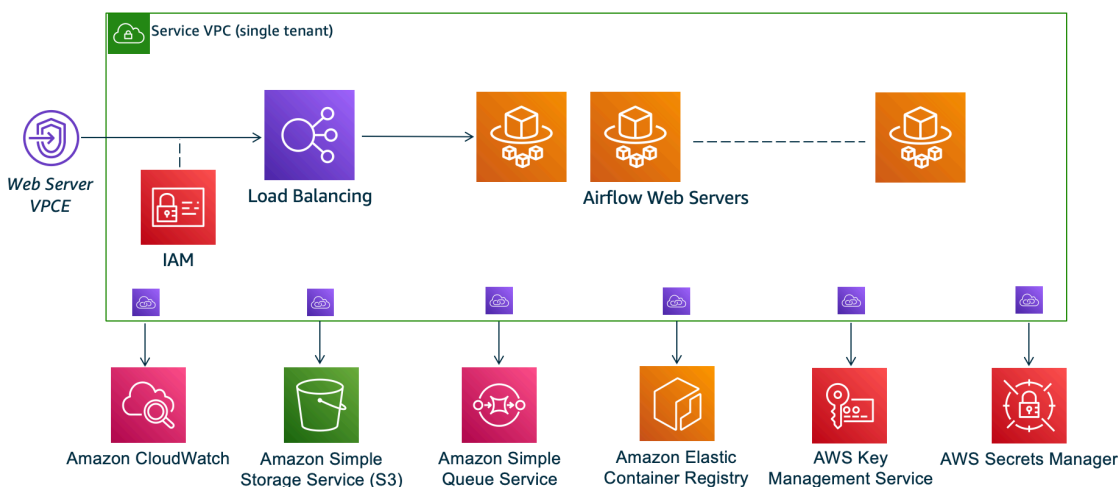
- [Réseau privé](#)
- [Cas d'utilisation](#)
- [Avant de commencer](#)
- [Objectifs](#)
- [Première étape : créer l'instance de bastion](#)

- [Deuxième étape : créer le tunnel SSH](#)
- [Troisième étape : configurer le groupe de sécurité Bastion en tant que règle entrante](#)
- [Quatrième étape : 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.

Private Web Server Option



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 à un serveur Web privé, vous devez emballer toutes vos dépendances dans une archive Python Wheel (.whl), puis y faire référence .whl dans votre `requirements.txt`. Pour obtenir des instructions sur l'emballage et l'installation de vos dépendances à l'aide de wheel, consultez [la section Gestion des dépendances à l'aide de Python Wheel](#).

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

Web server access

Private network (Recommended)

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

Public network (No additional setup)

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

Cas d'utilisation

Vous pouvez utiliser ce didacticiel après avoir créé un environnement Amazon MWAA. Vous devez utiliser le même Amazon VPC, le même groupe 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 AWS Identity and Access Management (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 SSH Amazon EC2 (.pem) dans la même région que votre environnement Amazon MWAA pour vous connecter aux serveurs virtuels. Si vous n'avez pas de clé SSH, consultez la section [Créer ou importer une paire de clés](#) dans le guide de l'utilisateur Amazon EC2.

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 [AWS 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'afficher 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 [AWS CloudFormation modèle pour un VPC existant](#) sur la AWS CloudFormation console.

Pour créer l'hôte Linux Bastion

1. Ouvrez la page de [démarrage rapide du déploiement](#) sur la AWS CloudFormation console.
2. Utilisez le sélecteur de région dans la barre de navigation pour choisir la même AWS région que votre environnement Amazon MWAA.
3. Choisissez Suivant.
4. Tapez 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 AWS 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 de configuration Amazon EC2, 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, consultez [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 via la ligne de commande

1. Ouvrez la page [Instances](#) sur la console Amazon EC2.
2. Choisissez un type d'instance.
3. Copiez l'adresse dans le DNS IPv4 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 console Amazon EC2.
2. Choisissez un type d'instance.
3. Copiez l'adresse dans le DNS IPv4 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'outil [AWSSupport-TroubleShootSSH](#) 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.

Quatrième étape : 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 Web Amazon ResourceManagerEC2 en transférant le trafic sur le port local 8157.

1. Ouvrez une nouvelle fenêtre d'invite de commandes.
2. Tapez la commande suivante pour ouvrir un tunnel SSH.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-  
vpce.YOUR_REGION.airflow.amazonaws.com:443  
ubuntu@YOUR_PUBLIC_IPV4_DNS.YOUR_REGION.compute.amazonaws.com
```

-L signifie l'utilisation de la redirection de port local qui vous permet de spécifier un port local utilisé pour transférer les données vers le port distant identifié sur le serveur Web local du nœud.

3. Tapez `http://localhost:8157/` dans votre navigateur.

Note

Il se peut que vous deviez utiliser `https://localhost:8157/`.

Deuxième option : Proxies via la ligne de commande

La plupart des navigateurs Web vous permettent de configurer des proxys via une ligne de commande ou 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.YOUR_REGION.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 les URL qui correspondent aux 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. Cliquez FoxyProxy sur l'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 d'échantillon dans ***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 `mwa-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 du code DAG dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

Tutoriel : Restreindre l'accès d'un utilisateur Amazon MWAA à un sous-ensemble de DAG

[Amazon MWAA gère l'accès à votre environnement en mappant vos principaux IAM à un ou plusieurs rôles par défaut d'Apache Airflow.](#) Le didacticiel suivant montre comment vous pouvez empêcher les utilisateurs individuels d'Amazon MWAA de consulter et d'interagir uniquement avec un DAG spécifique ou un ensemble de DAG.

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

- [Prérequis](#)
- [Étape 1 : fournissez un accès au serveur Web Amazon MWAA à votre principal IAM avec le rôle Public Apache Airflow 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](#)

Prérequis

Pour suivre les étapes de ce didacticiel, vous aurez besoin des éléments suivants :

- Un [environnement Amazon MWAA avec plusieurs](#) DAG
- Un principal IAM, Admin avec des [AdministratorAccess](#) autorisations, et un utilisateur IAM `MWAAUser1`, en tant que principal pour lequel vous pouvez limiter l'accès au DAG. Pour plus d'informations sur les rôles d'administrateur, voir [Fonction 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 **Public** Apache Airflow par défaut.

Pour accorder une autorisation à l'aide du AWS Management Console

1. Connectez-vous à votre AWS compte 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 ayant le rôle Public Apache Airflow par défaut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "airflow:CreateWebLoginToken",
      "Resource": [
        "arn:aws:airflow:YOUR_REGION:YOUR_ACCOUNT_ID: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 afficher les rôles Apache Airflow par défaut.
3. Dans la liste des rôles, sélectionnez Utilisateur, puis en haut de la page, sélectionnez Actions pour ouvrir le menu déroulant. Choisissez Copier le rôle et confirmez OK

Note

Copiez les rôles Ops ou Viewer pour accorder plus ou moins d'accès, respectivement.

4. Localisez 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, saisissez 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 de DAG auquel 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 devriez désormais avoir un nouveau rôle qui limite l'accès à un sous-ensemble de DAG disponibles dans votre environnement Amazon MWAA. Vous pouvez désormais attribuer ce rôle à n'importe quel utilisateur 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 `MWAAUser1`, exécutez la commande CLI suivante pour récupérer l'URL du serveur Web de votre environnement.

```
$ aws mwaas get-environment --name YOUR_ENVIRONMENT_NAME | jq  
' .Environment.WebserverUrl '
```

En cas de réussite, vous verrez le résultat suivant :

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. Une fois `MWAAUser1` connecté au AWS Management Console, ouvrez une nouvelle fenêtre de navigateur et accédez à l'URL suivante. Remplacez `Webserver-URL` par vos informations.

```
https://<Webserver-URL>/home
```

En cas de succès, vous verrez une page `Forbidden` d'erreur car il n'y a 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 voir leurs DAG.

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 voir des exemples de politiques IAM JSON que vous pouvez utiliser pour les utilisateurs de votre environnement, consultez [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 gérez [AWS Organizations](#) plusieurs AWS comptes partageant des ressources, Amazon MWAA vous permet de créer et de gérer vos propres points de terminaison Amazon VPC. 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](#)
- [Créer 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 AWS 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 AWS 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 mwa-vpce-lambda \  
--zip-file file://mwa-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 "mwa-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 "mwa-lambda-rule" \  
--targets "Id"="1", "Arn"="arn:aws::lambda:region:123456789012:function:mwa-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 Amazon S3 et un dossier DAG, 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 par Internet).
 - c. Pour les groupes de sécurité, choisissez le groupe de sécurité que vous avez créé avec AWS 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, notamment des DAG et des plugins personnalisés, que vous pouvez utiliser dans un environnement Amazon Managed Workflows pour Apache Airflow. Pour d'autres exemples d'utilisation d'Apache Airflow avec AWS des services, consultez le [dags](#) répertoire dans le référentiel Apache Airflow GitHub .

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 dansAWS Secrets Managerpour une connexion Apache Airflow Snowflake](#)
- [Utilisation d'un DAG pour écrire des métriques personnalisées dansCloudWatch](#)
- [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 dansAWS Secrets Managerpour une variable Apache Airflow](#)
- [Utilisation d'une clé secrète dansAWS Secrets Managerpour une connexion Apache Airflow](#)
- [Création d'un plugin personnalisé avec Oracle](#)
- [Création d'un plugin personnalisé qui génère des variables d'environnement d'exécution](#)
- [Modifier le fuseau horaire d'un DAG sur Amazon MWAA](#)
- [Rafraîchir CodeArtifact jeton](#)
- [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#)
- [Création d'un plugin personnalisé pour Apache AirflowPythonVirtualenvOperator](#)
- [Invoquer des DAG avec une fonction Lambda](#)
- [Invocation de DAG dans différents environnements Amazon MAAA](#)
- [Utilisation d'Amazon MWAA avec Amazon RDS pour Microsoft SQL Server](#)
- [Utilisation avec Amazon EMR avec Amazon EMR](#)
- [Utilisation d'Amazon MAAA avec Amazon EKS](#)
- [Connexion à Amazon ECS à l'aide duECSOperator](#)

- [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 l'interface de ligne de commande 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 de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Autorisations

Votre AWS le compte doit accéder au Amazon MWAA Airflow CLI Access politique. Pour en savoir plus, consultez [Politique de la CLI Apache Airflow : Amazon MWAA Airflow CLI Access](#).

Dépendances

- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise [Installation de base d'Apache Airflow v2](#) sur votre environnement.

Exemple de code

L'exemple de code suivant nécessite trois entrées : le nom de votre environnement Amazon MWA (dans `mwa_env`), la région AWS de votre environnement (dans `aws_region`) et le fichier local qui contient les variables que vous souhaitez importer (dans `var_file`).

```
import boto3
import json
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwa_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 MWA environment -v variable file location and filename -r
aws region')
    else:
        for opt, arg in opts:
            if opt in ("-e"):
                mwa_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))
    mwa_env_name = "{}".format(mwa_env)

    client = boto3.client('mwa')
    mwa_cli_token = client.create_cli_token(
        Name=mwa_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 charger le code DAG de cet exemple vers le dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

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 instance Amazon EC2 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 ou version ultérieure en [Python 3.10](#).

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 instance Amazon EC2 et d'une instance située .pem dans la même région que votre environnement Amazon MWAA. Si vous n'avez pas de clé, consultez la section [Créer ou importer une paire de clés](#) dans le guide de l'utilisateur Amazon EC2.

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code présenté sur cette page.

Prérequis

Ajoutez le paramètre suivant `requirements.txt` pour 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 verrez apparaître 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://your-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 supérieure 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. Dans le champ Identifiant 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'instance Amazon EC2 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 instance Amazon EC2. 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 instance Amazon EC2 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. Par exemple :

```
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 verrez 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 dansAWS Secrets Managerpour une connexion Apache Airflow Snowflake

Les exemples d'appels suivantsAWS Secrets Managerpour obtenir une clé secrète pour une connexion Apache Airflow Snowflake sur Amazon Managed Workflows pour Apache Airflow. Cela suppose que vous avez suivi les étapes décrites dans[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 de cette page avecApache Airflow v2 et versions ultérieuresdans[Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Le backend de 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, comme indiqué 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, consultez [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 commandes, 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 `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 charger le code DAG de cet exemple vers le dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

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` pour 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 mesures personnalisées au niveau du système d'exploitation vous offrent une visibilité supplémentaire sur la façon dont les employés de votre environnement utilisent les ressources telles que la mémoire virtuelle et le processeur. Vous pouvez utiliser ces informations pour sélectionner [classe d'environnement](#) qui convient le mieux à votre charge de travail.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous devez disposer des éléments suivants :

- Un [Environnement Amazon MWA](#).

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Dépendances

- Aucune dépendance supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Exemple de code

1. Dans votre invite de commandes, accédez au dossier 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 `dag-custom-metrics.py`. Remplacez `MWAA-ENV-NAME` avec 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("MWAA_ENV_NAME")
    value_number=float(value)
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    print('writing value',value_number,'to metric',name)
    response = client.put_metric_data(
        Namespace='MWAA-Custom',
        MetricData=[
            {
                'MetricName': name,
                'Dimensions': [
                    {
                        'Name': 'Environment',
                        'Value': environment_name
                    },
                    {
                        'Name': 'Category',
                        'Value': cat
                    },
                    {
                        'Name': 'Host',
                        'Value': ip_address
                    },
                ],
                'Timestamp': datetime.now(),
                'Value': value_number,
                'Unit': unit
            },
        ]
    )
```

```
print(response)
return response

def python_fn(**kwargs):
    client = boto3.client('cloudwatch')

    cpu_stats = psutil.cpu_stats()
    print('cpu_stats', cpu_stats)

    virtual = psutil.virtual_memory()
    cpu_times_percent = psutil.cpu_times_percent(interval=0)

    publish_metric(client=client, name='virtual_memory_total',
cat='virtual_memory', value=virtual.total, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_available',
cat='virtual_memory', value=virtual.available, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_used', cat='virtual_memory',
value=virtual.used, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_free', cat='virtual_memory',
value=virtual.free, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_active',
cat='virtual_memory', value=virtual.active, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_inactive',
cat='virtual_memory', value=virtual.inactive, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_percent',
cat='virtual_memory', value=virtual.percent, unit='Percent')

    publish_metric(client=client, name='cpu_times_percent_user',
cat='cpu_times_percent', value=cpu_times_percent.user, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_system',
cat='cpu_times_percent', value=cpu_times_percent.system, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_idle',
cat='cpu_times_percent', value=cpu_times_percent.idle, unit='Percent')

    return "OK"

with DAG(dag_id=os.path.basename(__file__).replace(".py", ""),
schedule_interval='*/5 * * * *', catchup=False, start_date=days_ago(1)) as dag:
    t = PythonOperator(task_id="memory_test", python_callable=python_fn,
provide_context=True)
```

3. Exécutez la commande suivanteAWS CLIcommande permettant de copier le DAG dans le bucket de votre environnement, puis de déclencher 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 devriez voir 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](#)
- [Prérequis](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 ou version ultérieure en [Python 3.10](#).

Prérequis

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. Le code utilise l'[installation de base d'Apache Airflow v2](#) sur votre environnement.

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 au cours des

sept derniers jours. Pour ajuster la date à laquelle les entrées ont été supprimées, définissez `MAX_AGE_IN_DAYS` une valeur différente.

Apache Airflow v2

```
from airflow import settings
from airflow.utils.dates import days_ago
from airflow.models import DagTag, DagModel, DagRun, ImportError, Log, SlaMiss,
    RenderedTaskInstanceFields, TaskInstance, TaskReschedule, XCom
from airflow.decorators import dag, task
from airflow.utils.dates import days_ago
from time import sleep

from airflow.version import version
major_version, minor_version = int(version.split('.')[0]), int(version.split('.')[1])
if major_version >= 2 and minor_version >= 6:
    from airflow.jobs.job import Job
else:
    # The BaseJob class was renamed as of Apache Airflow v2.6
    from airflow.jobs.base_job import BaseJob as Job

# Delete entries for the past seven days. Adjust MAX_AGE_IN_DAYS to set how far back
# this DAG cleans the database.
MAX_AGE_IN_DAYS = 7
MIN_AGE_IN_DAYS = 0
DECREMENT = -7

# This is a list of (table, time) tuples.
# table = the table to clean in the metadata database
# time = the column in the table associated to the timestamp of an entry
# or None if not applicable.
TABLES_TO_CLEAN = [[Job, Job.latest_heartbeat],
    [TaskInstance, TaskInstance.execution_date],
    [TaskReschedule, TaskReschedule.execution_date],
    [DagTag, None],
    [DagModel, DagModel.last_parsed_time],
    [DagRun, DagRun.execution_date],
    [ImportError, ImportError.timestamp],
    [Log, Log.dttm],
    [SlaMiss, SlaMiss.execution_date],
    [RenderedTaskInstanceFields, RenderedTaskInstanceFields.execution_date],
    [XCom, XCom.execution_date],
```



```
]

@task()
def cleanup_db_fn(x):
    session = settings.Session()

    if x[1]:
        for oldest_days_ago in range(MAX_AGE_IN_DAYS, MIN_AGE_IN_DAYS, DECREMENT):
            earliest_days_ago = max(oldest_days_ago + DECREMENT, MIN_AGE_IN_DAYS)
            print(f"deleting {str(x[0])} entries between {earliest_days_ago} and
{oldest_days_ago} days old...")
            earliest_date = days_ago(earliest_days_ago)
            oldest_date = days_ago(oldest_days_ago)
            query = session.query(x[0]).filter(x[1] >= oldest_date).filter(x[1] <=
earliest_date)
            query.delete(synchronize_session= False)
            session.commit()
            sleep(5)
        else:
            # No time column specified for the table. Delete all entries
            print("deleting", str(x[0]), "...")
            query = session.query(x[0])
            query.delete(synchronize_session= False)
            session.commit()

    session.close()

@dag(
    dag_id="cleanup_db",
    schedule_interval="@weekly",
    start_date=days_ago(7),
    catchup=False,
    is_paused_upon_creation=False
)

def clean_db_dag_fn():
    t_last=None
    for x in TABLES_TO_CLEAN:
        t=cleanup_db_fn(x)
        if t_last:
            t_last >> t
        t_last = t
```

```
clean_db_dag = clean_db_dag_fn()
```

Exportation des métadonnées de l'environnement vers des fichiers CSV sur Amazon S3

L'exemple de code suivant montre comment 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 afin d'inspecter les données localement, de les archiver dans un stockage d'objets ou de 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 les analyses futures.

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 du DAG.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 ou version ultérieure en [Python 3.10](#).

Prérequis

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.

Autorisations

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:::your-new-export-bucket"
}
```

Cette politique limite l'accès en écriture uniquement à *your-new-export-bucket*.

Prérequis

- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise l'[installation de base d'Apache Airflow v2](#) sur votre environnement.

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 `metadata_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.execution_date],
    [TaskInstance, TaskInstance.execution_date],
]

@task()
def export_db_task(**kwargs):
    session = settings.Session()
    print("session: ", str(session))

    oldest_date = days_ago(MAX_AGE_IN_DAYS)
    print("oldest_date: ", oldest_date)

    s3 = boto3.client('s3')

    for x in OBJECTS_TO_EXPORT:
        query = session.query(x[0]).filter(x[1] >= days_ago(MAX_AGE_IN_DAYS))
        print("type", type(query))
        allrows=query.all()
        name=re.sub("[<>]", "", str(x[0]))
        print(name,": ", str(allrows))

        if len(allrows) > 0:
            outfileStr=""
            f = StringIO(outfileStr)
            w = csv.DictWriter(f, vars(allrows[0]).keys())
```

```

        w.writeheader()
        for y in allrows:
            w.writerow(vars(y))
        outkey = S3_KEY.format(name[6:])
        s3.put_object(Bucket=S3_BUCKET, Key=outkey, Body=f.getvalue())

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=days_ago(1),
)
def export_db():
    t = export_db_task()

metadb_to_s3_test = export_db()

```

3. 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 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 dans 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 pour Apache Airflow. Cela suppose que vous avez suivi les étapes décrites dans [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

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Le backend de 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 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, comme indiqué 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 v1, aucune dépendance supplémentaire n'est requise. Le code utilise le [Installation de base d'Apache Airflow v1](#) sur votre environnement.
- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise le [Installation de base d'Apache Airflow v2](#) sur votre environnement.

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 commandes, 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 `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 charger le code DAG de cet exemple vers le dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

Utilisation d'une clé secrète dans 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 pour Apache Airflow. Cela suppose que vous avez suivi les étapes décrites dans [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

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Le backend de 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, comme indiqué 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 v1, aucune dépendance supplémentaire n'est requise. Le code utilise [Installation de base d'Apache Airflow v1](#) sur votre environnement.

- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise [Installation de base d'Apache Airflow v2](#) sur votre environnement.

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui appelle Secrets Manager pour obtenir le secret.

Apache Airflow v2

1. Dans votre invite de commandes, 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 `secrets-manager.py`.

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

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

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

from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os
```

```
### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}

### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
    dag_id=os.path.basename(__file__).replace(".py", ""),
    default_args=default_args,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Apache Airflow v1

1. Dans votre invite de commandes, 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 `secrets-manager.py`.

```
from airflow import DAG, settings, secrets
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from airflow.contrib.hooks.aws_hook import AwsHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
}

### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsHook()
    client = hook.get_client_type('secretsmanager')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
    dag_id=os.path.basename(__file__).replace(".py", ""),
    default_args=default_args,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval=None
```

```
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Quelle est la prochaine étape ?

- Découvrez comment charger le code DAG de cet exemple vers le DAG dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).

Création d'un plugin personnalisé avec Oracle

L'exemple suivant explique comment créer un plug-in personnalisé à l'aide d'Oracle pour Amazon MWAA. Il peut être combiné à 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](#)
 - [Plugin personnalisé](#)
 - [Plugins.zip](#)
- [Options de configuration du débit d'air](#)
- [Quelle est la prochaine étape ?](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).

- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).
- Travailleur journalisation activée à n'importe quel niveau de journalisation, CRITICAL ou plus, 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 “Affichage des journaux de flux d'air”](#)

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, consultez [Installation des dépendances Python](#).

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

Apache Airflow v1

```
cx_Oracle==8.1.0
apache-airflow[oracle]==1.10.12
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui testera le plug-in personnalisé.

1. Dans votre invite de commandes, 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 `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 explique comment télécharger les dépendances, créer le plugin personnalisé et le fichier `plugins.zip`.

Dépendances de téléchargement

Amazon MWAA va extraire le contenu du fichier `plugins.zip` dans `/usr/local/airflow/plugins` sur chaque planificateur Amazon MWAA et chaque conteneur de travail. Ceci est utilisé pour ajouter des fichiers binaires à votre environnement. Les étapes suivantes décrivent comment assembler les fichiers nécessaires au plug-in 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. Par exemple :

```
docker pull amazonlinux
docker run -it amazonlinux:latest /bin/bash
```

Votre invite de commande doit appeler une ligne de commande bash. Par exemple :

```
bash-4.2#
```

2. Installez la fonction d'E/S asynchrones native pour Linux (libaio).

```
yum -y install libaio
```

3. Laissez cette fenêtre ouverte pour les étapes suivantes. Nous allons copier 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. Par exemple :

```
sudo yum install unzip
```

2. Créez un répertoire oracle_plugin. Par exemple :

```
mkdir oracle_plugin
cd oracle_plugin
```

3. Utilisez la commande curl suivante pour télécharger [instantclient-basic-linux.x64-18.5.0.0.0dbru.zip](#) à partir de [Téléchargements d'Oracle Instant Client 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. Par exemple :


```
unzip *.zip
```

Extraire des fichiers depuis Docker

1. Dans une nouvelle invite de commande, affichez et notez l'ID de votre conteneur Docker. Par exemple :

```
docker container ls
```

Votre invite de commande doit renvoyer tous les conteneurs et leurs identifiants. Par exemple :

```
debc16fd6970
```

2. Dans votre `oracle_plugin` répertoire, extrayez les `lib64/libaio.so.1`, `lib64/libaio.so.1.0.0`, `lib64/libaio.so.1.0.1` fichiers vers le `localinstantclient_18_5` dossier. Par exemple :

```
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/
```

Plugin personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier `plugins` au démarrage. Ceci est utilisé pour définir et modifier des variables d'environnement. Les étapes suivantes décrivent l'exemple de code pour le plug-in 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 montrent comment créer `plugins.zip`. Le contenu de cet exemple peut être combiné avec vos autres plugins et binaires en un seul `plugins.zip` dossier.

Compressez le contenu du répertoire du plugin

1. Dans votre invite de commandes, accédez au `oracle_plugin` répertoire. Par exemple :

```
cd oracle_plugin
```

2. Zippez le `instantclient_18_5` répertoire dans le fichier `plugins.zip`. Par exemple :

```
zip -r ../plugins.zip ./
```

3. Vous devriez voir ce qui suit dans votre invite de commande :

```
oracle_plugin$ ls
client.zip  instantclient_18_5
```

4. Supprimer le `client.zip` dossier. Par exemple :

```
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`. Par exemple :

```
zip plugins.zip env_var_plugin_oracle.py
```

2. Votre fichier `plugins.zip` doit maintenant inclure les éléments suivants :

```
env_var_plugin_oracle.py
instantclient_18_5/
```

Options de configuration du débit d'air

Si vous utilisez Apache Airflow v2, ajoutez `core.lazy_load_plugins : False` en tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, consultez [Utiliser les options de configuration pour charger des plugins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment charger le code DAG de cet exemple vers le `dags` dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- En savoir plus sur la façon de télécharger le `plugins.zip` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Création d'un plugin personnalisé qui génère des variables d'environnement d'exécution

L'exemple suivant vous explique les étapes à suivre pour créer un plugin personnalisé qui génère des variables d'environnement lors de l'exécution dans un environnement Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Plug-in personnalisé](#)
- [Plugins.zip](#)
- [Options de configuration du débit d'air](#)
- [Quelle est la prochaine étape ?](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous avez besoin des éléments suivants :

- Un [environnement Amazon MWAA](#).

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Prérequis

- Pour utiliser cet exemple de code avec Apache Airflow v1, aucune dépendance supplémentaire n'est requise. Le code utilise l'[installation de base d'Apache Airflow v1](#) sur votre environnement.

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 des variables d'environnement. Les étapes suivantes décrivent l'exemple de code pour le plugin personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel sont stockés vos plugins. Par exemple :

```
cd plugins
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le `leenv_var_plugin.py` sous.

```
from airflow.plugins_manager import AirflowPlugin
import os

os.environ["PATH"] = os.getenv("PATH") + " :/usr/local/airflow/.local/lib/python3.7/
site-packages"
```

```
os.environ["JAVA_HOME"]="/usr/lib/jvm/java-1.8.0-  
openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64"  
  
class EnvVarPlugin(AirflowPlugin):  
    name = 'env_var_plugin'
```

Plugins.zip

Les étapes suivantes montrent 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. Par exemple :

```
cd plugins
```

2. Comprimez le contenu dans votre `plugins` dossier.

```
zip -r ../plugins.zip ./
```

Options de configuration du débit d'air

Si vous utilisez Apache Airflow v2, ajoutez `core.lazy_load_plugins : False` comme option de configuration d'Apache Airflow. Pour en savoir plus, consultez [Utiliser les options de configuration pour charger des plugins dans 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` fichier dans cet exemple vers votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment charger le code DAG de cet exemple vers le `dags` dossier de votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- Découvrez comment charger le `plugins.zip` fichier dans cet exemple vers 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 montrent comment modifier le fuseau horaire dans lequel Amazon MWAA exécute vos DAG avec [Pendule](#). Cette rubrique explique éventuellement comment créer un plug-in personnalisé pour modifier le fuseau horaire des journaux Apache Airflow de votre environnement.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow](#)
- [Création d'un plugins.zip](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Créez un plugin pour modifier le fuseau horaire dans les journaux Airflow

Apache Airflow exécutera les fichiers Python du répertoire `plugins` au démarrage. Avec le plugin suivant, vous pouvez remplacer le fuseau horaire de l'exécuteur, ce qui modifie le fuseau horaire dans lequel Apache Airflow écrit les journaux.

1. Créez un répertoire nommé `plugins` pour votre plugin personnalisé, puis accédez au répertoire. Par exemple :

```
$ mkdir plugins
$ cd plugins
```

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

Création d'un `plugins.zip`

Les étapes suivantes montrent comment créer `plugins.zip`. Le contenu de cet exemple peut être combiné avec d'autres plugins et binaires en un seul `plugins.zip` dossier.

1. Dans votre invite de commandes, 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. Uploader `plugins.zip` vers 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 allons utiliser une bibliothèque appelée [Pendule](#), une bibliothèque Python permettant de travailler avec la fonction `datetime` adaptée au fuseau horaire.

1. Dans votre invite de commande, accédez au répertoire dans lequel vos DAG sont stockés. Par exemple :

```
$ cd dags
```

2. Copiez le contenu de l'exemple suivant et enregistrez-le sous `tz-aware-dag.py`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
# Import the Pendulum library.
import pendulum

# Instantiate Pendulum and set your timezone.
local_tz = pendulum.timezone("America/Los_Angeles")

with DAG(
    dag_id="tz_test",
    schedule_interval="0 12 * * *",
    catchup=False,
    start_date=datetime(2022, 1, 1, tzinfo=local_tz)
) as dag:
    bash_operator_task = BashOperator(
        task_id="tz_aware_task",
        dag=dag,
        bash_command="date"
    )
```


3. Exécutez la commande suivanteAWS CLIcommande permettant de copier le DAG dans le bucket de votre environnement, puis de déclencher 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 dutz_aware_taskdans letz_testJOUR :

```
[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 ?

- En savoir plus sur la façon de télécharger le `plugins.zip` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Rafraîchir CodeArtifact jeton

Si vous utilisez CodeArtifact pour installer les dépendances Python, Amazon MWAA nécessite un jeton actif. Pour autoriser Amazon MWAA à accéder à CodeArtifact dépôt au moment de l'exécution, vous pouvez utiliser un [script de démarrage](#) et réglez le `PIP_EXTRA_INDEX_URL` avec le jeton.

La rubrique suivante décrit comment créer un script de démarrage qui utilise [get_authorization_token](#) CodeArtifact Opération d'API permettant de 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 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code présenté sur cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWAA](#).
- UN [CodeArtifact référentiel](#) où vous stockez les dépendances de votre environnement.

Autorisations

Pour actualiser le CodeArtifact jeton et écriture du résultat dans Amazon S3 Amazon MWAA doit disposer des autorisations suivantes dans le rôle d'exécution.

- Le `codeartifact:GetAuthorizationToken` cette 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/*"
}
```

- `sts:GetServiceBearerToken` une action est requise pour appeler le CodeArtifact [GetAuthorizationToken](#) Fonctionnement de l'API. Cette opération renvoie un jeton qui doit être utilisé lors de l'utilisation d'un gestionnaire de packages tel que `pip` avec CodeArtifact. Pour utiliser un gestionnaire de packages avec CodeArtifact référentiel, le rôle d'exécution de votre environnement doit permettre `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 sous `code_artifact_startup_script.sh`.

```
#!/bin/sh

# Startup script for MWA, see https://docs.aws.amazon.com/mwaa/latest/userguide/using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION --domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION --domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
```

```
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"  
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-  
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"  
echo "CodeArtifact startup setup complete"
```

2. Accédez au dossier dans lequel vous avez enregistré le script. Utilisez `cp` dans une nouvelle fenêtre d'invite pour télécharger le script dans votre compartiment. Remplacez *your-s3-bucket* avec vos informations.

```
$ aws s3 cp code_artifact_startup_script.sh s3://your-s3-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://your-s3-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 sur le `dag` dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- En savoir plus sur la façon de télécharger le `plugins.zip` dans cet exemple, placez le fichier 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` pour `/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 dans le `plugins` dossier sur `startup`: vous permettant de définir et de modifier des variables d'environnement. L'exemple suivant explique comment créer un plug-in personnalisé à l'aide d'Apache Hive et Hadoop dans un environnement Amazon Managed Workflows pour Apache Airflow. Il peut être combiné à d'autres plug-ins et binaires personnalisés.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Dépendances de téléchargement](#)
- [Plugin personnalisé](#)
- [Plugins.zip](#)
- [Exemple de code](#)
- [Options de configuration du débit d'air](#)
- [Quelle est la prochaine étape ?](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, consultez [Installation des dépendances Python](#).

Apache Airflow v2

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Apache Airflow v1

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

Dépendances de téléchargement

Amazon MWAA va extraire le contenu du fichier `plugins.zip` dans `/usr/local/airflow/plugins` sur chaque planificateur Amazon MWAA et chaque conteneur de travail. Ceci est utilisé pour ajouter des fichiers binaires à votre environnement. Les étapes suivantes décrivent comment assembler les fichiers nécessaires au plug-in personnalisé.

1. Dans votre invite de commande, accédez au répertoire dans lequel vous souhaitez créer votre plugin. Par exemple :

```
cd plugins
```

2. Télécharger [Hadoop](#) à partir d'un [miroir](#), par exemple :

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

3. Télécharger [Ruche](#) à partir d'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. Par exemple :

```
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
```

Plugin personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier plugins au démarrage. Ceci est utilisé pour définir et modifier des variables d'environnement. Les étapes suivantes décrivent l'exemple de code pour le plug-in personnalisé.

1. Dans votre invite de commandes, accédez au `hive_plugin` répertoire. Par exemple :

```
cd hive_plugin
```

2. Copiez le contenu de l'exemple de code suivant et enregistrez-le localement sous `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 sous `airflowignore` dans le `hive_plugin` répertoire.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

Les étapes suivantes montrent comment créer `plugins.zip`. Le contenu de cet exemple peut être combiné avec d'autres plugins et binaires en un seul `plugins.zip` dossier.

1. Dans votre invite de commandes, accédez au `hive_plugin` répertoire de l'étape précédente. Par exemple :

```
cd hive_plugin
```

2. Comprimez le contenu dans votre `plugins` dossier.

```
zip -r ../hive_plugin.zip ./
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG qui testera le plug-in personnalisé.

1. Dans votre invite de commandes, 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 `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 du débit d'air

Si vous utilisez Apache Airflow v2, ajoutez `core.lazy_load_plugins : False` en tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, consultez [Utiliser les options de configuration pour charger des plugins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment charger le code DAG de cet exemple vers le `dag` dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- En savoir plus sur la façon de télécharger le `plugins.zip` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Création d'un plugin personnalisé pour Apache Airflow Python Virtualenv Operator

L'exemple suivant montre comment appliquer un correctif à Apache Airflow Python Virtualenv Operator avec un plugin personnalisé sur Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Prérequis](#)
- [Exemple de code de plugin personnalisé](#)
- [Plugins.zip](#)
- [Exemple de code](#)
- [Options de configuration du débit d'air](#)
- [Quelle est la prochaine étape ?](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).

Autorisations

- Aucune autorisation supplémentaire n'est requise pour utiliser l'exemple de code de cette page.

Prérequis

Pour utiliser l'exemple de code de cette page, ajoutez les dépendances suivantes à votre `requirements.txt`. Pour en savoir plus, consultez [Installation des dépendances Python](#).

```
virtualenv
```

Exemple de code de plugin personnalisé

Apache Airflow exécutera le contenu des fichiers Python dans le dossier plugins au démarrage. Ce plugin va patcher le plugin intégré `PythonVirtualenvOperator` au cours de ce processus de démarrage pour le rendre compatible avec Amazon MWAA. Les étapes suivantes présentent l'exemple de code pour le plug-in personnalisé.

Apache Airflow v2

1. Dans votre invite de commandes, accédez au répertoire ci-dessus. Par exemple :

```
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'
```

Apache Airflow v1

1. Dans votre invite de commandes, accédez au répertoire de plugins ci-dessus. Par exemple :

```
cd plugins
```

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

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

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

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

Plugins.zip

Les étapes suivantes montrent comment créer `plugins.zip`.

1. Dans votre invite de commandes, accédez au répertoire contenant `virtual_python_plugin.py` ci-dessus. Par exemple :

```
cd plugins
```

2. Comprimez le contenu dans votre `plugins` dossier.

```
zip plugins.zip virtual_python_plugin.py
```

Exemple de code

Les étapes suivantes décrivent comment créer le code DAG pour le plug-in personnalisé.

Apache Airflow v2

1. Dans votre invite de commandes, 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 `virtualenv_test.py`.

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

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

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

from airflow import DAG
from airflow.operators.python import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
```

```
        system_site_packages=False,  
        dag=dag,  
    )
```

Apache Airflow v1

1. Dans votre invite de commandes, 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 `virtualenv_test.py`.

```
"""  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
  
Permission is hereby granted, free of charge, to any person obtaining a copy of  
this software and associated documentation files (the "Software"), to deal in  
the Software without restriction, including without limitation the rights to  
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of  
the Software, and to permit persons to whom the Software is furnished to do so.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS  
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR  
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER  
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN  
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.  
"""  
  
from airflow import DAG  
from airflow.operators.python_operator import PythonVirtualenvOperator  
from airflow.utils.dates import days_ago  
import os  
  
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"  
  
def virtualenv_fn():  
    import boto3  
    print("boto3 version ",boto3.__version__)
```

```
with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Options de configuration du débit d'air

Si vous utilisez Apache Airflow v2, ajoutez `core.lazy_load_plugins : False` en tant qu'option de configuration d'Apache Airflow. Pour en savoir plus, consultez [Utiliser les options de configuration pour charger des plugins en 2](#).

Quelle est la prochaine étape ?

- Découvrez comment charger le `requirements.txt` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment charger le code DAG de cet exemple vers le dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- En savoir plus sur la façon de télécharger le `plugins.zip` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation de plugins personnalisés](#).

Invoquer des DAG 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](#)
- [Prérequis](#)
- [Autorisations](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code présenté sur cette page avec Apache Airflow v2 ou version ultérieure en [Python 3.10](#).

Prérequis

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 gérée.

Autorisations

Pour utiliser l'exemple de code présenté sur cette page, le rôle d'exécution de votre environnement Amazon MWAA doit avoir accès pour effectuer `airflow:CreateCliToken`. Vous pouvez fournir cette autorisation à l'aide de la politique `AmazonMWAAirflowCliAccess` AWS gérée :

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



```
}
```

Pour plus d'informations, consultez [Politique de la CLI Apache Airflow : AmazonMWAA AirflowCliAccess](#).

Dépendances

- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise l'[installation de base d'Apache Airflow v2](#) sur votre environnement.

Exemple de code

1. Ouvrez la AWS Lambda console à l'[adresse https://console.aws.amazon.com/lambda/](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 Deploy (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 page DAG, localisez votre nouveau DAG cible dans la liste des DAG.
 - 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 de DAG dans différents environnements Amazon MAAA

L'exemple de code suivant crée un jeton CLI 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](#)
- [Prérequis](#)
- [Autorisations](#)
- [Dépendances](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous devez disposer des éléments suivants :

- Deux [Environnements Amazon MWA](#) avec réseau public accès au serveur Web, y compris à votre environnement actuel.
- Un exemple de DAG chargé dans le compartiment Amazon Simple Storage Service (Amazon S3) de votre environnement cible.

Autorisations

Pour utiliser l'exemple de code de cette page, le rôle d'exécution de votre environnement doit être autorisé à créer un jeton CLI Apache Airflow. Vous pouvez joindre la [AWS politique](#) gérée [AmazonMWA AirflowCliAccess](#) pour accorder cette autorisation.

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

Pour plus d'informations, veuillez consulter [Politique de la CLI Apache Airflow : AmazonMWA AirflowCliAccess](#).

Dépendances

- Pour utiliser cet exemple de code avec Apache Airflow v2, aucune dépendance supplémentaire n'est requise. Le code utilise le [Installation de base d'Apache Airflow v2](#) sur votre environnement.

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 `sousinvoke_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 appeler.

```
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('mwa')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwa/cli"
    body = 'dags trigger your-target-dag-id'
    headers = {
        'Authorization': 'Bearer ' + token['CliToken'],
        'Content-Type': 'text/plain'
    }
    requests.post(url, data=body, headers=headers)

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
```

```
    dagrun_timeout=timedelta(minutes=60),
    catchup=False
)
def invoke_dag():
    t = invoke_dag_task()

invoke_dag_test = invoke_dag()
```

3. Exécutez la commande suivante AWS CLI command permettant de copier le DAG dans le bucket de votre environnement, puis de déclencher 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 verrez 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 appelé, accédez à l'interface utilisateur d'Apache Airflow correspondant à votre nouvel environnement, puis procédez comme suit :

- a. Sur le DAG page, localisez votre nouveau DAG cible dans la liste des DAG.
- b. Sous Dernière course, vérifiez l'horodatage de la dernière exécution du DAG. Cet horodatage doit correspondre étroitement au dernier horodatage pour `invoke_dag` dans votre autre environnement.
- c. Sous Tâches récentes, vérifiez que la dernière exécution s'est bien déroulée.

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 [RDS pour SQL Server](#). L'exemple de code suivant utilise des DAG dans un environnement Amazon Managed Workflows for Apache Airflow pour se connecter à un serveur Amazon RDS pour Microsoft SQL Server et y exécuter des requêtes.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Dépendances](#)
- [Connexion Apache Airflow v2](#)
- [Exemple de code](#)
- [Quelle est la prochaine étape ?](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).
- Amazon MWAA et RDS pour 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 d'Amazon MWAA

- Ou une règle sortante pour le port de 1433 ouvrir depuis Amazon MWA vers RDS
- La connexion Apache Airflow pour RDS pour 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 de cette section, ajoutez la dépendance suivante à votre `requirements.txt`. Pour en savoir plus, consultez [Installation des dépendances Python](#)

Apache Airflow v2

```
apache-airflow-providers-microsoft-mssql==1.0.1
apache-airflow-providers-odbc==1.0.1
pymssql==2.2.1
```

Apache Airflow v1

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

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. ID de connexion : `mssql_default`
2. Type de connecteur : `Amazon Web Services`
3. Hôte : `YOUR_DB_HOST`
4. Schéma :
5. Login : `administrateur`
6. Mot de passe :
7. Port : `1433`
8. Supplémentaire :

Exemple de code

1. Dans votre invite de commandes, 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` dans cet exemple, dans votre compartiment Amazon S3 dans [Installation des dépendances Python](#).
- Découvrez comment charger le code DAG de cet exemple vers le `dags` dossier dans votre compartiment Amazon S3 dans [Ajout ou mise à jour des DAG](#).
- Découvrez 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, utilisez [mssql_operator](#) dans le Guide de référence d'Apache Airflow.

Utilisation avec Amazon EMR avec Amazon EMR

L'exemple de code suivant montre comment activer une intégration à l'aide d'Amazon EMR et d'Amazon Managed Workflows pour Apache Airflow.

Rubriques

- [Version](#)
- [Exemple de code](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).

Exemple de code

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

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

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

from airflow.contrib.operators.emr_add_steps_operator import EmrAddStepsOperator
from airflow.contrib.operators.emr_create_job_flow_operator import
EmrCreateJobFlowOperator
from airflow.contrib.sensors.emr_step_sensor import EmrStepSensor

from airflow.utils.dates import days_ago
from datetime import timedelta
import os

DAG_ID = os.path.basename(__file__).replace(".py", "")

DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}
```

```
SPARK_STEPS = [  
  {  
    'Name': 'calculate_pi',  
    'ActionOnFailure': 'CONTINUE',  
    'HadoopJarStep': {  
      'Jar': 'command-runner.jar',  
      'Args': ['/usr/lib/spark/bin/run-example', 'SparkPi', '10'],  
    },  
  },  
]  
  
JOB_FLOW_OVERRIDES = {  
  'Name': 'my-demo-cluster',  
  'ReleaseLabel': 'emr-5.30.1',  
  'Applications': [  
    {  
      'Name': 'Spark'  
    },  
  ],  
  'Instances': {  
    'InstanceGroups': [  
      {  
        'Name': "Master nodes",  
        'Market': 'ON_DEMAND',  
        'InstanceRole': 'MASTER',  
        'InstanceType': 'm5.xlarge',  
        'InstanceCount': 1,  
      },  
      {  
        'Name': "Slave nodes",  
        'Market': 'ON_DEMAND',  
        'InstanceRole': 'CORE',  
        'InstanceType': 'm5.xlarge',  
        'InstanceCount': 2,  
      }  
    ],  
    'KeepJobFlowAliveWhenNoSteps': False,  
    'TerminationProtected': False,  
    'Ec2KeyName': 'mykeypair',  
  },  
  'VisibleToAllUsers': True,  
  'JobFlowRole': 'EMR_EC2_DefaultRole',  
  'ServiceRole': 'EMR_DefaultRole'
```

```
}

with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['emr'],
) as dag:

    cluster_creator = EmrCreateJobFlowOperator(
        task_id='create_job_flow',
        job_flow_overrides=JOB_FLOW_OVERRIDES
    )

    step_adder = EmrAddStepsOperator(
        task_id='add_steps',
        job_flow_id="{{ task_instance.xcom_pull(task_ids='create_job_flow',
key='return_value') }}",
        aws_conn_id='aws_default',
        steps=SPARK_STEPS,
    )

    step_checker = EmrStepSensor(
        task_id='watch_step',
        job_flow_id="{{ task_instance.xcom_pull('create_job_flow',
key='return_value') }}",
        step_id="{{ task_instance.xcom_pull(task_ids='add_steps',
key='return_value')[0] }}",
        aws_conn_id='aws_default',
    )

    cluster_creator >> step_adder >> step_checker
```

Utilisation d'Amazon MAAA avec Amazon EKS

L'exemple suivant montre comment utiliser Amazon Managed Workflows pour Apache Airflow avec Amazon EKS.

Rubriques

- [Version](#)

- [Prérequis](#)
- [Création d'une clé publique pour Amazon EC2](#)
- [Création du cluster](#)
- [Créez un namespace de noms](#)
- [Créez un rôle pour un namespace de 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 vers le compartiment Amazon S3](#)
- [Activer et déclencher l'exemple](#)

Version

- L'exemple de code de cette page peut être utilisé avec Apache Airflow v1 dans [Python 3.7](#).
- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple présenté dans cette rubrique, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).
- eksctl. Pour en savoir plus, voir [Installer eksctl](#).
- kubectl. Pour en savoir plus, voir [Installation et configuration de kubectl](#). Dans certains cas, il est installé avec eksctl.
- Une paire de clés EC2 dans la région dans laquelle vous créez votre environnement Amazon MWA. Pour en savoir plus, voir [Création ou importation d'une paire de clés](#).

Note

Lorsque vous utilisez une `eksctl` commande, vous pouvez inclure une `--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, voir [Récupération de la clé publique pour votre paire de clés](#).

Création du cluster

Utilisez la commande suivante pour créer le cluster. Si vous souhaitez attribuer un nom personnalisé au cluster ou le créer dans une autre région, remplacez les valeurs du nom et de la région. Vous devez créer le cluster dans la même région que celle dans laquelle vous avez créé 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 du `ssh-public-key` correspondant 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 que celle dans laquelle 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 que vous avez terminé, vous pouvez vérifier que le cluster a été créé avec succès 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 mwa-eks \  
--approve
```

Créez un `mwa` espace de noms

Après avoir confirmé que le cluster a été créé avec succès, utilisez la commande suivante pour créer un espace de noms pour les pods.

```
kubectl create namespace mwa
```

Créez un rôle pour `mwa` espace de noms

Après avoir créé l'espace de nommage, créez un rôle et une liaison de rôles pour un utilisateur Amazon MWAA sur EKS qui pourra exécuter des pods dans un espace de nommage MWAA. Si vous avez utilisé un nom différent pour l'espace de noms, remplacez `mwa` dans `-n mwa` avec le nom que vous avez utilisé.

```
cat << EOF | kubectl apply -f - -n mwa  
kind: Role  
apiVersion: rbac.authorization.k8s.io/v1  
metadata:  
  name: mwa-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: mwaas-role-binding
subjects:
- kind: User
  name: mwaas-service
roleRef:
  kind: Role
  name: mwaas-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 *mwaas*:

```
kubectl get pods -n mwaas --as mwaas-service
```

Vous devriez voir s'afficher un message indiquant :

```
No resources found in mwaas 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 utilisé uniquement 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 d'associer les politiques décrites dans cette rubrique, associez la politique suivante :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "airflow:PublishMetrics",
      "Resource": "arn:aws:airflow:${MWAA_REGION}:${ACCOUNT_NUMBER}:environment/
${MWAA_ENV_NAME}"
    },
    {
      "Effect": "Deny",
      "Action": "s3:ListAllMyBuckets",
      "Resource": [
        "arn:aws:s3:::{MWAA_S3_BUCKET}",
        "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "s3:GetBucket*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::{MWAA_S3_BUCKET}",
        "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents",
        "logs:GetLogEvents",
        "logs:GetLogRecord",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults",

```

```

        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:${MWSAA_REGION}:${ACCOUNT_NUMBER}:log-group:airflow-
        ${MWSAA_ENV_NAME}-*"
    ]
},
{
    "Effect": "Allow",
    "Action": "cloudwatch:PutMetricData",
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "sqs:ChangeMessageVisibility",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:${MWSAA_REGION}:*:airflow-celery-*"
},
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey*",
        "kms:Encrypt"
    ],
    "NotResource": "arn:aws:kms:*:${ACCOUNT_NUMBER}:key/*",
    "Condition": {
        "StringLike": {
            "kms:ViaService": [
                "sqs.${MWSAA_REGION}.amazonaws.com"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": [

```

```
        "eks:DescribeCluster"
    ],
    "Resource": "arn:aws:eks:${MWSAA_REGION}:${ACCOUNT_NUMBER}: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 aux rôles ARN dont les 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, voir [Les rôles avec des chemins ne fonctionnent pas lorsque le chemin est inclus dans leur ARN dans le configmap 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 :

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

Pour en savoir plus, voir [Comment utiliser les politiques de confiance avec les rôles IAM](#).

Créez le fichier requirements.txt

Pour utiliser l'exemple de code de cette section, assurez-vous d'avoir ajouté l'une des options de base de données suivantes à votre `requirements.txt`. Pour en savoir plus, consultez [Installation des dépendances Python](#).

Apache Airflow v2

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Apache Airflow v1

```
awscli
kubernetes==12.0.1
```

Création d'un mappage d'identité pour Amazon EKS

Utilisez l'ARN correspondant au rôle que vous avez créé dans la commande suivante afin de créer un mappage d'identité pour Amazon EKS. Changez de région *your région* vers la région dans laquelle vous avez créé l'environnement. Remplacez l'ARN pour le rôle, et enfin, remplacez *mwaa-execution-role* avec le rôle d'exécution de votre environnement.

```
eksctl create iamidentitymapping \
--region your-region \
--cluster mwaa-eks \
--arn arn:aws:iam::111222333444:role/mwaa-execution-role \
--username mwaa-service
```

Créer le kubeconfig

Utilisez la commande suivante pour créer le `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 lorsque vous avez couru `update-kubeconfig` devez supprimer la section `env` 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, tel que `mwa_pod_example.py` pour le DAG.

Apache Airflow v2

```
"""  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
Permission is hereby granted, free of charge, to any person obtaining a copy of  
this software and associated documentation files (the "Software"), to deal in  
the Software without restriction, including without limitation the rights to  
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of  
the Software, and to permit persons to whom the Software is furnished to do so.  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS  
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR  
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER  
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN  
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.  
"""  
  
from airflow import DAG  
from datetime import datetime  
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import  
    KubernetesPodOperator  
  
default_args = {  
    'owner': 'aws',  
    'depends_on_past': False,  
    'start_date': datetime(2019, 2, 20),  
    'provide_context': True  
}  
  
dag = DAG(  

```

```
'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
    namespace="mwa",
    image="ubuntu:18.04",
    cmds=["bash"],
    arguments=["-c", "ls"],
    labels={"foo": "bar"},
    name="mwa-pod-test",
    task_id="pod-task",
    get_logs=True,
    dag=dag,
    is_delete_operator_pod=False,
    config_file=kube_config_path,
    in_cluster=False,
    cluster_context='aws'
)
```

Apache Airflow v1

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""

from airflow import DAG
from datetime import datetime
from airflow.contrib.operators.kubernetes_pod_operator import KubernetesPodOperator

default_args = {
    'owner': 'aws',
```

```
'depends_on_past': False,
'start_date': datetime(2019, 2, 20),
'provide_context': True
}

dag = DAG(
    'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
    namespace="mwa",
    image="ubuntu:18.04",
    cmds=["bash"],
    arguments=["-c", "ls"],
    labels={"foo": "bar"},
    name="mwa-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 `kube_config.yaml` vers le compartiment Amazon S3

Mettez 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 s'est exécuté et s'est terminé correctement, utilisez la commande suivante pour vérifier le pod :

```
kubectl get pods -n mwa
```


Vous devez voir des résultats similaires à ce qui suit :

```
NAME READY STATUS RESTARTS AGE
mwa-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 mwa-pod-test-aa11bb22cc3344445555666677778888
```

Connexion à Amazon ECS à l'aide de `ECSOperator`

La rubrique décrit comment vous pouvez utiliser `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, en utilisant `AWS CloudFormation` modèle pour créer un cluster Amazon ECS Fargate, puis pour créer et charger un DAG qui se connecte à votre nouveau cluster.

Rubriques

- [Version](#)
- [Prérequis](#)
- [Autorisations](#)
- [Création d'un cluster Amazon ECS](#)
- [Exemple de code](#)

Version

- Vous pouvez utiliser l'exemple de code de cette page avec Apache Airflow v2 et versions ultérieures dans [Python 3.10](#).

Prérequis

Pour utiliser l'exemple de code de cette page, vous aurez besoin des éléments suivants :

- Un [Environnement Amazon MWA](#).

Autorisations

- Le rôle d'exécution de votre environnement doit être autorisé à exécuter des tâches dans Amazon ECS. Vous pouvez soit joindre le [Amazon ECS_FullAccess](#) AWS-une politique gérée pour votre rôle d'exécution, ou créez et associez la politique suivante à votre rôle d'exécution.

```
{
  "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 le `CloudWatchEnregistre` la déclaration de politique dans votre rôle d'exécution Amazon MWAA pour autoriser l'accès au groupe de journaux de tâches Amazon ECS, comme indiqué ci-dessous. Le groupe de journaux Amazon ECS est créé par `AWS 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:region:account-id:log-group:airflow-environment-name-*",
    "arn:aws:logs:*:*:log-group:ecs-mwaa-group:"
  ]
}

```

Pour plus d'informations sur le rôle d'exécution d'Amazon MWAA et sur la manière d'associer une politique, consultez [Rôle d'exécution](#).

Création d'un cluster Amazon ECS

En utilisant ce qui suit AWS CloudFormation modèle, vous allez créer un cluster Amazon ECS Fargate à utiliser avec votre flux de travail Amazon MWAA. Pour plus d'informations, voir [Création d'une définition de tâche](#) dans le Guide du développeur Amazon Elastic Container Service.

1. Créez un fichier JSON avec le code suivant et enregistrez-le sous `secs-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 MAAA."
    }
},
"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"
      ]
    }
  },
```


Vous pouvez également utiliser le script shell suivant. Le script extrait les valeurs requises pour les groupes de sécurité et les sous-réseaux de votre environnement à l'aide du [get-environment](#) AWS CLI commande, puis crée la pile en conséquence. Pour exécuter le script, procédez comme suit.

- a. Copiez et enregistrez le script `sousecs-stack-helper.sh` dans le même répertoire que votre AWS 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` avec vos informations.

```
$ chmod +x ecs-stack-helper.sh
$ ./ecs-stack-helper.sh environment-name stack-name
```

En cas de succès, vous verrez la sortie suivante affichant votre nouveau AWS CloudFormation ID de pile.

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

Après votre AWS CloudFormation la pile est terminée et AWS a provisionné vos ressources Amazon ECS, vous êtes prêt à créer et à charger votre DAG.

Exemple de code

1. Ouvrez une invite de commande et 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 `mwa-ecs-operator.py`, puis chargez 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="mwa-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
information.
CONTAINER_NAME="mwa-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="mwa-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 qu'un groupe de journaux nommé `mwa-ecs-zero`. Pour `awslogs_stream_prefix`, utilisez le préfixe de flux du journal des tâches Amazon ECS. L'exemple suppose un préfixe de flux de journal, `ecs`.

3. Exécutez la commande suivante AWS CLI command permettant de copier le DAG dans le bucket de votre environnement, puis de déclencher 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 verrez un résultat similaire à ce qui suit dans les journaux des tâches pour `ecs_operator_task` dans le `ecs_fargate_dag` JOUR :

```
[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-
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](#)
- [Prérequis](#)
- [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 ou version ultérieure en [Python 3.10](#).

Prérequis

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](#) depuis le 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://mwa-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/.local/bin/dbt --version"
    )
```

Procédez comme suit pour afficher les 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 sous 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 page décrit les meilleures pratiques que nous recommandons pour optimiser les performances d'un environnement Amazon Managed Workflows pour Apache Airflow en utilisant [Utilisation des options de configuration d'Apache Airflow sur Amazon MWAA](#).

Table des matières

- [Ajout d'une option de configuration Apache Airflow](#)
- [Planificateur Apache Airflow](#)
 - [Paramètres](#)
 - [Limites](#)
- [Dossiers DAG](#)
 - [Paramètres](#)
- [fichiers DAG](#)
 - [Paramètres](#)
- [Tâches](#)
 - [Paramètres](#)

Ajout d'une option de configuration Apache Airflow

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 saisissez 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, veuillez consulter la section [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 des DAG et la planification des tâches. Pour plus d'informations sur le réglage du planificateur Apache Airflow, voir [Affiner les performances de votre planificateur sur le site Web de documentation](#) d'Apache Airflow.

Paramètres

Cette section décrit les options de configuration disponibles pour le planificateur Apache Airflow et leurs cas d'utilisation.

Apache Airflow v2

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	celery.sync_parallelism	1	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 journaux des tâches à CloudWatch Logs. La définition de la valeur sur 0 signifie utiliser le nombre maximum de processus, mais cela peut entraîner des erreurs lors de la remise des journaux des tâches.

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	scheduler .processo r_poll_interval	1	Le nombre de secondes à attendre entre deux traitements consécutifs de fichiers DAG dans la « boucle » du planificateur.	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 <code>scheduler</code>

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
				.parsing_processes Apache Airflow v2 et Apache Airflow scheduler .max_threads v1. Cela peut réduire la capacité des planificateurs à analyser les DAG et augmenter le temps nécessaire pour que les DAG apparaissent sur le serveur Web.
v2	scheduler_max_dagruns_to_create_per_loop	10	Le nombre maximum de DAG à créer pour chaque « boucle » du planificateur.	Vous pouvez utiliser cette option pour libérer des ressources pour la planification des tâches en diminuant le nombre maximum DagRuns de « boucles » du planificateur.

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	scheduler .parsing_ processes	Définissez à l'aide de la formule suivante : (2 * number of vCPUs) - 1 par défaut.	Le nombre de threads que le planificateur peut exécuter en parallèle pour planifier les DAG.	Vous pouvez utiliser cette option pour libérer des ressources en diminuant le nombre de processus que le planificateur exécute en parallèle pour analyser les DAG. 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, consultez la section Limites .

Limites

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`

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 que deux threads ont multiplié quatre vCPU, moins un pour le planificateur lui-même. La valeur que vous spécifiez dans `scheduler.max_threads` et `scheduler.parsing_processes` doit pas dépasser le nombre de threads disponibles pour une classe d'environnement (comme indiqué ci-dessous :

- `mw1.small` — Ne doit pas dépasser le nombre de 1 threads pour les autres processus. Le thread restant est réservé au planificateur.
- `mw1.medium` — Le nombre de threads 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 dossier DAGs 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.

Paramètres

Cette section décrit les options de configuration disponibles pour le dossier DAGs et leurs cas d'utilisation.

Apache Airflow v2

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	scheduler.dag_dir_list_interval	300 secondes	Le nombre de secondes pendant lesquelles le dossier DAGs doit être scanné à la recherche de nouveaux fichiers.	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes nécessaires à l'analyse du dossier DAGs. 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 dossier DAGs.
v2	scheduler.min_file_process_interval	30 secondes	Le nombre de secondes après lequel le planificateur analyse un DAG	Vous pouvez utiliser cette option pour libérer des ressources en

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
			et les mises à jour du DAG sont reflétées.	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` doit s'écouler avant que le même fichier ne soit à nouveau analysé.

Paramètres

Cette section décrit les options de configuration disponibles pour les fichiers DAG Apache Airflow et leurs cas d'utilisation.

Apache Airflow v2

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.dag_file_processor_timeout	50 secondes	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 constatez des délais dans les journaux de traitement de vos DAG qui empêchent le chargement de DAG viables.

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.dagbag_import_timeout	30 secondes	Le délai de secondes avant l'importation d'un fichier Python est expiré.	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>core.dagbag_processor_timeout</code>

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.min_serialize_dag_update_interval	30	Nombre minimal de secondes après lequel les DAG sérialisés de la base de données sont mis à jour.	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre de secondes après lesquelles les DAG sérialisés de la base de données sont mis à jour. Nous vous recommandons d'augmenter cette valeur si vous avez un grand nombre de DAG ou des DAG complexes . L'augmentation de cette valeur réduit la charge sur le planificateur et la base de données lorsque les DAG sont sérialisés.

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.min_serialize_dag_fetch_interval	10	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	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_serialize_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

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
				lorsque les DAG sont sérialisés.

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 de la file d'attente. Le planificateur fait passer les tâches analysées prêtes à être planifiées du statut Aucune au statut Planifié. L'exécuteur, également exécuté 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.

Paramètres

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 marquées en rouge.

Apache Airflow v2

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.parallélisme	Réglé dynamiquement en fonction de <code>(maxWorkers * maxCeleryWorkers) / schedulers * 1.5</code> .	Nombre maximal d'instances de tâches pouvant avoir le statut « En cours d'exécution ».	Vous pouvez utiliser cette option pour libérer des ressources en augmentant le nombre d'instances de tâches pouvant

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
				être exécutées simultanément. La valeur spécifiée doit être le nombre de travailleurs disponibles « multiplié » par la densité de tâches des travailleurs. Nous vous recommandons de modifier cette valeur uniquement lorsque vous constatez qu'un grand nombre de tâches sont bloquées dans l'état « En cours » ou « En file d'attente ».

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	core.dag_concurrency	10 000	Le nombre d'instances de tâches autorisées à s'exécuter simultanément pour chaque DAG.	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 DAG avec dix tâches parallèles et que vous souhaitez que tous les DAG s'exécutent simultanément, vous pouvez calculer le parallélisme maximal comme le nombre de travailleurs disponibles « fois » la densité des tâches des travailleurs <code>celery.worker_conc</code>

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
				urrency , divisé par le nombre de DAG (par exemple 100).
v2	core.execute_tasks_new_python_interpreter	True	Détermine si Apache Airflow exécute des tâches en bifurquant le processus parent ou en créant un nouveau processus Python.	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.
v2	celery.worker_concurrency	N/A	Amazon MWAA remplace l'installation de base d'Airflow pour cette option afin de dimensionner Workers dans le cadre de son composant de dimensionnement automatique.	<i>Toute valeur spécifiée pour cette option est ignorée.</i>

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
v2	celery.worker_autoscale	<p>mw1.small - 5,0</p> <p>mw1.medium - 10,0</p> <p>mW1.large - 20,0</p> <p>mw1.xlarge - 40,0</p> <p>mW1,2xlarge - 80,0</p>	La simultanéité des tâches pour les travailleurs.	<p>Vous pouvez utiliser cette option pour libérer des ressources en réduisant la minimum simultanéité des maximum 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</p>

Version	Option de configuration	Par défaut	Description	Cas d'utilisation
				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 page décrit les meilleures pratiques que nous recommandons pour installer et gérer les dépendances Python dans un `requirements.txt` fichier pour un environnement Amazon Managed Workflows for Apache Airflow.

Table des matières

- [Test des DAG à l'aide de l'utilitaire Amazon MAWA 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](#)
- [Afficher les journaux sur la console CloudWatch Logs](#)
- [Affichage des erreurs dans l'interface utilisateur d'Apache Airflow](#)
 - [Connexion à Apache Airflow](#)

- [Exemples de requirements.txt scénarios](#)

Test des DAG à l'aide de l'utilitaire Amazon MAWA 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. Cela vous permet d'exécuter un environnement Apache Airflow local pour développer et tester des DAG, des plugins personnalisés et des dépendances avant le déploiement sur Amazon MWAA.
- Pour exécuter la CLI, reportez-vous à la section [aws-mwaa-local-runner](#) on 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 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, consultez [aws-mwaa-local-runner](#) le GitHub
2. Consultez les suppléments du package Apache Airflow. Pour consulter la liste des packages installés pour Apache Airflow v2 sur Amazon MWAA, consultez Amazon MWAA [local runner 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 instruction. `--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, consultez 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 consulter les 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, consultez [the section called "Affichage des journaux de flux d'air"](#).

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 votre `requirements.txt`.

Apache Airflow v1

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, consultez [aws-mwaa-local-runner](#) le GitHub
2. Consultez les extras du package Airflow. [Consultez la liste des packages disponibles pour Apache Airflow v1.10.12 à l'adresse `https://raw.githubusercontent.com/apache/airflow/constraints-1.10.12/constraints-3.7.txt`](#).
3. Ajoutez le fichier de contraintes. Ajoutez le fichier de contraintes pour Apache Airflow v1.10.12 en haut de votre fichier `requirements.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.

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

4. Paquets Apache Airflow v1.10.12. Ajoutez les [extras du package Airflow](#) et la version Apache Airflow v1.10.12 (). `==` 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]==1.10.12
```

Exemple Shell sécurisé (SSH)

Le `requirements.txt` fichier d'exemple suivant installe SSH pour Apache Airflow v1.10.12.

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

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

Le `requirements.txt` fichier d'exemple suivant installe la bibliothèque Boto3 pour Apache Airflow v1.10.12.

```
boto3 == 1.17.4
```

[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`.

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 votre `requirements.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 Apache Airflow (pour Apache Airflow v2.2.2 et versions ultérieures) recherchent des plugins personnalisés lors du démarrage sur le conteneur AWS Fargate géré pour votre environnement à l'adresse. `/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 doit 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 des DAG. 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 [local Amazon MWAA existant requirements.txt](#) ou 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 ci-dessous

```
--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 votre `requirements.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

Les URL ne sont pas accessibles depuis les serveurs Web privés qui installent les exigences 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 regrouper sur Amazon S3 :

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

S3_BUCKET = 'my-s3-bucket'
S3_KEY = 'backup/plugins_whl.zip'

with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,
        start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/
requirements/requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3
cp /tmp/plugins.zip s3://{S3_BUCKET}/{S3_KEY}"
    )
```

Après avoir exécuté le DAG, utilisez ce nouveau fichier comme Amazon MWAAplugins.zip, éventuellement, empaqueté avec d'autres plugins. Ensuite, mettez à jour requirements.txt votre précédent --find-links /usr/local/airflow/plugins et --no-index sans ajout--constraint.

Cette méthode vous permet d'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, consultez [Rôle d'exécution Amazon MWAA](#).

Vous pouvez activer les journaux Apache Airflow au CRITICAL niveau INFOWARNING,ERROR, ou. 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 de visualiser les journaux reçus pour le `requirements.txt`

Airflow scheduler logs

Log level

Specify which types of task events to log

INFO Log info and higher-severity events	▲
CRITICAL Log critical events only	
ERROR Log error and higher-severity events	
WARNING Log warning and higher-severity events	
INFO Log info and higher-severity events	

Afficher les journaux sur la console CloudWatch Logs

Vous pouvez consulter les journaux 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 afficher les journaux Apache Airflow sur la console Logs. CloudWatch

Pour consulter les 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. Vous devriez voir 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/00/2344469e2084fb28kjdsfiuyweb47389789vxbmbnjhsdgf5463acd6cf5e3db69324/
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, un message d'erreur similaire au suivant peut s'afficher :

```
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))
```

Affichage des erreurs 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 cette erreur s'affiche dans l'interface utilisateur d'Apache Airflow, il est probable qu'il vous manque une dépendance obligatoire dans votre `requirements.txt` fichier.

Connexion à Apache Airflow

Vous devez [Politique d'accès à l'interface utilisateur d'Apache Airflow : AmazonMWAA WebServerAccess](#) disposer d'autorisations pour accéder à votre AWS compte dans AWS Identity and Access Management (IAM) pour 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 les URL de dépôt personnalisées conformes 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 [CloudWatch tarification standard d'Amazon](#).

Pour plus d'informations sur la surveillance d'Apache Airflow, consultez la section [Logging & Monitoring](#) sur le site Web de documentation d'Apache Airflow.

Sections

- [Vue d'ensemble de la surveillance sur Amazon MWAA](#)
- [Afficher les journaux d'audit AWS CloudTrail](#)
- [Afficher les journaux Airflow sur Amazon CloudWatch](#)
- [Tableaux de bord de surveillance et alarmes sur Amazon MWAA](#)
- [Métriques de l'environnement Apache Airflow v2 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](#)

- [AWS CloudTrail vue d'ensemble](#)

CloudWatch Présentation d'Amazon

CloudWatch est un référentiel de métriques pour les AWS services qui vous permet de 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, consultez [Qu'est-ce qu'Amazon CloudWatch ?](#) .

AWS CloudTrail vue d'ensemble

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, consultez [Qu'est-ce que c'est AWS CloudTrail ?](#) .

Afficher les journaux d'audit AWS CloudTrail

AWS CloudTrail est activé sur votre AWS compte 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 CloudTrail qu'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 aux API Amazon 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](#)

- [Afficher les 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 journal pour consulter un enregistrement permanent des événements de votre AWS compte, 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 consulter 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, consultez la section [Création d'un historique pour votre AWS compte](#).

Afficher les 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 consulter les événements liés à la création, à la modification ou à la suppression de ressources (telles que les utilisateurs IAM ou d'autres AWS ressources) dans votre AWS compte par région. Pour en savoir plus, consultez la section [Affichage des événements avec 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 d'envoyer des événements sous forme de fichiers journaux à un compartiment Amazon S3 de votre choix.

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 n'apparaissent pas dans un ordre spécifique. L'exemple suivant est une entrée de journal pour l'CreateEnvironmentaction 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"
      }
    }
  },
  "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 les [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

Afficher les journaux Airflow sur Amazon CloudWatch

Amazon MWAA peut envoyer des journaux Apache Airflow à Amazon CloudWatch. Vous pouvez consulter les journaux de plusieurs environnements à partir d'un seul emplacement afin d'identifier facilement les retards dans les tâches Apache Airflow ou les erreurs de flux de travail sans avoir besoin d'outils tiers supplémentaires. Les journaux Apache Airflow doivent être activés sur la console Amazon Managed Workflows for Apache Airflow pour afficher le traitement du DAG Apache Airflow, les tâches, le serveur Web et les connexions des travailleurs. CloudWatch

Table des matières

- [Tarification](#)
- [Avant de commencer](#)

- [Types de journaux](#)
- [Activation des journaux Apache Airflow](#)
- [Afficher les journaux d'Apache Airflow](#)
- [Exemples de journaux du planificateur](#)
- [Quelle est la prochaine étape ?](#)

Tarifification

- CloudWatch Les frais de journalisation standard s'appliquent. Pour plus d'informations, consultez [CloudWatch les tarifs](#).

Avant de commencer

- Vous devez avoir un rôle qui permet de consulter les 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 Apache Airflow sont envoyés à **Airflow-v202-Public-Task**.

Type de journal	Description
YourEnvironmentName- DAGProces sing	Les journaux du gestionnaire du processeur DAG (la 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 CRITICAL niveau INFOWARNING,ERROR, ou. 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 de surveillance.
 - e. Choisissez le groupe de journaux de tâches Airflow dans le volet de surveillance.
 - f. Choisissez le niveau de journalisation dans Log level.
6. Choisissez Suivant.
7. Choisissez Enregistrer.

Afficher les journaux d'Apache Airflow

La section suivante décrit comment afficher les 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 consulter les journaux 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 afficher les journaux Apache Airflow sur la console Logs. CloudWatch

Pour consulter les 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. Vous devriez voir 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/00/2344469e2084fb28kjdsfiuyweb47389789vxbmbnjhsdgm5463acd6cf5e3db69324/
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, un message d'erreur similaire au suivant peut s'afficher :

```
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 vous permet de surveiller plus facilement l'état d'un environnement Amazon MWAA de manière proactive.

Apache Airflow expose les métriques d'un certain nombre de 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 est configuré pour envoyer 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 AWS 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 consulter la liste des indicateurs, voir :

- [Métriques de l'environnement Apache Airflow v2 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 métrique `executor.queued_tasks` Apache Airflow.
- TasksPending- Le nombre de tâches en attente dans l'exécuteur. Correspond à la métrique `scheduler.tasks.pending` Apache Airflow.

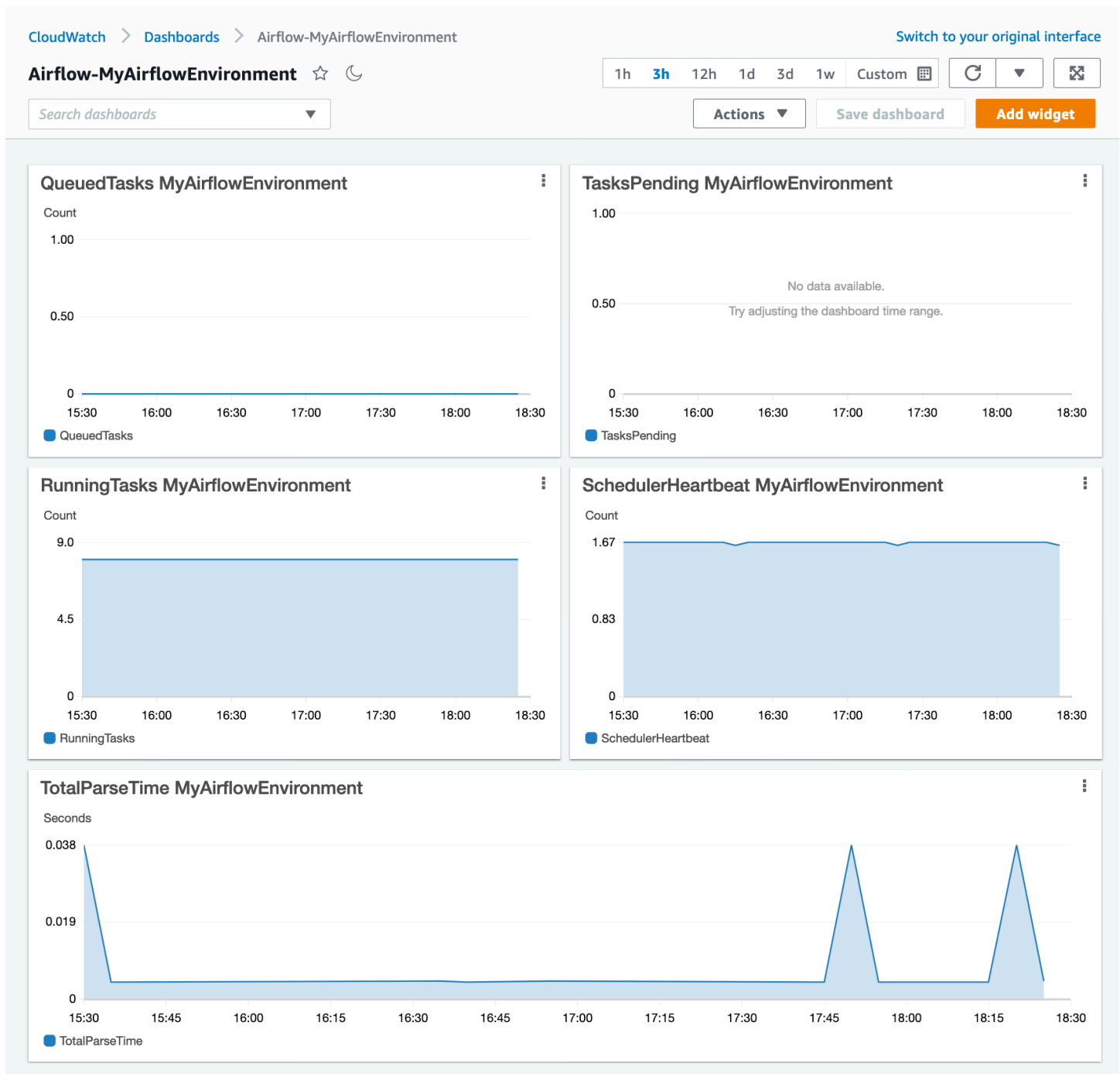
Note

Ne s'applique pas à Apache Airflow v2.2 et versions ultérieures.

- **RunningTasks**- Le nombre de tâches exécutées dans l'exécuteur. Correspond à la métrique `executor.running_tasks` 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 métrique `dag_processing.total_parse_time` 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 MWA](#)

En utilisant AWS CloudFormation

Vous pouvez utiliser la définition du AWS 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, consultez [la section Création d'une pile sur la AWS CloudFormation console](#). Pour ajouter une alarme au tableau de bord, consultez la section [Utilisation des alarmes](#).

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Creates MWA 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 les [CloudWatch FAQ Amazon](#).

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 v2 dans CloudWatch

Apache Airflow v2 est déjà configuré 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 Apache Airflow. Cette page décrit les métriques Apache Airflow disponibles dans CloudWatch la CloudWatch console et explique comment y accéder.

Table des matières

- [Conditions](#)
- [Dimensions](#)
- [Accès aux métriques dans la CloudWatch console](#)
- [Les métriques Apache Airflow sont disponibles dans CloudWatch](#)
 - [Compteurs Apache Airflow](#)
 - [Jauges de débit d'air Apache](#)
 - [Minuteries Apache Airflow](#)
- [Choix des indicateurs à signaler](#)
- [Quelle est la prochaine étape ?](#)

Conditions

Espace de noms

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 paire nom-valeur qui fait partie de l'identité d'une métrique.

Unité

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.			
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 un Job Apache Airflow exécuté par le planificateur. A toujours la valeur Job.			
Opérateur	Indique un opérateur Apache Airflow spécifique.			
Pool	Indique un pool de travailleurs Apache Airflow spécifique.			
Tâche	Indique une tâche Apache Airflow spécifique.			

Dimension	Description			
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 consulter les indicateurs de performance d'une dimension





1. Ouvrez la [page Metrics](#) sur la CloudWatch console.
2. Utilisez le sélecteur de AWS région pour sélectionner votre région.
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, TaskInstanceSuccessesou TaskInstanceDuration. Choisissez Représenter graphiquement tous les résultats de recherche.
6. Choisissez l'onglet Graphed metrics pour afficher les statistiques de performance des métriques Apache Airflow, telles que DAG, Environment, Task.



Les métriques 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	Unité	Dimension
SLA manqué <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note Disponible pour Apache Airflow v2.4.3 et versions ultérieures.</p> </div>	sla_missed	Nombre	Fonction, planificateur
Rappel SLA échoué <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note Disponible pour Apache Airflow v2.4.3 et versions ultérieures.</p> </div>	sla_callback_notification_failure	Nombre	Fonction, planificateur
Mises à jour <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.</p> </div>	jeu de données.mises à jour	Nombre	Fonction, planificateur
Orphelin <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.</p> </div>	jeu de données orphelin	Nombre	Fonction, planificateur


CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
FailedCeleryTaskExecution	celery.execute_command.failure	Nombre	Fonction, céleri	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E1F5FE;"> <p> Note Disponible pour Apache Airflow v2.4.3 et versions ultérieures.</p> </div>				
FilePathQueueUpdateCount	dag_processing.file_path_queue_update_count	Nombre	Fonction, planificateur	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E1F5FE;"> <p> Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.</p> </div>				
CriticalSectionBusy	scheduler.critical_section_busy	Nombre	Fonction, planificateur	
DagBagSize	taille_du_sac	Nombre	Fonction, traitement DAG	
DagCallbackExceptions	dag.callback_exceptions	Nombre	DAG, Tous	
SLA défaillant EmailAttempts	échec de notification par e-mail	Nombre	Fonction, planificateur	




CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
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
OperatorFailures	opérateur _failures_ {nom_opér ateur}	Nombre	Opérateur , {operator _name}
OperatorSuccesses	operator_ successes_ {nom_opér ateur}	Nombre	Opérateur , {operator _name}


CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
OtherCallbackCount	dag_processing.other_callback_count	Nombre	Fonction, planificateur	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p> Note Disponible dans Apache Airflow v2.6.3 et versions ultérieures.</p> </div>				
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	

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
SlaCallbackCount	dag_processing.sla_callback_count <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p> Note Disponible pour Apache Airflow v2.6.3 et versions ultérieures.</p> </div>	Nombre	Fonction, planificateur	
TasksKilledExternally	scheduler.tasks.killed_external	Nombre	Fonction, planificateur	
TaskTimeoutError	celery.task_timeout_error	Nombre	Fonction, céleri	
TaskInstanceCreatedUsingOperator	task_instance_created- {nom_opérateur}	Nombre	Opérateur, {operator_name}	

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
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_remo ved_from_ dag. {day_id}	Nombre	DAG, {day_id}
TaskRestoredToDAG	task_rest ored_to_day. {day_id}	Nombre	DAG, {day_id}
TriggersSucceeded	déclencheurs réussis	Nombre	Fonction, déclencheur

 **Note**
Disponible pour Apache Airflow v2.7.2 et versions ultérieures.

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
TriggersFailed	déclencheurs. Echec	Nombre	Fonction, déclencheur	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p> Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.</p> </div>				
TriggersBlockedMainThread	triggers.blocked_main_thread	Nombre	Fonction, déclencheur	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p> Note Disponible pour Apache Airflow v2.7.2 et versions ultérieures.</p> </div>				
TriggerHeartbeat	déclencheur du battement de cœur	Nombre	Fonction, déclencheur	
<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p> Note Disponible pour Apache Airflow v2.8.1 et versions ultérieures.</p> </div>				


CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
TaskInstanceCreatedUsingOperator	airflow.task_instance_created_{operator_name}	Nombre	Opérateur, {operator_name}
	<div data-bbox="589 590 794 1192" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>Disponible pour Apache Airflow v2.7.2 et versions ultérieures.</p> </div>		
ZombiesKilled	zombies tués	Nombre	DAG, Tous Tâche, tout

Jauges de débit d'air Apache


Les métriques Apache Airflow présentées dans cette section contiennent des données sur les jauges [Apache Airflow](#).

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
DAG FileRefreshError	dag_file_refresh_error	Nombre	Fonction, traitement DAG	
ImportErrors	dag_processing.import_errors	Nombre	Fonction, traitement DAG	
Exception Failures	smart_sensor_operator.exception_failures	Nombre	Fonction, opérateur de capteur intelligent	
ExecutedTasks	smart_sensor_operator.tâches exécutées	Nombre	Fonction, opérateur de capteur intelligent	
InfraFailures	smart_sensor_operator.infra_failures	Nombre	Fonction, opérateur de capteur intelligent	
LoadedTasks	smart_sensor_operator.loaded_tasks	Nombre	Fonction, opérateur de capteur intelligent	
TotalParseTime	dag_processing.total_parse_time	Secondes	Fonction, traitement DAG	
Triggered DagRuns	jeu de données .triggered_dagruns	Nombre	Fonction, planificateur	

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
<p>Note</p> <p>Disponible dans Apache Airflow v2.6.3 et versions ultérieures.</p>				
<p>Note</p> <p>Disponible dans Apache Airflow v2.7.2 et versions ultérieures.</p>	<p>TriggersRunning</p> <p>déclencheurs. Exécution. <i>{nom d'hôte}</i></p>	<p>Nombre</p>	<p>Fonction, déclencheur</p> <p>HostName, <i>{nom d'hôte}</i></p>	


CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
PoolDeferredSlots	pool.deferred_slots. {pool_name}	Nombre	Piscine, {pool_name}
<div style="border: 1px solid #00a0e3; border-radius: 10px; padding: 10px; background-color: #e1f5fe;"> <p> Note</p> <p>Disponible dans Apache Airflow v2.7.2 et versions ultérieures.</p> </div>			
DAG FileProcessingLastRunSecondsAgo	dag_processing.last_run. Il y a quelques secondes. {dag_filename}	Secondes	Nom du fichier DAG, {dag_filename}
OpenSlots	exécuteur. .open_slots	Nombre	Fonction, exécuteur
OrphanedTasksAdopted	scheduler. .orphaned_tasks.adopted	Nombre	Fonction, planificateur
OrphanedTasksCleared	scheduler. .orphaned_tasks.cleared	Nombre	Fonction, planificateur


CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
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	
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}	
PoolQueueSlots	pool.queued_slots.{nom_piscine}	Nombre	Piscine, {pool_name}	
PoolRunningSlots	pool.running_slots.{nom_piscine}	Nombre	Piscine, {pool_name}	
ProcessorTimeouts	dag_processing.processor_timeouts	Nombre	Fonction, traitement DAG	



CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
QueuedTasks	executor. queued_tasks	Nombre	Fonction, exécuteur
RunningTasks	exécuteur .running_tasks	Nombre	Fonction, exécuteur
TasksExecutable	scheduler .tasks.ex écutable	Nombre	Fonction, planificateur
TasksPending	scheduler .tasks.pending	Nombre	Fonction, planificateur
<div data-bbox="115 785 363 1339" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>Ne s'applique pas à Apache Airflow v2.2 et versions ultérieures.</p> </div>			
TasksRunning	scheduler .tasks.running	Nombre	Fonction, planificateur
TasksStarving	scheduler .tasks.starving	Nombre	Fonction, planificateur
TasksWithoutDagRun	scheduler .tasks.wi thout_dagrun	Nombre	Fonction, planificateur

Minuteries Apache Airflow

Les métriques Apache Airflow présentées dans cette section contiennent des données sur les minuteries [Apache Airflow](#).

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
Collectez les BDAG	collecte_db_dags	Millisecondes	Fonction, traitement DAG
CriticalSectionDuration	scheduler .critical_section_duration	Millisecondes	Fonction, planificateur
CriticalSectionQueryDuration	scheduler .critical_section_query_duration	Millisecondes	Fonction, planificateur
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>Disponible pour Apache Airflow v2.5.1 et versions ultérieures.</p> </div>			
DAG DependencyCheck	dagrun.dependency-check. {day_id}	Millisecondes	DAG, {day_id}
DAG DurationFailed	dagrun.duration. a échoué. {day_id}	Millisecondes	DAG, {day_id}

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension
DAG DurationSuccess	dagrun.duration.success. {day_id}	Millisecondes	DAG, {day_id}
DAG FileProcessingLastDuration	dag_processing.last_duration. {dag_filename}	Secondes	Nom du fichier DAG, {dag_filename}
DAG ScheduledDelay	dagrun.schedule_delay. {day_id}	Millisecondes	DAG, {day_id}
FirstTaskSchedulingDelay	dagrun. {dag_id}.first_task_scheduling_delay	Millisecondes	DAG, {day_id}
SchedulerLoopDuration	scheduler.schedule_loop_duration	Millisecondes	Fonction, planificateur
<div style="border: 1px solid #007bff; border-radius: 10px; padding: 10px; background-color: #e0f2f7;"> <p> Note</p> <p>Disponible pour Apache Airflow v2.5.1 et versions ultérieures.</p> </div>			
TaskInstanceDuration	jour. {day_id}. {task_id}.durée	Millisecondes	DAG, {day_id} Tâche, {task_id}

CloudWatch métrique	Métrique Apache Airflow	Unité	Dimension	
TaskInstanceQueuedDuration	jour. {dag_id}. {task_id} .queued_duration	Millisecondes	DAG, {day_id} Tâche, {task_id}	
	<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E1F5FE;"> <p> Note</p> <p>Disponible pour Apache Airflow v2.7.2 et versions ultérieures.</p> </div>			
TaskInstanceScheduledDuration	jour. {dag_id}. {task_id} .durée_planifiée	Millisecondes	DAG, {day_id} Tâche, {task_id}	
	<div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E1F5FE;"> <p> Note</p> <p>Disponible pour Apache Airflow v2.7.2 et versions ultérieures.</p> </div>			

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 lesquelles 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 à la `metrics.statsd_block_list` 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 Apache Airflow, vous pouvez surveiller les composants sous-jacents de vos flux de travail Amazon Managed Workflows pour les environnements Apache Airflow à l'aide d'un CloudWatch système qui collecte des données brutes et les traite en métriques lisibles quasiment en temps 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 mesures 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 envoient des notifications ou prennent 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

Espace de noms

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 paire nom-valeur qui fait partie de l'identité d'une métrique.

Unité

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 consulter les indicateurs de performance d'une dimension

1. Ouvrez la [page Metrics](#) sur la CloudWatch console.
2. Utilisez le sélecteur de AWS région pour sélectionner votre région.
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 Mesures graphiques pour afficher les mesures 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 consulter les 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, à chaque collaborateur de base, à chaque collaborateur supplémentaire et à chaque serveur Web. Pour plus d'informations et une description de chaque métrique de cluster, consultez la section [Mesures et dimensions disponibles](#) dans le manuel Amazon ECS Developer Guide.

Espace de noms	Mesure	Unité
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 sous 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. Vous pouvez le faire en représentant graphiquement le CPUUtilization ou la MemoryUtilization métrique et en définissant 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. Sous la dimension Cluster, pour le AdditionalWorker, choisissez soit le 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.

Web servers

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. Sous la dimension Cluster, pour le AdditionalWebServers, choisissez soit le 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, consultez 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.

Espace de noms	Mesure	Unité
AWS/MWAA	CPUUtilization	Pourcentage
AWS/MWAA	DatabaseConnections	Nombre
AWS/MWAA	DiskQueueDepth	Nombre
AWS/MWAA	FreeableMemory	Octets
AWS/MWAA	VolumeWriteIOPS	Comptez toutes les cinq minutes
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, consultez la section [CloudWatch Mesures disponibles pour Amazon SQS](#) dans le manuel Amazon Simple Queue Service Developer Guide.

Espace de noms	Mesure	Unité
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 métriques d'équilibreur de charge suivantes, consultez les [CloudWatch métriques de votre Application Load Balancer dans le Guide](#) de l'utilisateur des Application Load Balancers.

Espace de noms	Mesure	Unité
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 cela 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 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. Il vous explique comment configurer Amazon MWAA pour 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](#)
- [AWS Identity and Access Management](#)
- [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. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des AWS services que vous utilisez. Pour en savoir plus sur la confidentialité des données, consultez [Questions fréquentes \(FAQ\) sur la confidentialité des données](#). Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog [Modèle de responsabilité partagée AWS et RGPD \(Règlement général sur la protection des données\)](#) sur le Blog de sécuritéAWS .

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 AWS Identity and Access Management (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.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous recommandons TLS 1.2 ou 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 des AWS SDK. AWS CLI Toutes les données que vous saisissez dans des identifications ou des champs de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux 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 accorde à une clé KMS gérée par le client, consultez 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 des clés KMS AWS détenues ou gérées par le client sur Amazon MWAA. Pour plus d'informations, consultez [Tarification d'AWS KMS](#).

Artefacts de chiffrement

Vous spécifiez les artefacts de chiffrement utilisés pour le chiffrement au repos en spécifiant une [cléAWS 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 Logs sont chiffrés à CloudWatch l'aide du chiffrement côté serveur (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 du chiffrement côté serveur (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 du chiffrement côté serveur (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 susceptibles d'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 AWS services intégrés à Amazon

MWAA, tels qu'Amazon S3. Pour plus d'informations sur le chiffrement Amazon S3, consultez [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, consultez la section [Création de clés](#) dans le guide 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.region.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 annuler 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 AWS Fargate d'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": "mwaagrantsforenvmgmtrole-environment name",
    "GranteePrincipal": "airflow.region.amazonaws.com",
    "RetiringPrincipal": "airflow.region.amazonaws.com",
    "Operations": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ]
}
```

```
}

```

Subvention 2 : utilisée pour l'**ControllerLambdaExecutionRole**accès

```
{
  "Name": "mwa-grant-for-lambda-exec-environment name",
  "GranteePrincipal": "airflow.region.amazonaws.com",
  "RetiringPrincipal": "airflow.region.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Subvention 3 : utilisée pour l'**CfnManagementLambdaExecutionRole**accès

```
{
  "Name": " mwa-grant-for-cfn-mgmt-environment name",
  "GranteePrincipal": "airflow.region.amazonaws.com",
  "RetiringPrincipal": "airflow.region.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ]
}
```

Grant 4 : utilisé pour le rôle d'exécution Fargate pour accéder aux secrets du backend

```
{
  "Name": "mwa-fargate-access-for-environment name",
  "GranteePrincipal": "airflow.region.amazonaws.com",
  "RetiringPrincipal": "airflow.region.amazonaws.com",

```

```
    "Operations": [  
      "kms:Encrypt",  
      "kms:Decrypt",  
      "kms:ReEncrypt*",  
      "kms:GenerateDataKey*",  
      "kms:DescribeKey",  
      "kms:RetireGrant"  
    ]  
  }  
}
```

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 CloudWatch journaux chiffrés. Pour plus d'informations, consultez le service [Chiffrer les données du journal à CloudWatch l'aide 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-west-2.amazonaws.com"  
  },  
  "Action": [  
    "kms:Encrypt*",  
    "kms:Decrypt*",  
    "kms:ReEncrypt*",  
    "kms:GenerateDataKey*",  
    "kms:Describe*"  
  ],  
  "Resource": "*",  
  "Condition": {  
    "ArnLike": {  
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-west-2:*:*"  
    }  
  }  
}
```

```
    }  
  }  
}
```

AWS Identity and Access Management

AWS Identity and Access Management (IAM) est un AWS service qui permet à un administrateur de 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 AWS Identity and Access Management (IAM). Pour en savoir plus sur la gestion de l'accès à Amazon MWAA, consultez [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](#)
- [Autoriser des utilisateurs à afficher 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 AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Amazon MWAA.

Utilisateur du service — Si vous utilisez le service Amazon MWAA pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de plus en plus de fonctionnalités Amazon MWAA pour effectuer votre travail, il se peut que vous ayez besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne parvenez pas à accéder à une fonctionnalité d'Amazon MWAA, consultez [Résolution des problèmes liés à l'identité et à l'accès à Amazon Managed Workflows pour Apache Airflow](#).

Administrateur du service — Si vous êtes responsable des ressources Amazon MWAA au sein de votre entreprise, vous avez probablement un accès complet à Amazon MWAA. C'est à vous de déterminer les fonctionnalités et ressources Amazon MWAA auxquelles les utilisateurs de vos services doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser IAM avec Amazon MWAA, consultez. [Comment Amazon MWAA fonctionne avec IAM](#)

Administrateur IAM : si vous êtes administrateur IAM, vous souhaitez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à Amazon MWAA. Pour consulter des exemples de politiques basées sur l'identité Amazon MWAA que vous pouvez utiliser dans IAM, consultez. [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 (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section [Comment vous connecter à votre compte Compte AWS dans](#) le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section [Signature des demandes AWS d'API](#) dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Utilisation de l'authentification multifactorielle \(MFA\) dans l'interface AWS](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes AWS services les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez [Tâches nécessitant 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 au sein de votre Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez [Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification](#) dans le Guide de l'utilisateur IAM.

Un [groupe IAM](#) est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent

des informations d'identification temporaires. Pour en savoir plus, consultez [Quand créer un utilisateur IAM \(au lieu d'un rôle\)](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité au sein de votre Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console [changeant de rôle](#). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez [Utilisation de rôles IAM](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le 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 la rubrique [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains AWS services cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.
- Accès multiservices — Certains AWS services utilisent des fonctionnalités dans d'autres AWS services. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.

- **Sessions d'accès direct (FAS)** : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et AWS service, associées AWS service à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes AWS services ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez [Sessions de transmission d'accès](#).
- **Rôle de service** : il s'agit d'un [rôle IAM](#) attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un AWS service](#) dans le Guide de l'utilisateur IAM.
- **Rôle lié à un service** — Un rôle lié à un service est un type de rôle de service lié à un. AWS service Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- **Applications exécutées sur Amazon EC2** : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez [Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2](#) dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez [Quand créer un rôle IAM \(au lieu d'un utilisateur\)](#) 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 est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal

(utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

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.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Création de politiques IAM](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique 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. Des politiques basées sur les ressources sont, par exemple, les politiques de

confiance de rôle IAM et des politiques de compartiment. 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. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. AWS services

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 (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez [Vue d'ensemble des listes de contrôle d'accès \(ACL\)](#) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- **Limite d'autorisations** : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.

- **Politiques de contrôle des services (SCP)** — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée les comptes AWS multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chaque utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les organisations et les SCP, consultez [Fonctionnement des SCP](#) dans le Guide de l'utilisateur AWS Organizations .
- **Politiques de séance** : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez [politiques de séance](#) 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.

Autoriser des utilisateurs à afficher 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 vos AWS services 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, un utilisateur doit 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 ne dispose pas des autorisations nécessaires pour transférer 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 mon AWS compte 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 (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, 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, consultez [the section called "Accès à un environnement Amazon MWAA"](#)

Pour obtenir une vue d'ensemble de la manière dont Amazon MWAA et d'autres AWS services fonctionnent avec IAM, consultez 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 montrent 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 l'adresse <https://console.aws.amazon.com/iam/>.
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 politiques, 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 de plus amples 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, consultez [Références des éléments de politique 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. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

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 consulter la liste des actions Amazon MWAA, consultez 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 consulter les politiques d'Amazon MWAA, consultez [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 IAM basée sur l'identité à l'aide de ces exemples de documents de politique JSON, consultez [Création de politiques dans l'onglet JSON](#) dans le Guide de l'utilisateur IAM.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console Amazon MWAA](#)
- [Autoriser des utilisateurs à afficher 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.
- Accorder 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 de IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.

- Utiliser 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 AWS service, tel que AWS CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : IAM Access Analyzer 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 politique 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 le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Configuration de l'accès aux API protégé par 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 consulter les politiques d'Amazon MWAA, consultez [Gestion de l'accès à un environnement Amazon MWAA](#).

Autoriser des utilisateurs à afficher 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:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Validation de conformité pour les flux de travail gérés par Amazon pour Apache Airflow


Pour savoir si un [programme AWS services de conformité AWS service s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez AWS services 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 AWS services est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- [Guides de démarrage rapide sur la sécurité et la conformité](#) : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- [Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services](#) : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.

 Note

Tous ne AWS services sont pas éligibles à la loi HIPAA. Pour plus d'informations, consultez le [HIPAA Eligible Services Reference](#).

- AWS Ressources de <https://aws.amazon.com/compliance/resources/> de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- [AWS Guides de conformité destinés aux clients](#) — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation AWS services et décrivent les directives relatives aux contrôles de sécurité dans plusieurs cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- [Évaluation des ressources à l'aide des règles](#) du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- [AWS Security Hub](#)— Cela AWS service fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour

obtenir la liste des services et des contrôles pris en charge, consultez [Référence des contrôles Security Hub](#).

- [Amazon GuardDuty](#) — Cela AWS service détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- [AWS Audit Manager](#)— Cela vous AWS service permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans les flux de travail gérés par Amazon pour Apache Airflow

L'infrastructure AWS mondiale est construite autour des AWS régions et des 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 AWS les régions et les zones de disponibilité, consultez la section [Infrastructure AWS mondiale](#).

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.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

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. Vous devez vous assurer que les politiques d'accès appropriées sont utilisées pour vos VPC.

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 lors de l'élaboration et de la mise en œuvre de 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 Amazon S3 et les ACL d'objets accordent aux utilisateurs de l'environnement Amazon MWAA associé l'autorisation de 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 des DAG qui peuvent modifier les privilèges des utilisateurs d'Apache Airflow et 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 des métadonnées et des DAG de votre environnement.

- 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 dossier DAG d'Amazon S3. Utilisez plutôt les outils CI/CD pour écrire des DAG 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 des DAG sur la base d'un fichier de définition YAML, JSON ou autre stocké dans un emplacement distinct de votre compartiment Amazon MWAA Amazon S3 dans lequel vous stockez les DAG.
- Stockez les [secrets dans Secrets Manager](#). Cela n'empêchera pas les utilisateurs capables d'écrire des DAG de lire des secrets, mais cela les empêchera de modifier les secrets utilisés par votre environnement.

Détection des modifications apportées aux privilèges des utilisateurs d'Apache Airflow

Vous pouvez utiliser CloudWatch Logs Insights pour détecter les cas où des DAG modifient les 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 DAG modifie les privilèges utilisateur d'Apache Airflow.

Prérequis

Pour effectuer les étapes suivantes, vous aurez besoin des éléments suivants :

- Un environnement Amazon MWAA dans lequel tous les types de journaux Apache Airflow sont activés au niveau du INFO journal. Pour plus d'informations, consultez [the section called "Affichage des journaux de flux d'air"](#).

Pour configurer les notifications relatives aux modifications apportées aux privilèges utilisateur 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 MWAA (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 d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow

Cette page décrit les versions d'Apache Airflow prises en charge par Amazon Managed Workflows pour Apache Airflow et les stratégies que nous recommandons pour passer à 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](#)
- [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 partir d'Apache Airflow v2.2.2, Amazon MWAA prend en charge l'installation des exigences 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 instruction. `--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, consultez [Installation des dépendances Python](#).

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.9.2	Guide de référence d'Apache Airflow v2.9.2	Fichier de contraintes Apache Airflow v2.9.2	Python 3.11
v2.8.1	Guide de référence d'Apache Airflow v2.8.1	Fichier de contraintes Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guide de référence d'Apache Airflow v2.7.2	Fichier de contraintes Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guide de référence d'Apache Airflow v2.6.3	Fichier de contraintes Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guide de référence d'Apache Airflow v2.5.1	Fichier de contraintes Apache Airflow v2.5.1	Python 3.10

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.4.3	Guide de référence d'Apache Airflow v2.4.3	Fichier de contraintes Apache Airflow v2.4.3	Python 3.10
v2.2.2	Guide de référence d'Apache Airflow v2.2.2	Fichier de contraintes Apache Airflow v2.2.2	Python 3.7
v2.0.2	Guide de référence d'Apache Airflow v2.0.2	Fichier de contraintes Apache Airflow v2.0.2	Python 3.7

[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

Version d'Apache Airflow	Planificateur (par défaut)	Planificateur (min)	Planificateur (max)	
Apache Airflow v2 et versions ultérieures	2	2	5	

Workers

Version Airflow	Travailleurs (min)	Travailleurs (max.)	Travailleurs (par défaut)
Apache Airflow version 2	1	25	10

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 non vers une nouvelle version majeure, par exemple de `1.y.z` à `2.y.z`.

Note

Vous ne pouvez pas rétrograder la version d'Apache Airflow pour votre environnement.

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, consultez [the section called "Mise à niveau de la version"](#).

Versions obsolètes d'Apache Airflow

Le tableau suivant répertorie les versions obsolètes d'Apache Airflow dans Amazon MWAA, ainsi que les dates de publication initiale et de fin de support pour 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 de support limitée d'Amazon MWAA	Date de fin du support Amazon MWAA
v1.10.12	25 août 2020	24 novembre 2020	21 août 2023	21 février 2024

Version d'Apache Airflow	Date de sortie d'Apache Airflow	Date de disponibilité d'Amazon MWAA	Date de support limitée d'Amazon MWAA	Date de fin du support Amazon MWAA
v2.0.2	19 avril 2021	25 mai 2021	23 novembre 2023	29 avril 2024
v2.2.2	15 novembre 2021	27 janvier 2022	25 janvier 2024	27 juin 2024

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 90 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 mineure d'Apache Airflow pendant au moins 12 mois après sa première mise à disposition.

Q : Suis-je averti lorsque le support prend fin 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 AWS Health Dashboard de fin du support.

Q : Que se passera-t-il à la date d'assistance limitée ?

R : À la date de support limitée, vous ne pouvez plus créer de nouveaux environnements Amazon MWAA avec la version associée. Vos environnements existants resteront disponibles jusqu'à la date 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 pourrez toujours accéder à vos environnements Amazon MWAA existants qui exécutent la version obsolète associée d'Apache Airflow à vos propres risques. Pour obtenir des instructions sur la 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, voir [Shared responsibility in the AWS Well-Architected Framework](#).

Amazon Managed Workflows pour les points de terminaison et les 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, correspondent au nombre maximal de ressources ou d'opérations de service pour votre AWS compte.

Table des matières

- [Points de terminaison de service](#)
- [Quotas de service](#)
- [Augmenter les quotas](#)

Points de terminaison de service

Pour consulter la liste des points de terminaison pour Amazon MWAA, consultez [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](#)
 - [Que prend en charge Amazon MWAA pour Apache Airflow v2 ?](#)
 - [Pourquoi les anciennes versions d'Apache Airflow ne sont-elles pas prises en charge ?](#)
 - [Quelle version de Python dois-je utiliser ?](#)
 - [Quelle est la version d'pipAmazon MWAA utilisée ?](#)
- [Cas d'utilisation](#)
 - [Quand dois-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 l'Apache Airflow Worker ?](#)
 - [Puis-je spécifier plus de 25 travailleurs Apache Airflow ?](#)
 - [Amazon MWAA prend-il en charge les VPC Amazon partagés ou les sous-réseaux partagés ?](#)
- [Métriques](#)
 - [Quels indicateurs sont utilisés pour déterminer s'il convient de faire évoluer Workers ?](#)
 - [Puis-je créer des métriques personnalisées dans CloudWatch ?](#)
- [DAG, 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 ne s'affichent-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 \(DMS\) ?](#)

Versions prises en charge

Que prend en charge Amazon MWAA pour Apache Airflow v2 ?

Pour savoir ce que prend en charge Amazon MWAA, consultez [Versions d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow](#).

Pourquoi les anciennes versions d'Apache Airflow ne sont-elles pas prises en charge ?

Nous ne prenons en charge que la dernière version (au lancement) d'Apache Airflow, Apache Airflow v1.10.12, en raison de problèmes de sécurité liés aux anciennes versions.

Quelle version de Python dois-je utiliser ?

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

Note

- À partir d'Apache Airflow v2.2.2, Amazon MWAA prend en charge l'installation des exigences Python, des packages de fournisseurs et des plugins 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 instruction. `--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, consultez [Installation des dépendances Python](#).

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.9.2	Guide de référence d'Apache Airflow v2.9.2	Fichier de contraintes Apache Airflow v2.9.2	Python 3.11
v2.8.1	Guide de référence d'Apache Airflow v2.8.1	Fichier de contraintes Apache Airflow v2.8.1	Python 3.11
v2.7.2	Guide de référence d'Apache Airflow v2.7.2	Fichier de contraintes Apache Airflow v2.7.2	Python 3.11
v2.6.3	Guide de référence d'Apache Airflow v2.6.3	Fichier de contraintes Apache Airflow v2.6.3	Python 3.10
v2.5.1	Guide de référence d'Apache Airflow v2.5.1	Fichier de contraintes Apache Airflow v2.5.1	Python 3.10
v2.4.3	Guide de référence d'Apache Airflow v2.4.3	Fichier de contraintes Apache Airflow v2.4.3	Python 3.10
v2.2.2	Guide de référence d'Apache Airflow v2.2.2	Fichier de contraintes Apache Airflow v2.2.2	Python 3.7

Version d'Apache Airflow	Guide d'Apache Airflow	Contraintes d'Apache Airflow	Version Python
v2.0.2	Guide de référence d'Apache Airflow v2.0.2	Fichier de contraintes Apache Airflow v2.0.2	Python 3.7

[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.](#)

Quelle est la version d'**pip** Amazon MWAA utilisée ?

Pour les environnements exécutant Apache Airflow v1.10.12, Amazon MWAA installe la version 21.1.2. pip

Note

Amazon MWAA ne sera pas mis à niveau pip pour les environnements Apache Airflow v1.10.12.

Pour les environnements exécutant Apache Airflow v2 ou version ultérieure, Amazon MWAA installe pip la version 21.3.1.

Cas d'utilisation

Quand dois-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 10 Go et est spécifié par [Amazon ECS Fargate 1.3](#). La quantité de RAM est déterminée par la classe d'environnement que vous spécifiez. Pour plus d'informations sur les classes d'environnement, consultez [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 l'AMI Amazon Linux.

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, consultez [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, veuillez consulter la section [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 AWS Identity and Access Management, mais le filtrage de visibilité n'est pas disponible dans la AWS console. Si un utilisateur peut voir un environnement, il peut voir tous les environnements.

Puis-je stocker des données temporaires sur l'Apache Airflow Worker ?

Vos opérateurs Apache Airflow peuvent stocker des données temporaires sur les Workers. Les travailleurs 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 10 Go, selon [Amazon ECS Fargate 1.3](#). 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, consultez [Demande d'augmentation de quota](#).

Amazon MWAA prend-il en charge les VPC Amazon partagés ou les sous-réseaux partagés ?

Amazon MWAA ne prend pas en charge les VPC Amazon partagés ni les sous-réseaux partagés. Le VPC Amazon 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 pour voir un exemple de routage du trafic vers un Amazon VPC partagé, consultez la section [Routage sortant centralisé vers Internet dans le guide](#) des passerelles de transit Amazon VPC.

Métriques

Quels indicateurs sont utilisés pour déterminer s'il convient de faire évoluer Workers ?

Amazon MWAA surveille le QueuedTaskset l'entrée RunningTasks CloudWatch afin de déterminer s'il convient d'adapter Apache Airflow Workers à votre environnement. Pour en savoir plus, veuillez consulter la section [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 "Utilisation d'un DAG pour écrire des métriques personnalisées"](#).

DAG, 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, consultez [the section called "Plugin personnalisé à patcher PythonVirtualenvOperator"](#).

Combien de temps faut-il à Amazon MWAA pour reconnaître un nouveau fichier DAG ?

Les DAG 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 nouveaux DAG et 30 secondes pour les mises à jour des DAG existants, correspondent aux options de configuration d'Apache Airflow [dag_dir_list_interval](#), respectivement. [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, veuillez consulter la section [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, veuillez consulter la section [Créer un compartiment Amazon S3 pour Amazon S3 pour Amazon S3 pour Amazon S3](#).
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, veuillez consulter la section [Suppression de fichiers sur Amazon S3](#).

Pourquoi mes plugins ne s'affichent-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 présenté permet à Amazon MWAA d'authentifier vos utilisateurs Apache Airflow dans AWS Identity and Access Management (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 ou version

ultérieure. 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.

Amazon Managed Workflows for Apache Airflow

Cette rubrique décrit les problèmes et erreurs courants que vous pouvez rencontrer lors de l'utilisation d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow, ainsi que les étapes recommandées pour résoudre ces erreurs.

Table des matières

- [Résolution des problèmes : DAG, opérateurs, connexions et autres problèmes liés à Apache Airflow v2](#)
 - [Connexions](#)
 - [Je ne parviens pas à me connecter à Secrets Manager](#)
 - [Comment configurer les conditions du gestionnaire desecretsmanager:ResourceTag/<tag-key> secrets ou une restriction de ressources dans ma politique des rôles d'exécution ?](#)
 - [Je ne parviens pas à me connecter à Snowflake](#)
 - [Je ne vois pas ma connexion dans l'interface utilisateur Airflow](#)
 - [Serveur web](#)
 - [Je vois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche](#)
 - [Tâches](#)
 - [Je constate que mes tâches sont bloquées ou ne sont pas terminées](#)
 - [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)
 - [Pourquoi la commandedags backfill Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution ?](#)
 - [Opérateurs](#)
 - [J'ai reçu unePermissionError: \[Errno 13\] Permission denied erreur lors de l'utilisation de l'opérateur S3Transform](#)
- [Résolution des problèmes : DAG, opérateurs, connexions et autres problèmes liés à Apache Airflow v1](#)
 - [Mise à jour de requirements.txt](#)
 - [L'ajoutapache-airflow-providers-amazon entraîne l'échec de mon environnement](#)
 - [DAG cassé](#)

- [J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Amazon DynamoDB](#)
- [J'ai reçu l'erreur « Broken DAG : aucun module nommé psycpg2 »](#)
- [J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Slack](#)
- [J'ai reçu diverses erreurs lors de l'installation de Google/GCP/BigQuery](#)
- [J'ai reçu l'erreur « Broken DAG : No module named Cython »](#)
- [Opérateurs](#)
 - [J'ai reçu un message d'erreur en utilisant l'BigQueryopérateur](#)
- [Connexions](#)
 - [Je ne parviens pas à me connecter à Snowflake](#)
 - [Je ne parviens pas à me connecter à Secrets Manager](#)
 - [Je n'arrive pas à me connecter à mon serveur MySQL sur '<DB-identifiant-name>.cluster-id.
« <region>.rds.amazonaws.com »](#)
- [Serveur web](#)
 - [J'utilise leBigQueryOperator et cela provoque le blocage de mon serveur Web](#)
 - [Je vois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche](#)
- [Tâches](#)
 - [Je constate que mes tâches sont bloquées ou ne sont pas terminées](#)
- [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)
- [Résolution : créer et mettre à jour un environnement Amazon MWAA : créer et mettre à jour un environnement Amazon](#)
- [Mise à jour des requirements.txt](#)
 - [J'ai spécifié une nouvelle version de monrequirements.txt et la mise à jour de mon environnement prend plus de 20 minutes](#)
- [Plugins](#)
 - [Amazon MWAA prend-il en charge la mise en œuvre d'une interface utilisateur personnalisée ?](#)
 - [Je suis capable d'implémenter des modifications personnalisées de l'interface utilisateur sur le lanceur local Amazon MWAA via des plugins, mais lorsque j'essaie de faire de même](#)

sur Amazon MWAA, je ne vois pas mes modifications ni aucune erreur. Pourquoi cela se produit-il ?

- Créer un compartiment.
 - Je n'arrive pas à sélectionner l'option pour les paramètres de blocage public S3
- Créer un environnement
 - J'ai essayé de créer un environnement et il est bloqué dans l'état « Création »
 - J'ai essayé de créer un environnement mais le statut est « Échec de la création »
 - J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur « Défaillance réseau »
 - J'ai essayé de créer un environnement et j'ai reçu un message d'erreur « doit être transmis » à un service, à une partition ou à une ressource
 - J'ai essayé de créer un environnement et l'état est « Disponible », mais lorsque j'essaie d'accéder à l'interface utilisateur d'Airflow, une erreur « Réponse vide du serveur » ou « 502 Mauvaise passerelle » s'affiche
 - J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires
- Environnement de mise à jour
 - 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 Apache Airflow
- Résolution des problèmes : CloudWatch journaux et CloudTrail erreurs
 - Journaux
 - Je ne peux pas voir mes journaux de tâches ou j'ai reçu le message d'erreur « Lire le journal distant depuis Cloudwatch log_group »
 - Les tâches échouent sans aucun journal
 - Je vois une erreur ResourceAlreadyExistsException « » dans CloudTrail
 - Je vois une erreur « Demande non valide » dans CloudTrail
 - Je vois un message « Impossible de localiser une bibliothèque client Oracle 64 bits : « libcintsh.so : impossible d'ouvrir un fichier objet partagé : aucun fichier ou répertoire de ce type » dans les journaux d'Apache Airflow
 - Je vois que le serveur de psycopg2 a fermé la connexion de façon inattendue dans les journaux de mon planificateur

- [Je vois « L'exécuteur signale que l'instance de tâche %s est terminée \(%s\) alors que la tâche indique %s » dans les journaux de traitement de mon DAG](#)
- [Je vois « Impossible de lire les journaux distants depuis log_group : airflow-^{*}{*EnvironmentName}-Task log_stream :^{*} {*DAG_ID} /^{*} {*TASK_ID} /^{*} {*time} /^{*} {*n} .log. » dans mes journaux de tâches](#)

Résolution des problèmes : DAG, opérateurs, connexions et autres problèmes liés à Apache Airflow v2

Les rubriques de cette page décrivent les solutions aux dépendances Python d'Apache Airflow v2, aux plugins personnalisés, aux DAG, aux opérateurs, aux connexions, 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 ne parviens pas à me connecter à Secrets Manager](#)
 - [Comment configurer les conditions du gestionnaire desecretsmanager:ResourceTag/<tag-key> secrets ou une restriction de ressources dans ma politique des rôles d'exécution ?](#)
 - [Je ne parviens pas à me connecter à Snowflake](#)
 - [Je ne vois pas ma connexion dans l'interface utilisateur Airflow](#)
- [Serveur web](#)
 - [Je vois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche](#)
- [Tâches](#)
 - [Je constate que mes tâches sont bloquées ou ne sont pas terminées](#)
- [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)
 - [Pourquoi la commandedags backfill Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution ?](#)
- [Opérateurs](#)
 - [J'ai reçu unePermissionError: \[Errno 13\] 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 d'une autre AWS base de données.

Je ne parviens pas à me connecter à Secrets Manager

Nous vous recommandons la procédure suivante :

1. Apprenez à 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 dans 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 dans AWS Secrets Manager pour une connexion Apache Airflow](#).

Comment configurer les conditions du gestionnaire `desecretsmanager:ResourceTag/<tag-key>` secrets ou une restriction de ressources dans ma politique des rôles 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 conditionnelles ou d'autres restrictions de ressources dans le rôle d'exécution de votre environnement, en raison d'un problème connu dans Apache Airflow.

Je ne parviens pas à me connecter à Snowflake

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) 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 de connexion : `snowflake_conn`
2. Type de cône : `Snowflake`
3. Hôte : `<my account>. <my region if not us-west-2>.snowflakecomputing.com`
4. Schéma : `<my schema>`
5. Login : `<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='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...     schema='YOUR_SCHEMA'
...     login='YOUR_USERNAME',
...     password='YOUR_PASSWORD',
...     port='YOUR_PORT'
...     extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
... database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Je ne vois pas ma connexion dans l'interface utilisateur 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 à l'aide du modèle de connexion HTTP.

Nous vous recommandons la procédure suivante :

1. Consultez les 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. Consultez les 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 de l'interface utilisateur Apache Airflow de manière interchangeable avec les types de connexion qui ne sont pas disponibles dans l'interface utilisateur Apache Airflow sur Amazon MWAA à l'adresse [Présentation des types de connexion 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 vois une erreur 5xx lors de l'accès au serveur Web

Nous vous recommandons la procédure suivante :

1. Vérifiez Apache Airflow configuration options. Vérifiez que les paires clé-valeur que vous avez spécifiées comme option de configuration d'Apache Airflow, telles que AWS Secrets Manager, ont été correctement configurées. Pour en savoir plus, consultez [the section called “Je ne parviens 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 vôtre `requirements.txt` sont compatibles avec votre version d'Apache Airflow.
3. Découvrez les moyens de spécifier les dépendances Python dans un `requirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche

Si le planificateur ne semble pas fonctionner ou si le dernier « battement de cœur » a été reçu il y a plusieurs heures, il est possible que vos DAG n'apparaissent pas dans Apache Airflow et que de nouvelles tâches ne soient pas planifiées.

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 et l'erreur devrait disparaître. Pour en savoir plus, consultez [the section called "Sécurité dans votre VPC"](#).

Note

- La métadatabase Aurora PostgreSQL fait partie de [l'architecture de service Amazon MWAA](#) et n'est pas visible dans votre Compte AWS.
- Les erreurs liées à la base de données sont généralement le symptôme d'une défaillance du planificateur et non la cause première.

2. Si le planificateur ne fonctionne pas, cela peut être dû à un certain nombre de facteurs, tels que des [échecs d'installation liés à une dépendance](#) ou une [surcharge du planificateur](#). Vérifiez que vos DAG, plug-ins et exigences fonctionnent correctement en consultant les groupes de CloudWatch journaux correspondants dans Logs. Pour en savoir plus, consultez [Surveillance et mesures](#).

Tâches

La rubrique suivante décrit les erreurs que vous pouvez recevoir pour les tâches Apache Airflow dans un environnement.

Je constate que 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. Un grand nombre de DAG peuvent être définis. Réduisez le nombre de DAG et effectuez une mise à jour de l'environnement (par exemple en modifiant un niveau de journalisation) pour forcer une réinitialisation.
 - a. Airflow analyse les DAG, qu'ils soient 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 CloudWatch les métriques ou de longs temps de traitement des DAG dans CloudWatch les journaux. Il existe d'autres méthodes d'optimisation des configurations Apache Airflow qui ne sont pas présentées dans ce guide.
 - b. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, consultez [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 à l'état « Aucune », ou par un grand nombre dans Tâches en file d'attente et/ou Tâches en attente CloudWatch. Plusieurs raisons sont possibles :
 - a. S'il y a plus de tâches à exécuter que ce que l'environnement a la capacité d'exécuter, et/ou qu'un grand nombre de tâches étaient mises en file d'attente avant que l'autoscaling ait le temps de détecter les tâches et de déployer des Workers supplémentaires.
 - b. Si le nombre de tâches à exécuter est supérieur à la capacité d'un environnement, nous vous recommandons de réduire le nombre de tâches que vos DAG exécutent simultanément et/ou d'augmenter le nombre minimum d'Apache Airflow Workers.
 - c. Si un grand nombre de tâches ont été mises en file d'attente avant que l'autoscaling n'ait eu le temps de détecter et de déployer des opérateurs supplémentaires, nous recommandons d'échelonner le déploiement des tâches et/ou d'augmenter 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 Workers exécutés sur votre environnement.

```
aws mwaas 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, consultez [the section called "Optimisation des performances pour Apache Airflow"](#).

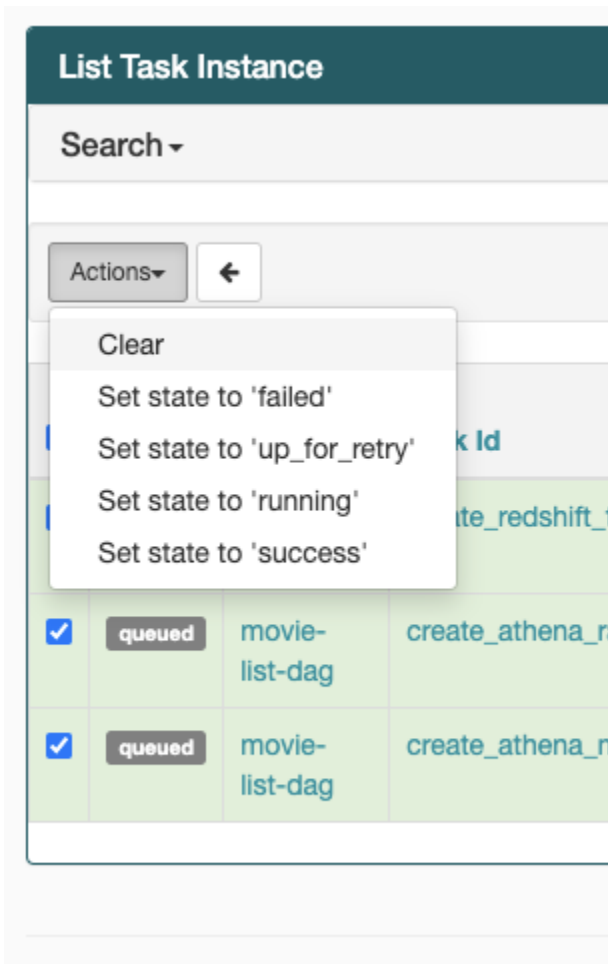
3. Certaines tâches supprimées en cours d'exécution apparaissent sous forme de journaux de tâches qui s'arrêtent sans autre indication dans Apache Airflow. Plusieurs raisons sont possibles :
 - a. S'il y a un bref moment où 1) les tâches en cours dépassent la capacité actuelle de l'environnement, suivi par 2) quelques minutes sans exécution ou mise en file d'attente, 3) de nouvelles tâches sont mises en file d'attente.
 - b. L'autoscaling Amazon MWAA réagit au premier scénario en ajoutant des travailleurs supplémentaires. Dans le second scénario, les travailleurs supplémentaires sont supprimés. Certaines des tâches mises en file d'attente peuvent entraîner la suppression des opérateurs et se termineront lorsque le conteneur sera supprimé.
 - c. Nous vous recommandons d'augmenter le nombre minimum de travailleurs sur votre environnement. Une autre option consiste à ajuster le calendrier de vos DAG et de vos tâches pour vous assurer que ces scénarios ne se produisent pas.
 - d. Vous pouvez également définir le nombre minimum de travailleurs égal au nombre maximum de travailleurs de votre environnement, désactivant ainsi la mise à l'échelle automatique. Utilisez la commande [update-environment](#) dans le AWS Command Line Interface (AWS CLI) pour désactiver la mise à l'échelle automatique en définissant le même nombre minimum et maximum de travailleurs.

```
aws mwaa update-environment --name MyEnvironmentName --min-workers 5 --max-workers 5
```

- e. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, consultez [the section called "Optimisation des performances pour Apache Airflow"](#).
4. 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 exécutés sur votre environnement. L'image suivante montre un exemple de tâche isolée.



- Choisissez le cercle correspondant à la tâche isolée, puis sélectionnez Effacer (comme indiqué). Cela permet à Amazon MWAA de réduire la taille du personnel ; sinon, Amazon MWAA ne peut pas déterminer quels DAG sont activés ou désactivés, et ne peut pas réduire la taille des tâches s'il reste des tâches en file d'attente.



5. Pour en savoir plus sur le cycle de vie des tâches Apache Airflow, consultez la section [Concepts](#) du guide de référence d'Apache Airflow.

INTERFACE DE LIGNE DE COMMANDE (CLI)

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'exécution des commandes Airflow CLI dans le AWS Command Line Interface.

Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI

L'interface de ligne de commande Airflow s'exécute sur le serveur Web Apache Airflow, dont la simultanéité est limitée. En général, un maximum de 4 commandes CLI peuvent être exécutées simultanément.

Pourquoi la commande `airflow dags backfill` Apache Airflow CLI échoue-t-elle ? Existe-t-il une solution ?

Note

Ce qui suit s'applique uniquement aux environnements Apache Airflow v2.0.2.

La commande `airflow dags backfill`, comme les autres commandes CLI d'Apache Airflow, analyse tous les DAG localement avant que les DAG ne soient traités, quel que soit le DAG auquel s'applique l'opération CLI. Dans les environnements Amazon MWAA utilisant Apache Airflow v2.0.2, étant donné que les plug-ins et la configuration requise ne sont pas encore installés sur le serveur Web au moment de l'exécution de la commande CLI, l'opération d'analyse échoue et n'est pas appelée. Si vous n'aviez aucune exigence ni aucun plug-in dans votre environnement, l'opération réussirait.

Pour pouvoir exécuter la commande `airflow dags backfill` CLI, nous vous recommandons de l'invoquer dans un opérateur bash. Dans un opérateur bash, `airflow dags backfill` est lancé par le travailleur, ce qui permet aux DAG d'effectuer une analyse réussie lorsque toutes les exigences et tous les plug-ins nécessaires sont disponibles et installés. L'exemple suivant montre comment créer un DAG avec un `BashOperator` à exécuter `airflow dags 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`, consultez [Installation de plugins personnalisés](#).

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) sur GitHub.

2. Créez votre script de « transformation ».

```
#!/bin/bash
cp $1 $2
```

3. (facultatif) Les utilisateurs macOS et Linux peuvent avoir besoin d'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 [Charger le fichier plugins.zip vers Amazon S3](#).
6. Suivez les étapes décrites dans [Spécification de la version 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://YOUR_S3_BUCKET/files/input.txt',  
    dest_s3_key='s3://YOUR_S3_BUCKET/files/output.txt'  
)
```

8. Suivez les étapes décrites dans [Chargement du code DAG vers Amazon S3](#).

Résolution des problèmes : DAG, opérateurs, connexions et autres problèmes liés à Apache Airflow v1

Les rubriques de cette page contiennent des résolutions concernant les dépendances Python d'Apache Airflow v1.10.12, les extensions personnalisées, les DAG, les opérateurs, les connexions, les tâches et les problèmes de serveur Web que vous pouvez rencontrer dans un environnement Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [Mise à jour de requirements.txt](#)
 - [L'ajoutapache-airflow-providers-amazon entraîne l'échec de mon environnement](#)
- [DAG cassé](#)
 - [J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Amazon DynamoDB](#)
 - [J'ai reçu l'erreur « Broken DAG : aucun module nommé pycopg2 »](#)
 - [J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Slack](#)
 - [J'ai reçu diverses erreurs lors de l'installation de Google/GCP/BigQuery](#)
 - [J'ai reçu l'erreur « Broken DAG : No module named Cython »](#)
- [Opérateurs](#)
 - [J'ai reçu un message d'erreur en utilisant l'BigQueryopérateur](#)
- [Connexions](#)
 - [Je ne parviens pas à me connecter à Snowflake](#)
 - [Je ne parviens pas à me connecter à Secrets Manager](#)
 - [Je n'arrive pas à me connecter à mon serveur MySQL sur '<DB-identifiant-name>.cluster-id.< <region>.rds.amazonaws.com »](#)

- [Serveur web](#)
 - [J'utilise leBigQueryOperator et cela provoque le blocage de mon serveur Web](#)
 - [Je vois une erreur 5xx lors de l'accès au serveur Web](#)
 - [Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche](#)
- [Tâches](#)
 - [Je constate que mes tâches sont bloquées ou ne sont pas terminées](#)
- [INTERFACE DE LIGNE DE COMMANDE \(CLI\)](#)
 - [Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI](#)

Mise à jour de requirements.txt

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la mise à jour de `votrequirements.txt`.

L'ajout **apache-airflow-providers-amazon** entraîne l'échec de mon environnement

`apache-airflow-providers-xyzest` uniquement compatible avec Apache Airflow v2. `apache-airflow-backport-providers-xyzest` compatible avec Apache Airflow 1.10.12.

DAG cassé

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'exécution de DAG.

J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Amazon DynamoDB

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Ajoutez le package suivant à `votrequirements.txt`.

```
boto
```

3. Découvrez les moyens de spécifier les dépendances Python dans un `requirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

J'ai reçu l'erreur « Broken DAG : aucun module nommé psycopg2 »

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Ajoutez ce qui suit `requirements.txt` à votre version d'Apache Airflow. Par exemple :

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

3. Découvrez les moyens de spécifier les dépendances Python dans `unrequirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

J'ai reçu un message d'erreur « Broken DAG » lors de l'utilisation des opérateurs Slack

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Ajoutez le package suivant à votre `requirements.txt` et spécifiez votre version d'Apache Airflow. Par exemple :

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

3. Découvrez les moyens de spécifier les dépendances Python dans `unrequirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

J'ai reçu diverses erreurs lors de l'installation de Google/GCP/BigQuery

Amazon MWAA utilise Amazon Linux qui nécessite une version spécifique de Cython et des bibliothèques de cryptographie. Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Ajoutez le package suivant à votre `requirements.txt`.

```
grpcio==1.27.2  
cython==0.29.21
```

```
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow-backport-providers-amazon[google]
```

3. Si vous n'utilisez pas de fournisseurs de backport, vous pouvez utiliser :

```
grpcio==1.27.2
cython==0.29.21
pandas-gbq==0.13.3
cryptography==3.3.2
apache-airflow[gcp]==1.10.12
```

4. Découvrez les moyens de spécifier les dépendances Python dans `unrequirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

J'ai reçu l'erreur « Broken DAG : No module named Cython »

Amazon MWAA utilise Amazon Linux qui nécessite une version spécifique de Cython. Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) sur GitHub.
2. Ajoutez le package suivant à votre `requirements.txt`.

```
cython==0.29.21
```

3. Les bibliothèques Cython ont besoin de différentes versions de dépendance pip. Par exemple, l'utilisation de `aws wrangler==2.4.0` requiert `pyarrow<3.1.0, >=2.0.0`, pip3 essaie d'installer `pyarrow==3.0.0` ce qui provoque une erreur Broken DAG. Nous vous recommandons de spécifier explicitement la version acceptable la plus ancienne. Par exemple, si vous spécifiez la valeur minimale `pyarrow==2.0.0` avant `aws wrangler==2.4.0`, l'erreur disparaît et l'installation `requirements.txt` s'effectue correctement. Les exigences finales doivent se présenter comme suit :

```
cython==0.29.21
pyarrow==2.0.0
aws wrangler==2.4.0
```

4. Découvrez les moyens de spécifier les dépendances Python dans `unrequirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

Opérateurs

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'utilisation des opérateurs.

J'ai reçu un message d'erreur en utilisant l'BigQueryopérateur

Amazon MWAA ne prend pas en charge les opérateurs dotés d'extensions d'interface utilisateur. Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Une solution consiste à remplacer l'extension en ajoutant une ligne dans le DAG à définir `<operator name>.operator_extra_links = None` après avoir importé les opérateurs problématiques. Par exemple :

```
from airflow.contrib.operators.bigquery_operator import BigQueryOperator
BigQueryOperator.operator_extra_links = None
```

3. Vous pouvez utiliser cette approche pour tous les DAG en ajoutant ce qui précède à un plugin. Pour voir un exemple, consultez [the section called "Plugin personnalisé à patcherPythonVirtualenvOperator"](#).

Connexions

La rubrique suivante décrit les erreurs que vous pouvez recevoir lorsque vous utilisez une connexion Apache Airflow ou une autre AWS base de données.

Je ne parviens pas à me connecter à Snowflake

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) on GitHub.
2. Ajoutez les entrées suivantes au fichier `requirements.txt` correspondant à votre environnement.

```
asn1crypto == 0.24.0
snowflake-connector-python == 1.7.2
```

3. Ajoutez les importations suivantes à votre DAG :


```
from airflow.contrib.hooks.snowflake_hook import SnowflakeHook
from airflow.contrib.operators.snowflake_operator import SnowflakeOperator
```

Assurez-vous que l'objet de connexion Apache Airflow inclut les paires clé/valeur suivantes :

1. Identifiant de connexion : snowflake_conn
2. Type de cône : Flocon de neige
3. Hôte : <my account>. <my region if not us-west-2>.snowflakecomputing.com
4. Schéma : <my schema>
5. Login : <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='YOUR_ACCOUNT.YOUR_REGION.snowflakecomputing.com',
...     schema='YOUR_SCHEMA',
...     login='YOUR_USERNAME',
...     password='YOUR_PASSWORD',
...     port='YOUR_PORT',
...     extra=json.dumps(dict(account='YOUR_ACCOUNT', warehouse='YOUR_WAREHOUSE',
... database='YOUR_DB_OPTION', region='YOUR_REGION')),
... )
```

Je ne parviens pas à me connecter à Secrets Manager

Nous vous recommandons la procédure suivante :

1. Apprenez à 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 dans 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 dans AWS Secrets Manager pour une connexion Apache Airflow](#).

Je n'arrive pas à me connecter à mon serveur MySQL sur '`<DB-identifier-name>.cluster-id. <region>.rds.amazonaws.com`' »

Le groupe de sécurité Amazon MWAA et le groupe de sécurité RDS ont besoin d'une règle d'entrée pour autoriser le trafic entrant et sortant l'un de l'autre. Nous vous recommandons la procédure suivante :

1. Modifiez le groupe de sécurité RDS pour autoriser tout le trafic provenant du groupe de sécurité VPC d'Amazon MWAA.
2. Modifiez le groupe de sécurité VPC d'Amazon MWAA pour autoriser tout le trafic provenant du groupe de sécurité RDS.
3. Réexécutez vos tâches et vérifiez si la requête SQL a réussi en consultant les journaux Apache Airflow dans CloudWatch les journaux.

Serveur web

La rubrique suivante décrit les erreurs que vous pouvez recevoir pour votre serveur Web Apache Airflow sur Amazon MWAA.

J'utilise le `BigQueryOperator` et cela provoque le blocage de mon serveur Web

Nous vous recommandons la procédure suivante :

1. Les opérateurs Apache Airflow tels que `BigQueryOperator` et `QuboleOperator` qui contiennent `operator_extra_links` peuvent provoquer le blocage de votre serveur Web

Apache Airflow. Ces opérateurs tentent de charger du code sur votre serveur Web, ce qui n'est pas autorisé pour des raisons de sécurité. Nous vous recommandons de corriger les opérateurs de votre DAG en ajoutant le code suivant après vos instructions d'importation :

```
BigQueryOperator.operator_extra_links = None
```

2. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) sur GitHub.

Je vois 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 comme option de configuration d'Apache Airflow, telles que AWS Secrets Manager, ont été correctement configurées. Pour en savoir plus, consultez [the section called “Je ne parviens 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 vôtre `requirements.txt` sont compatibles avec votre version d'Apache Airflow.
3. Découvrez les moyens de spécifier les dépendances Python dans un `requirements.txt` fichier, voir [Gestion des dépendances Python dans requirements.txt](#).

Le message d'erreur « Le planificateur ne semble pas être en cours d'exécution » s'affiche

Si le planificateur ne semble pas fonctionner ou si le dernier « battement de cœur » a été reçu il y a plusieurs heures, il est possible que vos DAG n'apparaissent pas dans Apache Airflow et que de nouvelles tâches ne soient pas planifiées.

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 et l'erreur devrait disparaître. Pour en savoir plus, consultez [the section called “Sécurité dans votre VPC”](#).

Note

- La métadatabase Aurora PostgreSQL fait partie de l'[architecture de service Amazon MWAA](#) et n'est pas visible dans votre Compte AWS.
- Les erreurs liées à la base de données sont généralement le symptôme d'une défaillance du planificateur et non la cause première.

2. Si le planificateur ne fonctionne pas, cela peut être dû à un certain nombre de facteurs, tels que des [échecs d'installation liés à une dépendance](#) ou une [surcharge du planificateur](#). Vérifiez que vos DAG, plug-ins et exigences fonctionnent correctement en consultant les groupes de CloudWatch journaux correspondants dans Logs. Pour en savoir plus, consultez [Surveillance et mesures](#).

Tâches

La rubrique suivante décrit les erreurs que vous pouvez recevoir pour les tâches Apache Airflow dans un environnement.

Je constate que 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. Un grand nombre de DAG peuvent être définis. Réduisez le nombre de DAG et effectuez une mise à jour de l'environnement (par exemple en modifiant un niveau de journalisation) pour forcer une réinitialisation.
 - a. Airflow analyse les DAG, qu'ils soient 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 CloudWatch les métriques ou de longs temps de traitement des DAG dans CloudWatch les journaux. D'autres méthodes d'optimisation des configurations Apache Airflow ne sont pas présentées dans ce guide.
 - b. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, consultez [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 à l'état « Aucune », ou par un grand nombre dans Tâches en file d'attente et/ou Tâches en attenteCloudWatch. Plusieurs raisons sont possibles :
 - a. S'il y a plus de tâches à exécuter que ce que l'environnement a la capacité d'exécuter, et/ou qu'un grand nombre de tâches étaient mises en file d'attente avant l'autoscaling a le temps de détecter les tâches et de déployer des Workers supplémentaires.
 - b. S'il y a plus de tâches à exécuter qu'un environnement ne peut en exécuter, nous vous recommandons de réduire le nombre de tâches que vos DAG exécutent simultanément et/ou d'augmenter le nombre minimum d'Apache Airflow Workers.
 - c. Si un grand nombre de tâches ont été mises en file d'attente avant que l'autoscaling n'ait eu le temps de détecter et de déployer des opérateurs supplémentaires, nous recommandons d'échelonner le déploiement des tâches et/ou d'augmenter le nombre minimum de travailleurs Apache Airflow.
 - d. Vous pouvez utiliser la commande [update-environment](#) dans leAWS Command Line Interface (AWS CLI) pour modifier le nombre minimum ou maximum de Workers exécutés sur votre environnement.

```
aws mwaas 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, consultez [the section called "Optimisation des performances pour Apache Airflow"](#).
3. Certaines tâches supprimées en cours d'exécution apparaissent sous forme de journaux de tâches qui s'arrêtent sans autre indication dans Apache Airflow. Plusieurs raisons sont possibles :
 - a. S'il y a un bref moment où 1) les tâches en cours dépassent la capacité actuelle de l'environnement, suivi par 2) quelques minutes sans exécution ou mise en file d'attente, 3) de nouvelles tâches sont mises en file d'attente.
 - b. La mise à l'échelle automatique Amazon MWAA réagit au premier scénario en ajoutant des travailleurs supplémentaires. Dans le second scénario, les travailleurs supplémentaires sont supprimés. Certaines des tâches mises en file d'attente peuvent entraîner la suppression des opérateurs et se termineront lorsque le conteneur sera supprimé.

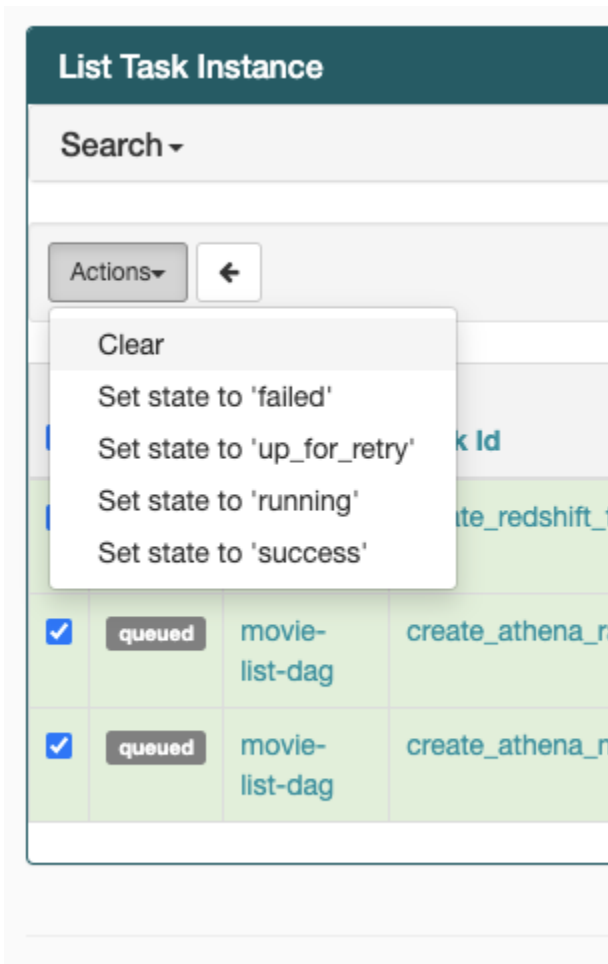
- c. Nous vous recommandons d'augmenter le nombre minimum de travailleurs sur votre environnement. Une autre option consiste à ajuster le calendrier de vos DAG et de vos tâches pour vous assurer que ces scénarios ne se produisent pas.
- d. Vous pouvez également définir le nombre minimum de travailleurs égal au nombre maximum de travailleurs de votre environnement, désactivant ainsi la mise à l'échelle automatique. Utilisez la commande [update-environment](#) dans le AWS Command Line Interface (AWS CLI) pour désactiver la mise à l'échelle automatique en définissant le même nombre minimum et maximum de travailleurs.

```
aws mwaas update-environment --name MyEnvironmentName --min-workers 5 --max-workers 5
```

- e. Pour en savoir plus sur les meilleures pratiques que nous recommandons pour optimiser les performances de votre environnement, consultez [the section called "Optimisation des performances pour Apache Airflow"](#).
4. 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 exécutés sur votre environnement. L'image suivante montre un exemple de tâche autre que ne sont pas présentées.



- Choisissez le cercle correspondant à la tâche isolée, puis sélectionnez Effacer (comme indiqué). Cela permet à Amazon MWAA de réduire la taille du personnel ; sinon, Amazon MWAA ne peut pas déterminer quels DAG sont activés ou désactivés, et ne peut pas réduire la taille des tâches s'il reste des tâches en file d'attente.



5. Pour en savoir plus sur le cycle de vie des tâches Apache Airflow, consultez la section [Concepts](#) du guide de référence d'Apache Airflow.

INTERFACE DE LIGNE DE COMMANDE (CLI)

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'exécution des commandes Airflow CLI dans leAWS Command Line Interface.

Je vois une erreur « 503 » lors du déclenchement d'un DAG dans la CLI

L'interface de ligne de commande Airflow s'exécute sur le serveur Web Apache Airflow, dont la simultanéité est limitée. En général, un maximum de 4 commandes CLI peuvent être exécutées simultanément.

Résolution : créer et mettre à jour un environnement Amazon MWAA : créer et mettre à jour un environnement Amazon

Les rubriques de cette page contiennent les 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 expliquent comment les résoudre.

Table des matières

- [Mise à jour des requirements.txt](#)
 - [J'ai spécifié une nouvelle version de monrequirements.txt et la mise à jour de mon environnement prend plus de 20 minutes](#)
- [Plugins](#)
 - [Amazon MWAA prend-il en charge la mise en œuvre d'une interface utilisateur personnalisée ?](#)
 - [Je suis capable d'implémenter des modifications personnalisées de l'interface utilisateur sur le lanceur local Amazon MWAA via des plugins, mais lorsque j'essaie de faire de même sur Amazon MWAA, je ne vois pas mes modifications ni aucune erreur. Pourquoi cela se produit-il ?](#)
- [Créer un compartiment.](#)
 - [Je n'arrive pas à sélectionner l'option pour les paramètres de blocage public S3](#)
- [Créer un environnement](#)
 - [J'ai essayé de créer un environnement et il est bloqué dans l'état « Création »](#)
 - [J'ai essayé de créer un environnement mais le statut est « Échec de la création »](#)
 - [J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur « Défaillance réseau »](#)
 - [J'ai essayé de créer un environnement et j'ai reçu un message d'erreur « doit être transmis » à un service, à une partition ou à une ressource](#)
 - [J'ai essayé de créer un environnement et l'état est « Disponible », mais lorsque j'essaie d'accéder à l'interface utilisateur d'Airflow, une erreur « Réponse vide du serveur » ou « 502 Mauvaise passerelle » s'affiche](#)
 - [J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires](#)
- [Environnement de mise à jour](#)
 - [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 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 `requirements.txt` et la mise à jour de mon environnement prend plus de 20 minutes

S'il faut plus de vingt minutes à votre environnement pour installer une nouvelle version d'un `requirements.txt` fichier, la mise à jour de l'environnement a échoué 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 Groupes de journaux](#) de la CloudWatch console. Si vous voyez des journaux vides, la raison la plus courante est l'absence d'autorisations dans votre rôle d'exécution CloudWatch ou dans Amazon S3 où les journaux sont écrits. Pour en savoir plus, consultez [Rôle d'exécution](#).
3. Vérifiez les options de configuration 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, consultez [the section called "Configuration de Secrets Manager"](#).
4. Vérifiez la configuration du réseau VPC. Pour en savoir plus, consultez [the section called "Environnement bloqué"](#).
5. Vérifiez les autorisations des rôles d'exécution. Un rôle d'exécution est un rôle AWS Identity and Access Management (IAM) doté d'une politique d'autorisations qui autorise Amazon MAA à invoquer les ressources d'autres AWS services (tels qu'Amazon S3 CloudWatch, Amazon SQS, 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, consultez [Rôle d'exécution](#).
6. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils de AWS support sur GitHub.

Plugins

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 d'Apache Airflow v2.2.2, Amazon MWAA prend en charge l'installation de plugins 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, consultez [Versions](#).

Je suis capable d'implémenter des modifications personnalisées de l'interface utilisateur sur le [lanceur local Amazon MWAA](#) via des plugins, mais lorsque j'essaie de faire de même sur Amazon MWAA, je ne vois pas mes modifications ni aucune erreur. Pourquoi cela se produit-il ?

le générateur local Amazon MWAA regroupe tous les composants d'Apache Airflow dans une seule image, ce qui vous permet d'appliquer des modifications personnalisées au plugin d'interface utilisateur.

Créez un compartiment.

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de la création d'un compartiment Amazon S3.

Je n'arrive pas à sélectionner l'option pour les paramètres de blocage public S3

Le [rôle d'exécution](#) de votre environnement Amazon MWAA doit être autorisé à effectuer l'`GetBucketPublicAccessBlock`action sur le compartiment Amazon S3 afin de vérifier que le compartiment a bloqué l'accès public. Nous vous recommandons la procédure suivante :

1. Suivez les étapes pour [associer une politique JSON à votre rôle d'exécution](#).
2. Attachez la politique JSON suivante :

```
{
```

```
"Effect": "Allow",
"Action": [
  "s3:GetObject*",
  "s3:GetBucket*",
  "s3:List*"
],
"Resource": [
  "arn:aws:s3:::YOUR_S3_BUCKET_NAME",
  "arn:aws:s3:::YOUR_S3_BUCKET_NAME/*"
]
}
```

Remplacez les exemples d'espaces réservés dans *YOUR_S3_BUCKET_NAME* par le nom de votre compartiment Amazon S3, tel que *my-mwaa-unique-s3-bucket-name*.

3. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils de AWS support 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 bloqué dans l'état « Création »

Nous vous recommandons la procédure suivante :

1. Vérifiez le réseau VPC avec routage public. Si vous utilisez un Amazon VPC avec accès à Internet, vérifiez les points suivants :
 - Que votre Amazon VPC est 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 selon une règle d'autoréférencement, soit spécifier éventuellement la plage de ports pour la plage de ports HTTPS 443 et une plage de ports TCP 5432.
2. Vérifiez le réseau VPC avec 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 différentesAWS 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 une instance NAT), ni de passerelle Internet.
3. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils deAWS support surGitHub.

J'ai essayé de créer un environnement mais le statut est « Échec de la création »

Nous vous recommandons la procédure suivante :

1. Vérifiez la configuration du réseau VPC. Pour en savoir plus, consultez [the section called “Environnement bloqué”](#).
2. Vérifiez les autorisations des utilisateurs. Amazon MWAA effectue un essai à sec sur la base des informations d'identification de l'utilisateur avant de créer un environnement. VotreAWS compte n'est peut-être pas autorisé dansAWS Identity and Access Management (IAM) à créer certaines ressources pour un environnement. Par exemple, si vous avez choisi le mode d'accès au réseau privé Apache Airflow, votre administrateur doit avoir autorisé votreAWS compte à accéder à la politique de contrôle d'FullConsoleAccessaccès [AmazonMWAA](#) 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 rôleAWS Identity and Access Management (IAM) doté d'une politique d'autorisations qui autorise Amazon MAA à invoquer les ressources d'autresAWS services (tels qu'Amazon S3CloudWatch, Amazon SQS, 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, consultez [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 Groupes de journaux](#) de laCloudWatch console. Si vous voyez des journaux vides, la raison la plus courante est l'absence d'autorisations dans votre rôle d'exécutionCloudWatch ou dans Amazon S3 où les journaux sont écrits. Pour en savoir plus, consultez [Rôle d'exécution](#).
5. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils deAWS support surGitHub.

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 les ressources suivantes :
 - [Création d'un réseau Amazon VPC sans accès à Internet](#)
 - [Créez le point de terminaison de Amazon S3 Elastic Container Container Container Container Registry](#) :

J'ai essayé de sélectionner un VPC et j'ai reçu un message d'erreur « Défaillance réseau »

Nous vous recommandons la procédure suivante :

- Si un message d'erreur « Défaillance réseau » s'affiche lorsque vous essayez de sélectionner un Amazon VPC lors de la création de votre environnement, désactivez tous les proxys en cours d'exécution dans le navigateur, puis réessayez.

J'ai essayé de créer un environnement et j'ai reçu un message d'erreur « doit être transmis » à un service, à une partition ou à une ressource

Nous vous recommandons la procédure suivante :

- Vous recevez peut-être cette erreur parce que 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://your-bucket-name
```

J'ai essayé de créer un environnement et l'état est « Disponible », mais lorsque j'essaie d'accéder à l'interface utilisateur d'Airflow, une erreur « Réponse vide du serveur » ou « 502 Mauvaise passerelle » s'affiche

Nous vous recommandons la procédure suivante :

1. Vérifiez la configuration du groupe de sécurité VPC. Pour en savoir plus, consultez [the section called "Environnement bloqué"](#).

2. Vérifiez que tous les packages Apache Airflow que vous avez répertoriés dans `requirements.txt` correspondent à la version d'Apache Airflow que vous utilisez sur Amazon MWAA. Pour en savoir plus, consultez [Installation des dépendances Python](#).
3. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils de AWS support sur GitHub.

J'ai essayé de créer un environnement et mon nom d'utilisateur est un tas de noms de personnages aléatoires

- Apache Airflow dispose d'un maximum de 64 caractères pour les noms d'utilisateur. Si votre rôle AWS Identity and Access Management (IAM) dépasse cette longueur, un algorithme de hachage est utilisé pour le réduire, tout en restant unique.

Environnement de mise à jour

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 classe d'environnement différente (par exemple, si vous remplacez une par `mw1.small`) et que la demande de mise à jour de votre environnement échoue, `UPDATE_FAILED` l'état de l'environnement passe à un état et l'environnement est rétabli et facturé en fonction de la version stable précédente d'un environnement `mw1.medium`

Nous vous recommandons la procédure suivante :

1. Testez vos DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de [aws-mwaa-local-runner](#) sur GitHub.
2. Pour exécuter un script de résolution des problèmes qui vérifie l'installation et la configuration du réseau Amazon VPC pour votre environnement Amazon MWAA, consultez le script [Verify Environment](#) dans les outils de AWS support 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 Apache Airflow

Nous vous recommandons la procédure suivante :

1. Vérifiez les autorisations des utilisateurs. Vous n'avez peut-être pas accès à une politique d'autorisation vous permettant de consulter l'interface utilisateur d'Apache Airflow. Pour en savoir plus, consultez [the section called “Accès à un environnement Amazon MWAA”](#).
2. Vérifiez l'accès au réseau. Cela est 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-vpce.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 réseau public, soit créer un mécanisme pour accéder au point de terminaison VPC de votre serveur Web Apache Airflow. Pour en savoir plus, consultez la section [the section called “Gestion de l'accès aux points de terminaison VPC”](#).

Résolution des problèmes : CloudWatch journaux et CloudTrail erreurs

Les rubriques de cette page contiennent des solutions à Amazon CloudWatch Logs et des AWS CloudTrail erreurs que vous pouvez rencontrer dans un environnement Amazon Managed Workflows pour Apache Airflow.

Table des matières

- [Journaux](#)
 - [Je ne peux pas voir mes journaux de tâches ou j'ai reçu le message d'erreur « Lire le journal distant depuis Cloudwatch log_group »](#)
 - [Les tâches échouent sans aucun journal](#)
 - [Je vois une erreur ResourceAlreadyExistsException « » dans CloudTrail](#)
 - [Je vois une erreur « Demande non valide » dans CloudTrail](#)
 - [Je vois un message « Impossible de localiser une bibliothèque client Oracle 64 bits : « libcintsh.so : impossible d'ouvrir un fichier objet partagé : aucun fichier ou répertoire de ce type » dans les journaux d'Apache Airflow](#)
 - [Je vois que le serveur de psycopg2 a fermé la connexion de façon inattendue dans les journaux de mon planificateur](#)

- [Je vois « L'exécuteur signale que l'instance de tâche %s est terminée \(%s\) alors que la tâche indique %s » dans les journaux de traitement de mon DAG](#)
- [Je vois « Impossible de lire les journaux distants depuis log_group : airflow-^{*}{*EnvironmentName} -Task log_stream :^{*} {^{*}DAG_ID} /^{*} {^{*}TASK_ID} /^{*} {^{*}time} /^{*} {^{*}n} .log. » dans mes journaux de tâches](#)

Journaux

La rubrique suivante décrit les erreurs que vous pouvez recevoir lors de l'affichage des journaux Apache Airflow.

Je ne peux pas voir mes journaux de tâches ou j'ai reçu le message d'erreur « Lire le journal distant depuis Cloudwatch log_group »

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, le message d'erreur suivant s'affichera :

```
*** 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, veuillez consulter [Afficher les 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, dispose de suffisamment de ressources pour analyser le DAG et dispose des bibliothèques Python appropriées à charger. 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 l'outil [Amazon MWAA local-runner](#).

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 vois une erreur ResourceAlreadyExistsException « » 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 utilisée par Amazon MWAA CloudWatch pour communiquer. Nous vous recommandons la procédure suivante :

- Ajoutez la bibliothèque suivante à votre `requirements.txt`

```
watchtower==1.0.6
```

Je vois une erreur « Demande non valide » 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 AWS CloudFormation modèle, vous devez ajouter une `DependsOn` section dans votre AWS CloudFormation modèle. Les deux ressources (MWAA Environment et MWAA Execution Policy) sont dépendantes de. AWS CloudFormation Nous vous recommandons la procédure suivante :

- Ajoutez la **DependsOn** déclaration suivante à votre AWS 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 voir un exemple, consultez [Tutoriel de démarrage rapide pour Amazon Managed Workflows pour Apache Airflow](#).

Je vois un message « Impossible de localiser une bibliothèque client Oracle 64 bits : « libclntsh.so : impossible d'ouvrir un fichier objet partagé : aucun fichier ou répertoire de ce type » dans les journaux 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, consultez la section [Utilisation des options de configuration pour charger des plugins en 2](#).

Je vois que le serveur de psycopg2 a fermé la connexion de façon inattendue dans les journaux de mon planificateur

Si une erreur similaire à la suivante s'affiche, 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, qui vous permet de spécifier jusqu'à 5 planificateurs.

Je vois « L'exécuteur signale que l'instance de tâche %s est terminée (%s) alors que la tâche indique %s » dans les journaux de traitement de mon DAG

Si un message d'erreur similaire à ce qui suit s'affiche, 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 Apache Airflow Sensors interroge pour savoir quand il est terminé. Si un capteur tombe en panne, il peut être réessayé en toute sécurité sans affecter les fonctionnalités de l'opérateur.

Je vois « Impossible de lire les journaux distants depuis log_group : airflow-`{*EnvironmentName}` -Task log_stream :`{*DAG_ID} /{*TASK_ID} /{*time} /{*n}` .log. » dans mes journaux de tâches

Si une erreur similaire à la suivante s'affiche, le rôle d'exécution de votre environnement ne contient peut-être 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 DAG, vos plugins personnalisés et vos dépendances Python localement à l'aide de l'option [aws-mwaa-local-runner](#) on GitHub.

Historique du document 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
Amazon MWAA prend en charge la configuration de noms de domaine de serveur Web personnalisés	<p>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.</p> <ul style="list-style-type: none">• the section called “Configuration d'un domaine personnalisé”	18 juin 2024
Amazon MWAA prend en charge le dimensionnement automatique du serveur Web et l'API REST Apache Airflow	<p>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.</p> <ul style="list-style-type: none">• the section called “Configuration de la mise à l'échelle automatique du serveur Web”	16 mai 2024

<u>Description améliorée du comportement de mise à l'échelle automatique</u>	<ul style="list-style-type: none">• <u>the section called "Utilisation de l'API REST Apache Airflow"</u>	10 mai 2024
<u>Support pour des instances de plus grande taille</u>	<p>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.</p> <ul style="list-style-type: none">• <u>the section called "Configuration de la mise à l'échelle automatique des travailleurs"</u> <p>Amazon MWAA prend désormais en charge deux options de taille d'instance plus importante pour les charges de travail plus importantes : <code>mw1.xlarge</code> et <code>mw1.2xlarge</code></p> <ul style="list-style-type: none">• <u>the section called "Capacités environnementales"</u>	16 avril 2024

[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 pour les connexions Apache Airflow v2.8.1”](#)

[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 pour les connexions Apache Airflow v2.7.2”](#)

[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, ainsi que des informations sur l'utilisation d'Apache Airflow v2.6.3 sur Amazon MWAA,

9 août 2023

- [Versions](#)
- [the section called “Packages de fournisseurs pour les connexions Apache Airflow v2.6.3”](#)

[Informations relatives à la dépréciation des versions](#)

Rubrique mise à jour sur la dépréciation des versions afin d'inclure les avis de dépréciation et les chronologies 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 “Mise à niveau de la version”](#)

Rubrique mise à jour

Politiques IAM gérées par le client mises à jour 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` afin de permettre à un utilisateur de 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 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.

12 avril 2023

- [Création du 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, ainsi que des informations sur l'utilisation d'Apache Airflow v2.5.1 sur Amazon MWAA,

11 avril 2023

- [Versions](#)
- [the section called “Packages de fournisseurs pour les connexions Apache Airflow v2.5.1”](#)

[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 à un serveur Web privé, vous devez utiliser une archive Python Wheel (.whl) pour empaqueter et installer les dépendances.

24 février 2023

- [Mode d'accès au serveur Web privé](#)

[Informations ajoutées sur les versions obsolètes d'Apache Airflow](#)

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](#).

17 février 2023

- [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](#)

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.

20 janvier 2023

- [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](#)
- [the section called “Packages de fournisseurs pour les connexions Apache Airflow v2.4.3”](#)

[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 sous 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 sous l'espace de AWS/MWAA noms. Il s'agit notamment des indicateurs de cluster Amazon ECS, des planificateurs, des travailleurs et des serveurs Web, des mesures Amazon SQS pour les files d'attente qui permettent à Amazon MWAA de dissocier les planificateurs et les 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 “Spécifier les nouveaux packages de fournisseurs”](#)

Entrée de FAQ mise à jour	<p>Informations mises à jour relatives à l'éligibilité d'Amazon MWAA à la loi HIPAA.</p> <ul style="list-style-type: none">• the section called “Conformité à la loi américaine HIPAA”	15 novembre 2022
Nouvelle rubrique	<p>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.</p> <ul style="list-style-type: none">• the section called “Prévention du cas de figure de l'adjoint désorienté entre services”	21 octobre 2022
Nouvel exemple de code	<p>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</p> <ul style="list-style-type: none">• the section called “Utilisation d'un DAG pour écrire des métriques personnalisées”	13 septembre 2022

[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.

12 septembre 2022

- [the section called “Invoquer des DAG avec Lambda”](#)

[Nouveaux schémas architecturaux](#)

Ajout de nouveaux diagrammes architecturaux illustrant un environnement Amazon MWAA avec un serveur Web public et privé.

12 septembre 2022

- [the section called “Modes d'accès à Apache Airflow”](#)

[Nouvel exemple de code](#)

Ajout d'instructions mises à jour et d'un nouvel exemple de code DAG qui récupère un jeton de CLI Apache Airflow, puis invoque un autre DAG dans un autre environnement Amazon MWAA.

16 août 2022

- [the section called “Invoquer des DAG dans différents environnements”](#)

[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 “Rafraîchissant AWS CodeArtifact jeton au moment 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 du 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. <ul style="list-style-type: none">• 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. <ul style="list-style-type: none">• 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 Python Wheel pour les environnements Amazon MWAA avec accès public et privé. <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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 créer des DAG sensibles au fuseau horaire à 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 “Changer le fuseau horaire d'un DAG”](#)

[Lancement d'Apache Airflow v2.2.2](#)

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, consultez les rubriques suivantes.

27 janvier 2022

- [Versions d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow](#).
- [the section called “Packages de fournisseurs pour les connexions Apache Airflow v2.2.2”](#).
- 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 du rôle à un utilisateur Apache Airflow mappé depuis IAM afin de limiter l'accès de l'utilisateur à un sous-ensemble de DAG spécifiés.

8 décembre 2021

- [the section called “Tutoriel : Restreindre les utilisateurs à un sous-ensemble de DAG”](#)

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

- [Optimisation des performances de la façon dont le planificateur analyse les DAG](#)
- [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 DAG sont traités à l'aide d'un plugin personnalisé, et d'une nouvelle rubrique de résolution des problèmes pour appeler la commande de la CLI `dags backfill` Apache Airflow depuis un opérateur bash.

1er novembre 2021

- [the section called “Changer le fuseau horaire d'un DAG”](#)
- [Remplir la commande 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 les rubriques suivantes :

24 septembre 2021

- [the section called “Packages de fournisseurs pour les connexions Apache Airflow v2.0.2”](#).

[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 du ECSOperator"](#).
- Ajout de nouvelles rubriques de résolution des problèmes liés à la configuration des plugins Apache Airflow dans [the section called "Plugins"](#).

[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 rubriques suivantes :

- [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 AWS CloudFormation modèle qui utilisait une règle d'autoréférence entrant pour le groupe de sécurité dans. [Création du réseau VPC](#)
- Correction du AWS CloudFormation modèle qui utilisait une règle d'autoréférence 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

- Ajout du décorateur DAG à la liste des fonctionnalités prises en charge par Apache Airflow v2.0.2. [Versions d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow](#)

[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 les quotas du service Apache Airflow](#).
- Prérequis de mise en réseau clarifiés sur la base des commentaires des utilisateurs sur. [Démarrez 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 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](#)
- Ajout du résumé des modifications apportées à la version 2.0.2 depuis la page des versions d'Airflow en [Installation des dépendances Python](#) fonction des commentaires des utilisateurs.
- Ajout du résumé des modifications apportées à la version 2.0.2 depuis la page des versions d'Airflow en [Référence des commandes de la CLI Apache Airflow](#) fonction des commentaires des utilisateurs.
- Ajout du résumé des modifications apportées à la version 2.0.2 depuis la page des versions d'Airflow en [Présentation des types de connexion des types de connexion](#) fonction des commentaires des utilisateurs.
- Ajout du résumé des modifications apportées à la version 2.0.2 depuis la

page des versions d'Airflow en [Installation de plugins personnalisés](#) fonction des commentaires des utilisateurs.

- Ajout du résumé des modifications apportées à la version 2.0.2 depuis la page des versions d'Airflow en [Ajout ou mise à jour des DAG](#) 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 [Invoquer des DAG 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 vois pas ma connexion dans l'interface utilisateur Airflow » à l'adresse. [Amazon Managed Workflows for Apache Airflow](#)
- Ajout d'une liste des Amazon VPC pris en charge par Amazon MWAA. [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 [Amazon Managed Workflows for 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 v2 dans CloudWatch](#) dans le menu de navigation de surveillance et de mesures.

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 [Afficher les journaux d'audit AWS CloudTrail](#).
- Créé [Afficher les 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 ajout du paramètre 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 [Amazon Managed Workflows for 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 affichés dans le menu Admin > Plugins d'Airflow 2.0 à [l'Questions fréquemment posées sur Amazon MWAA](#)adresse.

Nouveaux guides	Les modifications suivantes ont été ajoutées :	9 juillet 2021
	<ul style="list-style-type: none">• Créé Suppression de fichiers sur Amazon S3.	
Nouveaux sujets et cas d'utilisation	Les modifications suivantes ont été ajoutées :	2 juillet 2021
	<ul style="list-style-type: none">• 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
	<ul style="list-style-type: none">• Ajout d'un exemple de code Apache Airflow v1.10.12 pour 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 v2 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 MAAA avec Amazon EKS](#)

[Nouveaux guides](#)

Les modifications suivantes ont été ajoutées :

25 juin 2021

- Créé [Optimisation des performances pour Apache Airflow sur Amazon MAAA](#).

[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 AWS 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 Apache Airflow Workers 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 [Amazon Managed Workflows for Apache Airflow](#)
- Ajout d'une rubrique pour le journal « Le serveur a fermé la connexion de façon inattendue » à [Amazon](#)

[Managed Workflows for 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. [Amazon Managed Workflows for Apache Airflow](#)
- Ajout d'une rubrique pour une erreur 503 lors de l'exécution d'un DAG dans la CLI pour [Amazon Managed Workflows for 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 `core.lazy_load_plugins` :
False charger les plugins au début de chaque processus Airflow 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 les plugins Apache Airflow v2.0.2, exemple de code sur. [Création d'un plugin personnalisé qui génère des variables d'environnement d'exécution](#)
- 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 AirflowPythonVirtualenvOperator](#)
- 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. [Utilisation d'une clé secrète dansAWS Secrets Managerpour 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. [Créer un compartiment Amazon S3 pour Amazon S3 pour Amazon S3 pour Amazon S3](#)
- 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 Apache Airflow à [Questions fréquemment posées sur Amazon MWAA](#).
- Ajout d'une question concernant les métriques utilisées pour déterminer s'il faut adapter Apache Airflow Workers 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 à l'aide de 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 Cloud Watch](#)

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. [Présentation des types de connexion des types de connexion](#)

Correctifs

Les modifications suivantes ont été ajoutées : 2 juin 2021

- Ajout de points de terminaison VPC Apache Airflow au modèle dans AWS CloudFormation 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 d'Apache Airflow sur Amazon Managed Workflows pour Apache Airflow](#).
- Créé [Métriques de l'environnement Apache Airflow v2 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 dansAWS Secrets Managerpour une connexion Apache Airflow](#)
- Ajout d'un exemple de code Apache Airflow v2.0.2 à. [Création d'un plugin personnalisé pour Apache AirflowPythonVirtu alenvOperator](#)
- Ajout de commandes Apache Airflow v2.0.2 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 la dernière version d'Apache Airflow par défaut 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. [Amazon Managed Workflows for Apache Airflow](#)

[Correctifs](#)

Les modifications suivantes ont été ajoutées :

12 mai 2021

- Nous avons mis à jour l'exemple de code des plugins pour utiliser la dernière version de Java dans [Création d'un plugin personnalisé avec Apache Hive et Hadoop](#). Auparavant, c'était le cas `environment["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 [Amazon Managed Workflows for 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 [Utilisation des DAG 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 haut niveau 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 de la CLI Airflow 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 MWAA CLI a été ajouté aux [Installation des dépendances Python](#) pages [Ajout ou mise à jour des DAG](#) [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 dans 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

- Oups ! 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 `apache-airflow-providers-amazon`, qui est uniquement compatible avec Apache Airflow 2.0 pour [Amazon Managed Workflows for 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 dans 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 AWS 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](#).
- Modification du nom de la page d'accès au réseau en modes d'accès Apache Airflow en fonction des commentaires des utilisateurs, et rationalisation de la documentation dans [Modes d'accès à 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. [Amazon Managed](#)

[Workflows for 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 dans [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 dans [Création d'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

- Oups ! 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 [Amazon Managed Workflows for 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 résolution des problèmes a été ajouté comme prochaines étapes pour toutes les rubriques relatives au VPC et à la configuration de l'environnement à l'adresse. [Amazon Managed Workflows for 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 d'un guide des connexions Apache Airflow pour AWS Secrets Manager [atConfiguration d'une connexion Apache Airflow à l'aide d'un secret AWS Secrets Manager.](#)
- Création d'un didacticiel de démarrage rapide à l'aide d'un AWS 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 [Amazon Managed Workflows for 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 de la rubrique de résolution des problèmes BigQuery Google/GC P/ à [Amazon Managed Workflows for Apache Airflow](#)
- Ajout de la rubrique de résolution des problèmes liés à Cython à [Amazon Managed Workflows for Apache Airflow](#)
- Ajout d'une rubrique de résolution des problèmes MySQL à [Amazon Managed Workflows for Apache Airflow](#).
- Ajout de la rubrique de résolution des erreurs du serveur Web 5xx à [Amazon Managed Workflows for Apache Airflow](#).

Désormais pris en charge

Les modifications suivantes ont été ajoutées :

4 mars 2021

- Auparavant, `backend_kwargs` 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. Consultez la rubrique consacrée à la AWS Secrets Manager résolution des problèmes dans [Amazon Managed Workflows for Apache Airflow](#).

Correctifs

Les modifications suivantes ont été ajoutées :

4 mars 2021

- Oups ! 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 à Apache Airflow](#)
- Des vérifications supplémentaires ont été ajoutées pour la rubrique de résolution des problèmes liés à la création d'un environnement à [Amazon Managed Workflows for Apache Airflow](#).
- Ajout d'étapes pour afficher les 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 [Installation des dépendances Python](#) l'adresse.
- Ajout de la rubrique de résolution des problèmes de mise à [Amazon Managed Workflows for 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 à 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 [Invoquer des DAG 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 [Utilisation des DAG sur Amazon MWAA](#) pages.
- Apache Airflow ne prend pas en charge les DAG sérialisés dans les commandes 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 `backfill` des commandes non prises en charge à l'adresse. [Accès à Apache Airflow](#)

[GitHub lancement](#)

Les documents du guide de l'utilisateur sont désormais open source sur GitHub. Choisissez « Modifier cette page sur 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 [Amazon Managed Workflows for Apache Airflow](#).
- Ajout de la politique d'accès à la CLI 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 [Démarrez 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 MWAA 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 [Amazon Managed Workflows for Apache Airflow](#).
- Ajout d'une rubrique de AWS Secrets Manager résolution des problèmes à [Amazon Managed Workflows for 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
- AWS 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.