

CodeArtifact Guide de l'utilisateur

CodeArtifact



CodeArtifact: CodeArtifact Guide de l'utilisateur

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

Les marques et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service qui n'appartient pas à Amazon, de toute manière susceptible de créer une confusion chez les clients ou de toute manière dénigrant ou discréditant 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 que c'est AWS CodeArtifact ?	1
Comment CodeArtifact fonctionne ?	1
Concepts	2
Ressource	2
Domaine	3
Référentiel	3
Package	3
Groupe de packages	4
Espace de noms du package	4
Version du package	4
Révision de la version du package	4
Référentiel en amont	5
Comment puis-je commencer CodeArtifact ?	5
Configuration	6
Inscrivez-vous pour AWS	6
Installez ou mettez à niveau, puis configurez le AWS CLI	7
Approvisionner un utilisateur IAM	8
Installez votre gestionnaire de packages ou votre outil de compilation	10
Étapes suivantes	10
Premiers pas	12
Prérequis	12
Mise en route à l'aide de la console	13
Commencer à utiliser le AWS CLI	15
Utilisation des référentiels	23
Création d'un référentiel	23
Création d'un référentiel (console)	24
Création d'un référentiel (AWS CLI)	25
Création d'un référentiel avec un référentiel en amont	26
Connexion à un référentiel	27
Utiliser un client de gestion de packages	28
Supprimer un dépôt	28
Supprimer un dépôt (console)	28
Supprimer un dépôt (AWS CLI)	29
Empêcher la suppression des référentiels	29

Répertorier les référentiels	31
Répertorier les référentiels d'un compte AWS	31
Répertorier les référentiels du domaine	32
Afficher ou modifier la configuration d'un référentiel	34
Afficher ou modifier la configuration d'un référentiel (console)	34
Afficher ou modifier la configuration d'un référentiel (AWS CLI)	36
Politiques de référentiel	38
Créez une politique de ressources pour accorder l'accès en lecture	38
Définissez une politique	40
Lire une politique	41
Supprimer une politique	42
Accorder un accès en lecture aux principaux	42
Accorder l'accès en écriture aux packages	43
Accorder l'accès en écriture à un dépôt	44
Interaction entre le référentiel et les politiques de domaine	45
Marquer un dépôt	46
Référentiels de balises (CLI)	46
Référentiels de balises (console)	50
Utilisation de référentiels en amont	54
Quelle est la différence entre les référentiels en amont et les connexions externes ?	54
Ajouter ou supprimer des référentiels en amont	55
Ajouter ou supprimer des référentiels en amont (console)	55
Ajouter ou supprimer des référentiels en amont (AWS CLI)	56
Connect un CodeArtifact dépôt à un dépôt public	59
Connect à un référentiel externe (console)	59
Se connecter à un référentiel externe (CLI)	60
Référentiels de connexions externes pris en charge	62
Supprimer une connexion externe (CLI)	63
Demande d'une version de package avec des référentiels en amont	63
Conservation des packages depuis les référentiels en amont	64
Récupérez des packages via une relation en amont	64
Rétention des packages dans des référentiels intermédiaires	67
Demande de packages à partir de connexions externes	68
Récupérer des packages depuis une connexion externe	68
Latence de connexion externe	70
CodeArtifact comportement lorsqu'un référentiel externe n'est pas disponible	70

Disponibilité des nouvelles versions du package	71
Importation de versions de packages contenant plusieurs actifs	72
Ordre de priorité du référentiel en amont	72
Exemple d'ordre de priorité simple	73
Exemple d'ordre de priorité complexe	74
Comportement des API avec les référentiels en amont	75
Utilisation des packages	77
Vue d'ensemble des packages	77
Formats de packages pris en charge	78
Publication de packages	78
État de la version du package	81
Nom du package, version du package et normalisation du nom des actifs	82
Lister les noms de packages	83
Répertorier les noms des packages npm	84
Répertorier les noms des packages Maven	86
Répertorier les noms des paquets Python	87
Filtrer par préfixe de nom de package	87
Combinaisons d'options de recherche prises en charge	88
Formater la sortie	88
Valeurs par défaut et autres options	89
Lister les versions des packages	89
Répertorier les versions du package npm	91
Répertorier les versions du package Maven	92
Trier les versions	92
Version d'affichage par défaut	93
Formater la sortie	93
Répertorier les actifs de la version	94
Lister les actifs d'un package npm	95
Lister les actifs d'un package Maven	95
Télécharger les ressources de la version du package	96
Copier des packages entre des référentiels	97
Autorisations IAM requises pour copier des packages	97
Copier les versions du package	99
Copier un package depuis des référentiels en amont	100
Copier un package npm délimité	100
Copier les versions du package Maven	100

Versions qui n'existent pas dans le référentiel source	101
Versions qui existent déjà dans le référentiel de destination	101
Spécification d'une révision de version de package	103
Copier les packages npm	104
Supprimer un package ou une version de package	104
Suppression d'un package (AWS CLI)	105
Suppression d'un package (console)	106
Supprimer une version de package (AWS CLI)	106
Suppression d'une version de package (console)	107
Supprimer un package npm ou une version de package	107
Supprimer un package Maven ou une version de package	108
Bonnes pratiques pour supprimer des packages ou des versions de packages	108
Afficher et mettre à jour les détails et les dépendances des versions du package	109
Afficher les détails de la version du package	109
Afficher les détails de la version du package npm	110
Afficher les détails de la version du package Maven	112
Afficher les dépendances entre les versions du package	113
Afficher le fichier readme de la version du package	114
État de version du package de mise à jour	114
Mettre à jour le statut de version du package	115
Autorisations IAM requises pour mettre à jour l'état de la version d'un package	116
Mise à jour de l'état d'un package npm délimité	117
Mettre à jour le statut d'un package Maven	117
Spécification d'une révision de version de package	117
Utilisation du paramètre d'état attendu	118
Erreurs liées aux différentes versions de package	119
Élimination des versions du package	120
Modification des contrôles d'origine des packages	123
Scénarios courants de contrôle d'accès aux packages	123
Paramètres de contrôle de l'origine des packages	125
Paramètres de contrôle de l'origine des packages par défaut	126
Comment les contrôles d'origine des packages interagissent avec les contrôles d'origine des groupes de packages	127
Modification des contrôles d'origine des packages	128
Publication et référentiels en amont	129
Utilisation de groupes de packages	131

Création d'un groupe de packages	132
Création d'un groupe de packages (console)	132
Création d'un groupe de packages (AWS CLI)	134
Afficher ou modifier un groupe de packages	134
Afficher ou modifier un groupe de packages (console)	134
Afficher ou modifier un groupe de packages (AWS CLI)	135
Supprimer un groupe de packages	136
Supprimer un groupe de packages (console)	137
Supprimer un groupe de packages (AWS CLI)	137
Contrôles d'origine des groupes de packages	137
Paramètres de restriction	138
Listes de référentiels autorisés	140
Modification des paramètres de contrôle d'origine des groupes de packages	140
Exemples de configuration du contrôle d'origine des groupes de packages	142
Comment les paramètres de contrôle d'origine des groupes de packages interagissent avec les paramètres de contrôle de l'origine des packages	144
Syntaxe de définition du groupe de packages et comportement de correspondance	144
Syntaxe et exemples de définition de groupes de packages	145
Hiérarchie des groupes de packages et spécificité du modèle	146
Mots, limites de mots et correspondance de préfixes	147
Sensibilité à la casse	148
Match fort et faible	148
Variations supplémentaires	149
Marquer un groupe de packages	150
Groupes de packages de balises (CLI)	150
Utilisation des domaines	154
Vue d'ensemble du domaine	154
Domaines multi-comptes	155
Types de AWS KMS clés pris en charge dans CodeArtifact	156
Création d'un domaine	157
Création d'un domaine (console)	157
Création d'un domaine (AWS CLI)	158
Exemple de politique AWS KMS clé	159
Supprimer un domaine	160
Restrictions relatives à la suppression de domaines	161
Supprimer un domaine (console)	161

Supprimer un domaine (AWS CLI)	162
Politiques de domaine	162
Activer l'accès multicompte à un domaine	163
Exemple de politique de domaine	165
Exemple de politique de domaine avec AWS Organizations	166
Définissez une politique de domaine	167
Lire une politique de domaine	168
Supprimer une politique de domaine	168
Marquer un domaine	169
Domaines de balises (CLI)	169
Domaines de balises (console)	172
Utilisation du fret	176
Configuration et utilisation de Cargo	176
Configurez Cargo avec CodeArtifact	176
Installation de caisses de chargement	181
Publication de caisses Cargo	182
Assistance au commandement du fret	182
Commandes prises en charge qui nécessitent l'accès au registre	183
Commandes non prises en charge	183
Utilisation de Maven	184
Utiliser CodeArtifact avec Gradle	184
Récupérer les dépendances	185
Plug-ins de récupération	186
Publier des artefacts	187
Exécuter une compilation Gradle dans IntelliJ IDEA	189
Utiliser CodeArtifact avec MVN	193
Extraire les dépendances	185
Publier des artefacts	187
Publier des artefacts tiers	198
Restreindre les téléchargements de dépendances Maven à un référentiel CodeArtifact	199
Informations sur le projet Apache Maven	201
À utiliser CodeArtifact avec deps.edn	201
Extraire les dépendances	201
Publier des artefacts	203
Publier avec curl	204
Utiliser les checksums Maven	206

Stockage de Checksum	206
Incompatibilité des checksum lors de la publication	208
Réparation après des incohérences entre les checksum	209
Utiliser les instantanés Maven	209
Publication d'instantanés dans CodeArtifact	210
Consommation de versions instantanées	212
Supprimer des versions de snapshots	213
Publication d'instantanés avec curl	213
Instantanés et connexions externes	216
Instantanés et référentiels en amont	216
Demande de packages Maven depuis des connexions en amont et externes	216
Importation de noms d'actifs standard	217
Importation de noms d'actifs non standard	217
Vérifier l'origine des actifs	218
Importation de nouveaux actifs et de l'état de la version du package dans les référentiels en amont	218
Résolution des problèmes liés à Maven	219
Désactivez les mises en parallèle pour corriger l'erreur 429 : Too Many Requests	219
Utilisation de npm	221
Configurer et utiliser npm	221
Configuration de npm avec la commande login	222
Configuration de npm sans utiliser la commande de connexion	222
Exécution de commandes npm	225
Vérification de l'authentification et de l'autorisation npm	225
Revenir au registre npm par défaut	226
Résolution des problèmes d'installation lents avec npm 8.x ou supérieur	226
Configurer et utiliser Yarn	226
Configurez Yarn 1.X avec la commande <code>aws codeartifact login</code>	227
Configurez Yarn 2.X avec la commande <code>yarn config set</code>	229
support des commandes npm	231
Commandes prises en charge qui interagissent avec un référentiel	231
Commandes côté client prises en charge	233
Commandes non prises en charge	183
gestion des balises npm	237
Modifier les balises avec le client npm	237
les balises npm et l'API <code>CopyPackageVersions</code>	237

balises npm et référentiels en amont	238
Support pour les gestionnaires de paquets compatibles npm	240
En utilisant NuGet	241
Utilisation CodeArtifact avec Visual Studio	241
Configurer Visual Studio avec le fournisseur d' CodeArtifact informations d'identification	242
Utiliser la console Visual Studio Package Manager	243
Utiliser CodeArtifact avec Nuget ou Dotnet	244
Configuration de la CLI nuget ou dotnet	244
Consommez NuGet des colis	250
Publier des NuGet packages	251
CodeArtifact NuGet Référence du fournisseur d'informations d'identification	252
CodeArtifact NuGet Versions du fournisseur d'informations d'identification	253
NuGet normalisation du nom du package, de la version et du nom de l'actif	254
NuGet compatibilité	255
NuGet Compatibilité générale	255
NuGet support en ligne de commande	255
Utilisation de Python	257
Configurer et utiliser pip avec CodeArtifact	257
Configurez pip avec la commande login	257
Configurer pip sans la commande de connexion	258
Exédez pip	259
Configurer et utiliser Twine avec CodeArtifact	260
Configurer la ficelle avec la commande login	260
Configurer la ficelle sans la commande login	261
Run twine	262
Normalisation du nom des paquets Python	262
Compatibilité avec Python	262
support de la commande pip	263
Demande de packages Python depuis des connexions en amont et externes	264
Versions du package supprimées	265
Pourquoi ne CodeArtifact pas récupérer les dernières métadonnées ou ressources supprimées pour une version de package ?	266
Utiliser Ruby	268
Configuration et utilisation d'un RubyGems Bundler	268
Configure RubyGems (gem) et Bundler (bundle) avec CodeArtifact	268
Installation de Ruby Gems	274

Publier des rubis	275
RubyGems support aux commandes	276
Compatibilité avec le bundler	277
Compatibilité avec le bundler	277
Utilisation de Swift	278
Configurez Swift avec CodeArtifact	278
Configurer Swift avec la commande de connexion	278
Configurer Swift sans la commande de connexion	280
Consommation et publication de packages Swift	284
Consommer des packages Swift	284
Consommation de packages Swift dans Xcode	286
Publication de packages Swift	287
Récupération de packages Swift depuis GitHub et republication vers CodeArtifact	289
Normalisation rapide du nom du package et de l'espace de noms	291
Résolution rapide des problèmes	292
Je reçois une erreur 401 dans Xcode même après avoir configuré le Swift Package Manager	292
Xcode se bloque sur la machine CI en raison de l'invite de saisie du mot de passe par le trousseau	292
Utilisation de packages génériques	295
Vue d'ensemble des packages génériques	295
Contraintes de package génériques	295
Commandes prises en charge	296
Publication et consommation de packages génériques	297
Publication d'un package générique	297
Répertorier les actifs des packages génériques	299
Téléchargement des actifs de packages génériques	301
Utilisation CodeArtifact avec CodeBuild	302
Utilisation des packages npm dans CodeBuild	302
Configurer des autorisations avec des rôles IAM	302
Connectez-vous et utilisez npm	303
Utilisation de packages Python dans CodeBuild	304
Configurer des autorisations avec des rôles IAM	305
Connectez-vous et utilisez un pip ou une ficelle	306
Utilisation des packages Maven dans CodeBuild	308
Configurer des autorisations avec des rôles IAM	308

Utilisez Gradle ou MVN	309
Utilisation de NuGet packages dans CodeBuild	310
Configurer des autorisations avec des rôles IAM	310
Consommez NuGet des colis	312
Construire avec des NuGet packages	313
Publier des NuGet packages	315
Mise en cache des dépendances	317
Surveillance CodeArtifact	319
Surveillance des CodeArtifact événements	319
CodeArtifact format d'événement et exemple	321
Utiliser un événement pour démarrer une CodePipeline exécution	325
Configuration EventBridge des autorisations	326
Création de la EventBridge règle	326
Création de la cible de la EventBridge règle	326
Utiliser un événement pour exécuter une fonction Lambda	326
Création de la EventBridge règle	327
Création de la cible de la EventBridge règle	327
Configuration EventBridge des autorisations	327
Sécurité	328
Protection des données	329
Chiffrement des données	330
Confidentialité du trafic	330
Surveillance	331
Journalisation des appels d' CodeArtifact API avec AWS CloudTrail	331
Validation de conformité	335
Authentification et jetons	336
Jetons créés avec la login commande	337
Autorisations requises pour appeler l'GetAuthorizationTokenAPI	339
Tokens créés avec l'GetAuthorizationTokenAPI	339
Transmettre un jeton d'authentification à l'aide d'une variable d'environnement	340
Révocation des jetons CodeArtifact d'autorisation	341
Résilience	342
Sécurité de l'infrastructure	342
Attaques de substitution de la dépendance	343
Gestion de l'identité et des accès	344
Public ciblé	344

Authentification par des identités	345
Gestion de l'accès à l'aide de politiques	346
Comment AWS CodeArtifact fonctionne avec IAM	348
Exemples de stratégies basées sur l'identité	354
Utilisation de balises pour contrôler l'accès aux ressources CodeArtifact	365
AWS CodeArtifact référence aux autorisations	370
Résolution des problèmes	373
Utilisation des points de terminaison d'un VPC	375
Créer des points de terminaison d'un VPC	375
Créer le point de terminaison de la passerelle Amazon S3	377
Autorisations minimales de compartiment Amazon S3 pour AWS CodeArtifact	377
Utilisation CodeArtifact depuis un VPC	380
Utiliser le <code>codeartifact.repositories</code> point de terminaison sans DNS privé	380
Créer une politique de point de terminaison de VPC	381
AWS CloudFormation ressources	383
CodeArtifact et CloudFormation modèles	383
Empêcher la suppression de CodeArtifact ressources	383
En savoir plus sur CloudFormation	384
Résolution des problèmes	385
Je ne peux pas voir les notifications	385
Balisage de ressources	386
CodeArtifact répartition des coûts avec balises	387
Répartition des coûts de stockage des données dans CodeArtifact	387
Répartition des coûts de demande dans CodeArtifact	387
Quotas dans AWS CodeArtifact	388
Historique de la documentation	391
.....	cdiv

Qu'est-ce que c'est AWS CodeArtifact ?

AWS CodeArtifact est un service de référentiel d'artefacts géré, sécurisé et hautement évolutif qui aide les entreprises à stocker et à partager des progiciels pour le développement d'applications. Vous pouvez l'utiliser CodeArtifact avec des outils de construction et des gestionnaires de packages populaires tels que la NuGet CLI, Maven, Gradle, npm, yarn, pip et twine. CodeArtifact vous permet de ne plus avoir à gérer votre propre système de stockage d'artefacts ou à vous soucier de la mise à l'échelle de son infrastructure. Il n'existe aucune limite quant au nombre ou à la taille totale des packages que vous pouvez stocker dans un CodeArtifact référentiel.

Vous pouvez créer une connexion entre votre CodeArtifact dépôt privé et un dépôt public externe, tel que npmjs.com ou Maven Central. CodeArtifact récupérera et stockera ensuite les packages à la demande depuis le référentiel public lorsqu'ils sont demandés par un gestionnaire de packages. Cela facilite l'utilisation des dépendances open source utilisées par votre application et permet de garantir qu'elles sont toujours disponibles pour les builds et le développement. Vous pouvez également publier des packages privés dans un CodeArtifact référentiel. Cela vous permet de partager des composants logiciels propriétaires entre plusieurs applications et équipes de développement de votre organisation.

Pour de plus amples informations, veuillez consulter [AWS CodeArtifact](#).

Comment CodeArtifact fonctionne ?

CodeArtifact stocke les progiciels dans des référentiels. Les référentiels sont polyglottes : un référentiel unique peut contenir tous les types de packages pris en charge. Chaque CodeArtifact dépôt est membre d'un seul CodeArtifact domaine. Nous vous recommandons d'utiliser un seul domaine de production pour votre organisation avec un ou plusieurs référentiels. Par exemple, vous pouvez utiliser chaque référentiel pour une équipe de développement différente. Les packages de vos référentiels peuvent ensuite être découverts et partagés entre vos équipes de développement.

Pour ajouter des packages à un référentiel, configurez un gestionnaire de packages tel que npm ou Maven pour utiliser le point de terminaison (URL) du référentiel. Vous pouvez ensuite utiliser le gestionnaire de packages pour publier des packages dans le référentiel. Vous pouvez également importer des packages open source dans un référentiel en le configurant avec une connexion externe à un référentiel public tel que npmjs, NuGet Gallery, Maven Central ou PyPI. Pour de plus amples informations, veuillez consulter [Connect un CodeArtifact dépôt à un dépôt public](#).

Vous pouvez mettre les packages d'un référentiel à la disposition d'un autre référentiel du même domaine. Pour ce faire, configurez un référentiel en amont de l'autre. Toutes les versions de package disponibles dans le référentiel en amont sont également disponibles dans le référentiel en aval. En outre, tous les packages disponibles dans le référentiel en amont via une connexion externe à un référentiel public sont disponibles dans le référentiel en aval. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

CodeArtifact oblige les utilisateurs à s'authentifier auprès du service afin de publier ou de consommer des versions de package. Vous devez vous authentifier auprès du CodeArtifact service en créant un jeton d'autorisation à l'aide de vos AWS informations d'identification. Les packages des CodeArtifact référentiels ne peuvent pas être mis à la disposition du public. Pour plus d'informations sur l'authentification et l'accès CodeArtifact, consultez [AWS CodeArtifact authentification et jetons](#).

Concepts AWS CodeArtifact

Voici quelques concepts et termes à connaître lorsque vous les utilisez CodeArtifact.

Rubriques

- [Ressource](#)
- [Domaine](#)
- [Référentiel](#)
- [Package](#)
- [Groupe de packages](#)
- [Espace de noms du package](#)
- [Version du package](#)
- [Révision de la version du package](#)
- [Référentiel en amont](#)

Ressource

Un actif est un fichier individuel stocké dans CodeArtifact un fichier associé à une version de package, tel qu'un fichier npm ou des .tgz fichiers Maven POM et JAR.

Domaine

Les référentiels sont agrégés dans une entité de niveau supérieur appelée domaine. Toutes les ressources et métadonnées du package sont stockées dans le domaine, mais elles sont consommées via des référentiels. Un actif de package donné, tel qu'un fichier JAR Maven, est stocké une fois par domaine, quel que soit le nombre de référentiels dans lesquels il se trouve. Toutes les ressources et métadonnées d'un domaine sont chiffrées avec la même clé AWS KMS key (clé KMS) stockée dans AWS Key Management Service (AWS KMS).

Chaque dépôt est membre d'un seul domaine et ne peut pas être déplacé vers un autre domaine.

À l'aide d'un domaine, vous pouvez appliquer une politique organisationnelle à plusieurs référentiels. Cette approche vous permet de déterminer quels comptes peuvent accéder aux référentiels du domaine et quels référentiels publics peuvent être utilisés comme sources de packages.

Bien qu'une organisation puisse avoir plusieurs domaines, nous recommandons un seul domaine de production contenant tous les artefacts publiés. Ainsi, les équipes peuvent trouver et partager des packages au sein de votre organisation.

Référentiel.

Un CodeArtifact référentiel contient un ensemble de [versions de packages](#), chacune correspondant à un ensemble de [ressources](#). Les référentiels sont polyglottes : un référentiel unique peut contenir tous les types de packages pris en charge. Chaque référentiel expose des points de terminaison permettant de récupérer et de publier des packages à l'aide d'outils tels que la CLI nuget, la CLI npm, la CLI Maven (mvn) et pip. Vous pouvez créer jusqu'à 1 000 référentiels par domaine.

Package

Un package est un ensemble de logiciels et de métadonnées nécessaires pour résoudre les dépendances et installer le logiciel. Dans CodeArtifact, un package se compose d'un nom de package, d'un espace de [noms](#) facultatif tel que @types in @types/node, d'un ensemble de versions de package et de métadonnées au niveau du package telles que des balises npm.

AWS CodeArtifact [prend en charge les formats de package Cargo, generic, Maven, npm, NuGetPyPI, Ruby et Swift](#).

Groupe de packages

Les groupes de packages peuvent être utilisés pour appliquer une configuration à plusieurs packages qui correspondent à un modèle défini en utilisant le format du package, l'espace de noms du package et le nom du package. Vous pouvez utiliser des groupes de packages pour configurer plus facilement les contrôles d'origine des packages pour plusieurs packages. Les contrôles d'origine des packages sont utilisés pour bloquer ou autoriser l'ingestion ou la publication de nouvelles versions de packages, ce qui protège les utilisateurs contre les actions malveillantes connues sous le nom d'attaques de substitution de dépendances.

Espace de noms du package

Certains formats de package prennent en charge les noms de packages hiérarchiques afin d'organiser les packages en groupes logiques et d'éviter les collisions de noms. Par exemple, npm prend en charge les scopes. Pour plus d'informations, consultez la documentation de [npm scopes](#). Le package npm @types/node a une portée @types et un nom denode. Il existe de nombreux autres noms de packages dans le @types champ d'application. Dans CodeArtifact, la portée (« types ») est appelée espace de noms du package et le nom (« noeud ») est appelé nom du package. Pour les packages Maven, l'espace de noms du package correspond au Maven GroupID. Le package Maven org.apache.logging.log4j:log4j possède un groupId (espace de noms de package) org.apache.logging.log4j et un artifactID (nom du package). log4j Pour les packages génériques, un [espace de noms](#) est requis. Certains formats de package tels que PyPI ne prennent pas en charge les noms hiérarchiques avec un concept similaire à npm scope ou Maven GroupID. Sans moyen de regrouper les noms de packages, il peut être plus difficile d'éviter les collisions de noms.

Version du package

Une version de package identifie la version spécifique d'un package, telle que @types/node 12.6.9. Le format et la sémantique du numéro de version varient en fonction des différents formats de package. Par exemple, les versions du package npm doivent être conformes à la spécification de [version sémantique](#). Dans CodeArtifact, une version de package comprend l'identifiant de version, les métadonnées au niveau de la version du package et un ensemble de ressources.

Révision de la version du package

Une révision de version de package est une chaîne qui identifie un ensemble spécifique de ressources et de métadonnées pour une version de package. Chaque fois qu'une version de package

est mise à jour, une nouvelle révision de version de package est créée. Par exemple, vous pouvez publier une archive de distribution source (sdist) pour une version de package Python, puis ajouter une roue Python contenant du code compilé à la même version. Lorsque vous publiez la roue, une nouvelle version du package est créée.

Référentiel en amont

Un référentiel est situé en amont d'un autre lorsque les versions des packages qu'il contient sont accessibles depuis le point de terminaison du référentiel du référentiel en aval. Cette approche fusionne efficacement le contenu des deux référentiels du point de vue du client. À l'aide de CodeArtifact, vous pouvez créer une relation en amont entre deux référentiels.

Comment puis-je commencer CodeArtifact ?

Nous vous recommandons d'effectuer les étapes suivantes :

1. Apprenez-en plus CodeArtifact en lisant[Concepts AWS CodeArtifact](#).
2. Configurez votre Compte AWS AWS CLI, le et un utilisateur IAM en suivant les étapes décrites dans[Configuration avec AWS CodeArtifact](#).
3. À utiliser CodeArtifact en suivant les instructions figurant dans[Commencer avec CodeArtifact](#).

Configuration avec AWS CodeArtifact

Si vous êtes déjà inscrit à Amazon Web Services (AWS), vous pouvez commencer à utiliser AWS CodeArtifact dès maintenant. Vous pouvez ouvrir la CodeArtifact console, choisir Créez un domaine et un référentiel, puis suivre les étapes de l'assistant de lancement pour créer votre premier domaine et votre premier référentiel.

Si vous n'êtes pas AWS encore inscrit ou si vous avez besoin d'aide pour créer votre premier domaine et votre premier référentiel, effectuez les tâches suivantes pour configurer votre utilisation CodeArtifact :

Rubriques

- [Inscrivez-vous pour AWS](#)
- [Installez ou mettez à niveau, puis configurez le AWS CLI](#)
- [Approvisionner un utilisateur IAM](#)
- [Installez votre gestionnaire de packages ou votre outil de compilation](#)

Inscrivez-vous pour AWS

Lorsque vous vous inscrivez à Amazon Web Services (AWS), seuls les services et les ressources que vous utilisez vous sont facturés, notamment AWS CodeArtifact.

Si vous en avez déjà un Compte AWS, passez à la tâche suivante, [Installez ou mettez à niveau, puis configurez le AWS CLI](#). Si vous n'en avez pas Compte AWS, utilisez la procédure suivante pour en créer un.

Pour créer un Compte AWS

1. Ouvrez l'[https://portal.aws.amazon.com/billing/inscription.](https://portal.aws.amazon.com/billing/inscription)
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisissez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un

utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les [tâches nécessitant un accès utilisateur racine](#).

Installez ou mettez à niveau, puis configurez le AWS CLI

Pour appeler CodeArtifact des commandes depuis AWS Command Line Interface (AWS CLI) sur une machine de développement locale, vous devez installer le AWS CLI.

Si une ancienne version est AWS CLI installée, vous devez la mettre à niveau pour que les CodeArtifact commandes soient disponibles. CodeArtifact les commandes sont disponibles dans les AWS CLI versions suivantes :

1. AWS CLI 1 : 1.18.77 et versions ultérieures
2. AWS CLI 2 : 2.0.21 et versions ultérieures

Pour vérifier la version, utilisez la `aws --version` commande.

Pour installer et configurer le AWS CLI

1. Installez ou mettez à niveau le AWS CLI en suivant les instructions de [la section Installation du AWS Command Line Interface](#).
2. Configurez le AWS CLI, à l'aide de la commande `configure`, comme suit.

```
aws configure
```

Lorsque vous y êtes invité, spécifiez la clé AWS d'accès et la clé d'accès AWS secrète de l'utilisateur IAM que vous utiliserez avec CodeArtifact. Lorsque vous êtes invité à saisir le Région AWS nom par défaut, spécifiez la région dans laquelle vous allez créer le pipeline, par exemple `us-east-2`. Lorsque vous êtes invité à saisir le format de sortie par défaut, entrez `json`.

Important

Lorsque vous configurez le AWS CLI, vous êtes invité à spécifier un Région AWS. Choisissez l'une des régions prises en charge répertoriées dans [Région et points de terminaison](#) dans le Références générales AWS.

Pour plus d'informations, consultez [Configuration AWS Command Line Interface](#) et [gestion des clés d'accès pour les utilisateurs IAM](#).

3. Pour vérifier l'installation ou la mise à niveau, appelez la commande suivante à partir du AWS CLI.

```
aws codeartifact help
```

En cas de succès, cette commande affiche la liste des CodeArtifact commandes disponibles.

Vous pouvez ensuite créer un utilisateur IAM et lui accorder l'accès à CodeArtifact. Pour de plus amples informations, veuillez consulter [Approvisionner un utilisateur IAM](#).

Approvisionner un utilisateur IAM

Suivez ces instructions pour préparer un utilisateur IAM à l'utiliser CodeArtifact.

Pour approvisionner un utilisateur IAM

1. Créez un utilisateur IAM ou utilisez-en un qui est associé à votre Compte AWS. Pour plus d'informations, consultez les [sections Crédit d'un utilisateur IAM](#) et [Présentation des politiques AWS IAM](#) dans le Guide de l'utilisateur IAM.
2. Accordez à l'utilisateur IAM l'accès à CodeArtifact
 - Option 1 : créer une politique IAM personnalisée. Avec une politique IAM personnalisée, vous pouvez fournir les autorisations minimales requises et modifier la durée de vie des jetons d'authentification. Pour plus d'informations et obtenir des exemples de stratégie, consultez [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#).
 - Option 2 : utilisez la politique `AWSCodeArtifactAdminAccess` AWS gérée. L'extrait suivant montre le contenu de cette politique.

⚠️ Important

Cette politique accorde l'accès à tous CodeArtifact APIs. Nous vous recommandons de toujours utiliser les autorisations minimales requises pour accomplir votre tâche.

Pour plus d'informations, consultez [Bonnes pratiques IAM](#) dans le Guide de l'utilisateur IAM.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:*"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

Note

L'`sts:GetServiceBearerToken` autorisation doit être ajoutée à la politique de rôle ou d'utilisateur IAM. Bien qu'elle puisse être ajoutée à une politique de ressources de CodeArtifact domaine ou de référentiel, l'autorisation n'aura aucun effet sur les politiques de ressources.

L'`sts:GetServiceBearerToken` autorisation est requise pour appeler l'`CodeArtifactGetAuthorizationTokenAPI`. Cette API renvoie un jeton qui doit être utilisé lors

de l'utilisation d'un gestionnaire de packages tel que `npm` ou `pip` avec CodeArtifact. Pour utiliser un gestionnaire de packages avec un CodeArtifact référentiel, votre utilisateur ou rôle IAM doit l'autoriser, `sts:GetServiceBearerToken` comme indiqué dans l'exemple de politique précédent.

Si vous n'avez pas installé le gestionnaire de packages ou l'outil de génération que vous comptez utiliser CodeArtifact, consultez [Installez votre gestionnaire de packages ou votre outil de compilation](#).

Installez votre gestionnaire de packages ou votre outil de compilation

Pour publier ou consommer des packages depuis CodeArtifact, vous devez utiliser un gestionnaire de packages. Il existe différents gestionnaires de packages pour chaque type de package. La liste suivante contient certains gestionnaires de packages que vous pouvez utiliser CodeArtifact. Si ce n'est pas déjà fait, installez les gestionnaires de packages pour le type de package que vous souhaitez utiliser.

- [Pour npm, utilisez la CLI npm ou pnpm.](#)
- [Pour Maven, utilisez Apache Maven \(mvn\) ou Gradle.](#)
- Pour Python, utilisez [pip](#) pour installer des packages et [twine](#) pour publier des packages.
- Pour NuGet, utilisez le [Toolkit for Visual Studio](#) dans Visual Studio ou le [nuget](#) ou le [CLI](#) .
- Pour les packages [génériques](#), utilisez le SDK [AWS CLI](#) ou pour publier et télécharger le contenu du package.

Étapes suivantes

Les prochaines étapes dépendront du ou des types de packages que vous utilisez et de l'état de vos CodeArtifact ressources. CodeArtifact

Si vous lancez CodeArtifact pour la première fois pour vous-même, votre équipe ou votre organisation, consultez la documentation suivante pour obtenir des informations générales de démarrage et de l'aide pour créer les ressources dont vous aurez besoin.

- [Mise en route à l'aide de la console](#)
- [Commencer à utiliser le AWS CLI](#)

Si vos ressources ont déjà été créées et que vous êtes prêt à configurer votre gestionnaire de packages pour envoyer des packages vers ou installer des packages depuis un CodeArtifact référentiel, consultez la documentation correspondant à votre type de package ou à votre gestionnaire de packages.

- [Utilisation CodeArtifact avec npm](#)
- [Utilisation CodeArtifact avec Python](#)
- [Utilisation CodeArtifact avec Maven](#)
- [Utilisation CodeArtifact avec NuGet](#)
- [Utilisation CodeArtifact avec des packages génériques](#)

Commencer avec CodeArtifact

Dans ce didacticiel de mise en route, vous CodeArtifact devez créer les éléments suivants :

- Un domaine appelé `my-domain`.
- Un dépôt appelé `my-repo` qui est contenu dans `my-domain`.
- Un dépôt appelé `npm-store` qui est contenu dans `my-domain`. `npm-store` dispose d'une connexion externe au référentiel public npm. Cette connexion est utilisée pour ingérer un package npm dans le `my-repo` référentiel.

Avant de commencer ce didacticiel, nous vous recommandons de le consulter [CodeArtifact Concepts](#) [AWS CodeArtifact](#).

Note

Ce didacticiel nécessite que vous créez des ressources qui peuvent entraîner des frais sur votre compte AWS . Pour en savoir plus, consultez [Pricing](#) [CodeArtifact](#) (Tarification).

Rubriques

- [Prérequis](#)
- [Mise en route à l'aide de la console](#)
- [Commencer à utiliser le AWS CLI](#)

Prérequis

Vous pouvez terminer ce didacticiel en utilisant le AWS Management Console ou le AWS Command Line Interface (AWS CLI). Pour suivre le didacticiel, vous devez d'abord remplir les prérequis suivants :

- Suivez les étapes de [Configuration avec AWS CodeArtifact](#).
- Installez la CLI npm. Pour plus d'informations, consultez la section [Téléchargement et installation de Node.js et de npm](#) dans la documentation de npm.

Mise en route à l'aide de la console

Exécutez les étapes suivantes pour commencer à utiliser AWS Management Console. Ce guide utilise le gestionnaire de npm packages. Si vous utilisez un autre gestionnaire de packages, vous devrez modifier certaines des étapes suivantes.

1. Connectez-vous à la AWS CodeArtifact console AWS Management Console et ouvrez-la sur <https://console.aws.amazon.com/codesuite/codeartifact/start>. Pour de plus amples informations, veuillez consulter [Configuration avec AWS CodeArtifact](#).
2. Choisissez Créer un référentiel.
3. Dans Nom du référentiel, entrez **my-repo**.
4. (Facultatif) Dans Description du référentiel, entrez une description facultative pour votre référentiel.
5. Dans Référentiels publics en amont, sélectionnez npm-store pour créer un référentiel connecté à npmjs situé en amont de votre référentiel. **my-repo**

CodeArtifact attribue le nom **npm-store** à ce dépôt pour vous. Tous les packages disponibles dans le référentiel en amont **npm-store** sont également disponibles dans son référentiel en aval, **my-repo**.

6. Choisissez Suivant.
7. Dans le compte AWS, sélectionnez Ce compte AWS.
8. Dans Nom de domaine, entrez **my-domain**.
9. Développez Additional configuration (Configuration supplémentaire).
10. Vous devez utiliser une AWS KMS key (clé KMS) pour chiffrer tous les actifs de votre domaine. Vous pouvez utiliser une Clé gérée par AWS ou une clé KMS que vous gérez :

- Choisissez la clé gérée par AWS si vous souhaitez utiliser la clé par défaut Clé gérée par AWS.
- Choisissez Clé gérée par le client si vous souhaitez utiliser une clé KMS que vous gérez. Pour utiliser une clé KMS que vous gérez, dans ARN de la clé gérée par le client, recherchez et choisissez la clé KMS.

Pour plus d'informations, consultez [Clé gérée par AWS](#) la section « [Clé gérée par le client](#) » dans le guide du AWS Key Management Service développeur.

11. Choisissez Suivant.

12. Dans Réviser et créer, passez en revue ce qui CodeArtifact est créé pour vous.

- Le flux de packages montre comment my-domain et npm-store sont liés. my-repo
- Étape 1 : Créer un référentiel affiche des détails sur my-repo et npm-store.
- Étape 2 : Sélectionnez le domaine pour afficher les détails sur my-domain.

Lorsque vous êtes prêt, choisissez Create repository.

13. Sur la page my-repo, choisissez Afficher les instructions de connexion, puis choisissez npm.

14. Utilisez le AWS CLI pour exécuter la login commande indiquée sous Configurer votre client npm à l'aide de cette AWS CLI CodeArtifact commande.

```
aws codeartifact login --tool npm --repository my-repo --domain my-domain --domain-owner 111122223333
```

Vous devriez recevoir un message confirmant la réussite de votre connexion.

```
Successfully configured npm to use AWS CodeArtifact repository https://my-domain-111122223333.d.codeartifact.us-east-2.amazonaws.com/npm/my-repo/  
Login expires in 12 hours at 2020-10-08 02:45:33-04:00
```

Si le message d'erreur s'affiche Could not connect to the endpoint URL, assurez-vous que votre région AWS CLI est configurée et que le nom de votre région par défaut est défini sur la même région que celle dans laquelle vous avez créé votre référentiel. Consultez [Configuration de l'interface de ligne de commande AWS](#).

Pour plus d'informations, consultez [Configurer et utiliser npm avec CodeArtifact](#).

15. Utilisez la CLI npm pour installer un package npm. Par exemple, pour installer le populaire package lodash, utilisez la commande suivante.

```
npm install lodash
```

16. Retournez à la CodeArtifact console. Si votre dépôt my-repo est ouvert, actualisez la page. Sinon, dans le volet de navigation, choisissez Repositories, puis my-repo.

Sous Packages, vous devriez voir la bibliothèque npm, ou le package, que vous avez installé. Vous pouvez choisir le nom du package pour voir sa version et son statut. Vous pouvez choisir sa dernière version pour afficher les détails du package tels que les dépendances, les actifs, etc.

Note

Il peut y avoir un délai entre le moment où vous installez le package et le moment où il est ingéré dans votre dépôt.

17. Pour éviter d'autres AWS frais, supprimez les ressources que vous avez utilisées au cours de ce didacticiel :

Note

Vous ne pouvez pas supprimer un domaine qui contient des référentiels. Vous devez donc le supprimer `npm-store` avant `my-repo` de procéder à la suppression `my-domain`.

- a. Dans le volet de navigation, sélectionnez **Repositories**.
- b. Choisissez `npm-store`, choisissez **Supprimer**, puis suivez les étapes pour supprimer le référentiel.
- c. Choisissez `my-repo`, choisissez **Delete**, puis suivez les étapes pour supprimer le référentiel.
- d. Dans le volet de navigation, sélectionnez **Domains**.
- e. Choisissez `mon-domaine`, choisissez **Supprimer**, puis suivez les étapes pour supprimer le domaine.

Commencer à utiliser le AWS CLI

Exécutez les étapes suivantes pour commencer à utiliser le AWS Command Line Interface (AWS CLI). Pour de plus amples informations, veuillez consulter [Installez ou mettez à niveau, puis configurez le AWS CLI](#). Ce guide utilise le gestionnaire de npm packages. Si vous utilisez un autre gestionnaire de packages, vous devrez modifier certaines des étapes suivantes.

1. Utilisez le AWS CLI pour exécuter la `create-domain` commande.

```
aws codeartifact create-domain --domain my-domain
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre nouveau domaine.

```
{  
  "domain": {  
    "name": "my-domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my-domain",  
    "status": "Active",  
    "createdTime": "2020-10-07T15:36:35.194000-04:00",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0  
  }  
}
```

Si le message d'erreur s'affiche `Could not connect to the endpoint URL`, assurez-vous que votre région AWS CLI est configurée et que le nom de votre région par défaut est défini sur la même région que celle dans laquelle vous avez créé votre référentiel. Consultez [Configuration de l'interface de ligne de commande AWS](#).

2. Utilisez la `create-repository` commande pour créer un référentiel dans votre domaine.

```
aws codeartifact create-repository --domain my-domain --domain-owner 111122223333  
  --repository my-repo
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre nouveau dépôt.

```
{  
  "repository": {  
    "name": "my-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "my-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-  
domain/my-repo",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

}

3. Utilisez la `create-repository` commande pour créer un dépôt en amont pour votre `my-repo` dépôt.

```
aws codeartifact create-repository --domain my-domain --domain-owner 111122223333  
--repository npm-store
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre nouveau dépôt.

```
{  
  "repository": {  
    "name": "npm-store",  
    "administratorAccount": "111122223333",  
    "domainName": "my-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/npm-store",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

4. Utilisez la `associate-external-connection` commande pour ajouter une connexion externe au référentiel public npm à votre `npm-store` référentiel.

```
aws codeartifact associate-external-connection --domain my-domain --domain-owner 111122223333 --repository npm-store --external-connection "public:npmjs"
```

Les données au format JSON apparaissent dans la sortie avec des informations détaillées sur le référentiel et sa nouvelle connexion externe.

```
{  
  "repository": {  
    "name": "npm-store",  
    "administratorAccount": "111122223333",  
    "domainName": "my-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/npm-store",  
    "upstreams": [  
      {  
        "repository": "public:npmjs",  
        "connectionType": "NPM",  
        "connectionStatus": "PENDING",  
        "lastModified": "2023-01-12T12:00:00Z",  
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/public:npmjs"  
      }  
    ],  
    "externalConnections": [  
      {  
        "connectionType": "NPM",  
        "connectionStatus": "PENDING",  
        "lastModified": "2023-01-12T12:00:00Z",  
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/public:npmjs"  
      }  
    ]  
  }  
}
```

```
        "upstreams": [],
        "externalConnections": [
            {
                "externalConnectionName": "public:npmjs",
                "packageFormat": "npm",
                "status": "AVAILABLE"
            }
        ]
    }
}
```

Pour de plus amples informations, veuillez consulter [Connect un CodeArtifact dépôt à un dépôt public](#).

5. Utilisez la update-repository commande pour associer le npm-store référentiel en tant que référentiel en amont au my-repo référentiel.

```
aws codeartifact update-repository --repository my-repo --domain my-domain --
domain-owner 111122223333 --upstreams repositoryName=npm-store
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre dépôt mis à jour, y compris son nouveau référentiel en amont.

```
{
    "repository": {
        "name": "my-repo",
        "administratorAccount": "111122223333",
        "domainName": "my-domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/my-repo",
        "upstreams": [
            {
                "repositoryName": "npm-store"
            }
        ],
        "externalConnections": []
    }
}
```

Pour de plus amples informations, veuillez consulter [Ajouter ou supprimer des référentiels en amont \(\)AWS CLI](#).

6. Utilisez la `login` commande pour configurer votre gestionnaire de packages npm avec votre `my-repo` référentiel.

```
aws codeartifact login --tool npm --repository my-repo --domain my-domain --domain-owner 111122223333
```

Vous devriez recevoir un message confirmant la réussite de votre connexion.

```
Successfully configured npm to use AWS CodeArtifact repository https://my-domain-111122223333.d.codeartifact.us-east-2.amazonaws.com/npm/my-repo/  
Login expires in 12 hours at 2020-10-08 02:45:33-04:00
```

Pour de plus amples informations, veuillez consulter [Configurer et utiliser npm avec CodeArtifact](#).

7. Utilisez la CLI npm pour installer un package npm. Par exemple, pour installer le populaire package `npmlodash`, utilisez la commande suivante.

```
npm install lodash
```

8. Utilisez la `list-packages` commande pour afficher le package que vous venez d'installer dans votre `my-repo` dépôt.

 Note

Il peut y avoir un délai entre la fin de la commande d'npm `install` et le moment où le package est visible dans votre dépôt. Pour plus de détails sur la latence typique lors de la récupération de packages depuis des référentiels publics, consultez [Latence de connexion externe](#)

```
aws codeartifact list-packages --domain my-domain --repository my-repo
```

Les données au format JSON apparaissent dans la sortie avec le format et le nom du package que vous avez installé.

```
{  
  "packages": [  
    {  
      "format": "npm",
```

```
        "package": "lodash"
    }
]
```

Vous disposez désormais de trois CodeArtifact ressources :

- Le domaine `my-domain`.
 - Le référentiel `my-repo` contenu dans `my-domain`. Ce référentiel dispose d'un package npm.
 - Le référentiel `npm-store` contenu dans `my-domain`. Ce référentiel dispose d'une connexion externe au référentiel public npm et est associé en tant que référentiel en amont au `my-repo` référentiel.
9. Pour éviter d'autres AWS frais, supprimez les ressources que vous avez utilisées au cours de ce didacticiel :

 Note

Vous ne pouvez pas supprimer un domaine qui contient des référentiels. Vous devez donc le supprimer `npm-store` avant `my-repo` de procéder à la suppression `my-domain`.

- a. Utilisez la `delete-repository` commande pour supprimer le `npm-store` référentiel.

```
aws codeartifact delete-repository --domain my-domain --domain-owner 111122223333 --repository my-repo
```

Les données au format JSON apparaissent dans la sortie avec des informations sur le référentiel supprimé.

```
{  
    "repository": {  
        "name": "my-repo",  
        "administratorAccount": "111122223333",  
        "domainName": "my-domain",  
        "domainOwner": "111122223333",  
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/my-repo",  
        "upstreams": [  
    }  
}
```

```
        {
            "repositoryName": "npm-store"
        }
    ],
    "externalConnections": []
}
```

- b. Utilisez la `delete-repository` commande pour supprimer le `npm-store` référentiel.

```
aws codeartifact delete-repository --domain my-domain --domain-owner 111122223333 --repository npm-store
```

Les données au format JSON apparaissent dans la sortie avec des informations sur le référentiel supprimé.

```
{
    "repository": {
        "name": "npm-store",
        "administratorAccount": "111122223333",
        "domainName": "my-domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/npm-store",
        "upstreams": [],
        "externalConnections": [
            {
                "externalConnectionName": "public:npmjs",
                "packageFormat": "npm",
                "status": "AVAILABLE"
            }
        ]
    }
}
```

- c. Utilisez la `delete-domain` commande pour supprimer le `my-domain` référentiel.

```
aws codeartifact delete-domain --domain my-domain --domain-owner 111122223333
```

Les données au format JSON apparaissent dans la sortie avec des informations sur le domaine supprimé.

```
{  
  "domain": {  
    "name": "my-domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my-domain",  
    "status": "Deleted",  
    "createdTime": "2020-10-07T15:36:35.194000-04:00",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0  
  }  
}
```

Utilisation de référentiels dans CodeArtifact

Ces rubriques expliquent comment utiliser la CodeArtifact console et comment créer AWS CLI, CodeArtifact APIs répertorier, mettre à jour et supprimer des référentiels.

Rubriques

- [Création d'un référentiel](#)
- [Connexion à un référentiel](#)
- [Supprimer un dépôt](#)
- [Répertorier les référentiels](#)
- [Afficher ou modifier la configuration d'un référentiel](#)
- [Politiques de référentiel](#)
- [Marquer un dépôt dans CodeArtifact](#)

Création d'un référentiel

Comme tous les packages CodeArtifact sont stockés dans [des référentiels](#) CodeArtifact, vous devez en créer un pour pouvoir les utiliser. Vous pouvez créer un référentiel à l'aide de la CodeArtifact console, du AWS Command Line Interface (AWS CLI) ou CloudFormation. Chaque dépôt est associé au AWS compte que vous utilisez lorsque vous le créez. Vous pouvez avoir plusieurs référentiels, qui sont créés et regroupés en [domaines](#). Lorsque vous créez un dépôt, celui-ci ne contient aucun package. Les référentiels sont polyglottes, ce qui signifie qu'un dépôt unique peut contenir tous les types de packages pris en charge.

Pour plus d'informations sur les limites de CodeArtifact service, telles que le nombre maximum de référentiels autorisés dans un seul domaine, consultez[Quotas dans AWS CodeArtifact](#). Si vous atteignez le nombre maximum de référentiels autorisés, vous pouvez [supprimer des référentiels](#) pour faire de la place à d'autres.

Un référentiel peut être associé à un ou plusieurs CodeArtifact référentiels en tant que référentiels en amont. Cela permet à un client du gestionnaire de packages d'accéder aux packages contenus dans plusieurs référentiels à l'aide d'un seul point de terminaison URL. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

Pour plus d'informations sur la gestion des CodeArtifact référentiels avec CloudFormation, consultez[Création de CodeArtifact ressources avec AWS CloudFormation](#).

Note

Après avoir créé un référentiel, vous ne pouvez pas modifier son nom, le AWS compte associé ou le domaine.

Rubriques

- [Création d'un référentiel \(console\)](#)
- [Création d'un référentiel \(AWS CLI\)](#)
- [Création d'un référentiel avec un référentiel en amont](#)

Création d'un référentiel (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Repositories, puis Create repository.
3. Dans Nom du référentiel, entrez le nom de votre référentiel.
4. (Facultatif) Dans Description du référentiel, entrez une description facultative pour votre référentiel.
5. (Facultatif) Dans Publier les référentiels en amont, ajoutez des référentiels intermédiaires qui connectent vos référentiels aux autorités de package telles que Maven Central ou npmjs.com.
6. Choisissez Suivant.
7. Dans le compte AWS, choisissez Ce compte AWS si vous êtes connecté au compte propriétaire du domaine. Choisissez un autre compte AWS si un autre compte AWS possède le domaine.
8. Dans Domaine, choisissez le domaine dans lequel le référentiel sera créé.

S'il n'y a aucun domaine dans le compte, vous devez en créer un. Entrez le nom du nouveau domaine dans Nom de domaine.

Développez Additional configuration (Configuration supplémentaire).

Vous devez utiliser une AWS KMS key (clé KMS) pour chiffrer tous les actifs de votre domaine. Vous pouvez utiliser une Clé gérée par AWS ou une clé KMS que vous gérez :

⚠️ Important

CodeArtifact ne prend en charge que les [clés KMS symétriques](#). Vous ne pouvez pas utiliser de [clé KMS asymétrique](#) pour chiffrer vos CodeArtifact domaines. Pour savoir si une clés KMS est symétrique ou asymétrique, consultez [Identification de clés KMS symétriques et asymétriques](#).

- Choisissez la clé gérée par AWS si vous souhaitez utiliser la clé par défaut Clé gérée par AWS.
- Choisissez Clé gérée par le client si vous souhaitez utiliser une clé KMS que vous gérez. Pour utiliser une clé KMS que vous gérez, dans ARN de la clé gérée par le client, recherchez et choisissez la clé KMS.

Pour plus d'informations, consultez [Clés gérées par AWS](#) la section « [clé gérée par le client](#) » dans le guide du AWS Key Management Service développeur.

9. Choisissez Suivant.
10. Dans Réviser et créer, passez en revue ce qui CodeArtifact est créé pour vous.
 - Package Flow indique comment votre domaine et vos référentiels sont connectés.
 - L'étape 1 : Créer un référentiel affiche des détails sur le référentiel et les référentiels en amont facultatifs qui seront créés.
 - Étape 2 : Sélectionnez le domaine pour afficher les détails sur `my_domain`.

Lorsque vous êtes prêt, choisissez Create repository.

Création d'un référentiel (AWS CLI)

Utilisez la `create-repository` commande pour créer un référentiel dans votre domaine.

```
aws codeartifact create-repository --domain my_domain --domain-owner 111122223333 --repository my_repo --description "My new repository"
```

Exemple de sortie :

```
{  
  "repository": {  
    "name": "my_repo",  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:region-  
id:111122223333:repository/my_domain/my_repo",  
    "description": "My new repository",  
    "upstreams": "[]",  
    "externalConnections": "[]"  
  }  
}
```

Un nouveau dépôt ne contient aucun package. Chaque dépôt est associé au AWS compte auprès duquel vous êtes authentifié lors de sa création.

Création d'un référentiel avec des balises

Pour créer un référentiel avec des balises, ajoutez le `--tags` paramètre à votre `create-domain` commande.

```
aws codeartifact create-repository --domain my_domain --domain-owner 111122223333 --  
repository my_repo --tags key=k1,value=v1 key=k2,value=v2
```

Création d'un référentiel avec un référentiel en amont

Vous pouvez spécifier un ou plusieurs référentiels en amont lorsque vous créez un référentiel.

```
aws codeartifact create-repository --domain my_domain --domain-owner 111122223333 --  
repository my_repo \  
--upstreams repositoryName=my-upstream-repo --repository-description "My new  
repository"
```

Exemple de sortie :

```
{  
  "repository": {  
    "name": "my_repo",  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:region-  
id:111122223333:repository/my_domain/my_repo",  
    "description": "My new repository",  
    "upstreams": "[]",  
    "externalConnections": "[]"  
  }  
}
```

```
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/my_repo",
        "description": "My new repository",
        "upstreams": [
            {
                "repositoryName": "my-upstream-repo"
            }
        ],
        "externalConnections": []
    }
}
```

Note

Pour créer un référentiel avec un référentiel en amont, vous devez être autorisé à effectuer l'AssociateWithDownstreamRepositoryaction sur le référentiel en amont.

Pour ajouter un amont à un référentiel après sa création, reportez-vous aux sections [Ajouter ou supprimer des référentiels en amont \(console\)](#) et [Ajouter ou supprimer des référentiels en amont \(AWS CLI\)](#).

Connexion à un référentiel

Après avoir configuré votre profil et vos informations d'identification pour vous authentifier auprès de votre AWS compte, choisissez le référentiel dans CodeArtifact lequel vous souhaitez les utiliser. Vous avez les options suivantes :

- Créez un référentiel. Pour plus d'informations, consultez la section [Création d'un référentiel](#).
- Utilisez un référentiel qui existe déjà dans votre compte. Vous pouvez utiliser la `list-repositories` commande pour rechercher les référentiels créés dans votre AWS compte. Pour de plus amples informations, veuillez consulter [Répertorier les référentiels](#).
- Utilisez un dépôt dans un autre AWS compte. Pour plus d'informations, consultez la section [Politiques du référentiel](#).

Utiliser un client de gestion de packages

Une fois que vous savez quel dépôt vous souhaitez utiliser, consultez l'une des rubriques suivantes.

- [Utilisation CodeArtifact avec Maven](#)
- [Utilisation CodeArtifact avec npm](#)
- [Utilisation CodeArtifact avec NuGet](#)
- [Utilisation CodeArtifact avec Python](#)

Supprimer un dépôt

Vous pouvez supprimer un dépôt à l'aide de la CodeArtifact console ou du AWS CLI. Une fois qu'un dépôt a été supprimé, vous ne pouvez plus y envoyer de packages ni en extraire des packages. Tous les packages du référentiel deviennent définitivement indisponibles et ne peuvent pas être restaurés. Vous pouvez créer un dépôt portant le même nom, mais son contenu sera vide.

Important

La suppression d'un dépôt ne peut pas être annulée. Une fois que vous avez supprimé un dépôt, vous n'êtes plus en mesure de le récupérer et il ne peut pas être restauré.

Rubriques

- [Supprimer un dépôt \(console\)](#)
- [Supprimer un dépôt \(AWS CLI\)](#)
- [Empêcher la suppression des référentiels](#)

Supprimer un dépôt (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Repositories, puis choisissez le référentiel que vous souhaitez supprimer.
3. Choisissez Supprimer, puis suivez les étapes pour supprimer le domaine.

Supprimer un dépôt (AWS CLI)

Utilisez la `delete-repository` commande pour supprimer un dépôt.

```
aws codeartifact delete-repository --domain my_domain --domain-owner 111122223333 --repository my_repo
```

Exemple de sortie :

```
{  
  "repository": {  
    "name": "my_repo",  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "123456789012",  
    "arn": "arn:aws:codeartifact:region-id:123456789012:repository/my_domain/my_repo",  
    "description": "My new repository",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

Empêcher la suppression des référentiels

Vous pouvez empêcher la suppression accidentelle d'un dépôt en incluant une politique de domaine similaire à la suivante :

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyRepositoryDeletion",  
      "Action": [  
        "codeartifact:DeleteRepository"  
      ],  
      "Effect": "Deny",  
      "Resource": "*",  
      "Principal": "*"
```

```
    }
]
}
```

Cette politique empêche tous les principaux de supprimer le référentiel, mais si vous décidez ultérieurement que vous devez le supprimer, vous pouvez le faire en suivant ces étapes :

1. Dans la politique de domaine, mettez-la à jour comme suit :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRepositoryDeletion",
      "Action": [
        "codeartifact:DeleteRepository"
      ],
      "Effect": "Deny",
      "NotResource": "arn:aws:iam::*:role/Service*",
      "Principal": "*"
    }
  ]
}
```

repository-arn Remplacez-le par l'ARN du référentiel que vous souhaitez supprimer.

2. Dans la AWS CodeArtifact console, choisissez Repositories et supprimez le référentiel de votre choix.
3. Après avoir supprimé le référentiel, vous pouvez modifier à nouveau la politique afin d'éviter les sélections accidentnelles.

JSON

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

```

```
        "Sid": "DenyRepositoryDeletion",
        "Action": [
            "codeartifact:DeleteRepository"
        ],
        "Effect": "Deny",
        "Resource": "*",
        "Principal": "*"
    }
]
```

Vous pouvez également inclure la même déclaration de refus dans une politique de référentiel. Cela vous permet de bénéficier d'une plus grande flexibilité pour protéger les référentiels de grande valeur contre la suppression.

Répertorier les référentiels

Utilisez les commandes de cette rubrique pour répertorier les référentiels d'un AWS compte ou d'un domaine.

Répertorier les référentiels d'un compte AWS

Utilisez cette commande pour répertorier tous les référentiels de votre AWS compte.

```
aws codeartifact list-repositories
```

Exemple de sortie :

```
{
    "repositories": [
        {
            "name": "repo1",
            "administratorAccount": "123456789012",
            "domainName": "my_domain",
            "domainOwner": "123456789012",
            "arn": "arn:aws:codeartifact:region-
id:123456789012:repository/my_domain/repo1",
            "description": "Description of repo1"
        },
        {
            "name": "repo2",
            "administratorAccount": "123456789012",
            "domainName": "my_domain",
            "domainOwner": "123456789012",
            "arn": "arn:aws:codeartifact:region-
id:123456789012:repository/my_domain/repo2",
            "description": "Description of repo2"
        }
    ]
}
```

```
        "name": "repo2",
        "administratorAccount": "123456789012",
        "domainName": "my_domain",
        "domainOwner": "123456789012",
        "arn": "arn:aws:codeartifact:region-
id:123456789012:repository/my_domain/repo2",
        "description": "Description of repo2"

    },
    {
        "name": "repo3",
        "administratorAccount": "123456789012",
        "domainName": "my_domain2",
        "domainOwner": "123456789012",
        "arn": "arn:aws:codeartifact:region-
id:123456789012:repository/my_domain2/repo3",
        "description": "Description of repo3"
    }
]
```

Vous pouvez paginer la réponse à l'list-repositories aide des --next-token paramètres --max-results et. Pour --max-results, spécifiez un entier compris entre 1 et 1 000 pour spécifier le nombre de résultats renvoyés sur une seule page. Sa valeur par défaut est 50. Pour renvoyer les pages suivantes, exéutez list-repositories à nouveau et transmettez la nextToken valeur reçue dans la sortie de commande précédente à --next-token. Lorsque l'--next-token option n'est pas utilisée, la première page de résultats est toujours renvoyée.

Répertorier les référentiels du domaine

Permet list-repositories-in-domain d'obtenir la liste de tous les référentiels d'un domaine.

```
aws codeartifact list-repositories-in-domain --domain my_domain --domain-
owner 123456789012 --max-results 3
```

Le résultat indique que certains référentiels sont administrés par différents AWS comptes.

```
{
    "repositories": [
        {
            "name": "repo1",
            "administratorAccount": "123456789012",
```

```
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/repo1",
        "description": "Description of repo1"
    },
    {
        "name": "repo2",
        "administratorAccount": "444455556666",
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/repo2",
        "description": "Description of repo2"
    },
    {
        "name": "repo3",
        "administratorAccount": "444455556666",
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/repo3",
        "description": "Description of repo3"
    }
]
```

Vous pouvez paginer la réponse à l'list-repositories-in-domainaide des --next-token paramètres --max-results et. Pour--max-results, spécifiez un entier compris entre 1 et 1 000 pour spécifier le nombre de résultats renvoyés sur une seule page. Sa valeur par défaut est 50. Pour renvoyer les pages suivantes, exécutez list-repositories-in-domain à nouveau et transmettez la nextToken valeur reçue dans la sortie de commande précédente à--next-token. Lorsque l'--next-tokenoption n'est pas utilisée, la première page de résultats est toujours renvoyée.

Pour afficher les noms des référentiels dans une liste plus compacte, essayez la commande suivante.

```
aws codeartifact list-repositories-in-domain --domain my_domain --domain-
owner 111122223333 \
--query 'repositories[*].[name]' --output text
```

Exemple de sortie :

```
repo1
repo2
repo3
```

L'exemple suivant affiche l'ID du compte en plus du nom du référentiel.

```
aws codeartifact list-repositories-in-domain --domain my_domain --domain-owner 111122223333 \
--query 'repositories[*].[name,administratorAccount]' --output text
```

Exemple de sortie :

```
repo1 710221105108
repo2 710221105108
repo3 532996949307
```

Pour plus d'informations sur le `--query` paramètre, consultez [ListRepositories](#) la référence de l'CodeArtifact API.

Afficher ou modifier la configuration d'un référentiel

Vous pouvez afficher et mettre à jour les informations relatives à votre dépôt à l'aide de la CodeArtifact console ou du AWS Command Line Interface (AWS CLI).

Note

Après avoir créé un référentiel, vous ne pouvez pas modifier son nom, le AWS compte associé ou le domaine.

Rubriques

- [Afficher ou modifier la configuration d'un référentiel \(console\)](#)
- [Afficher ou modifier la configuration d'un référentiel \(AWS CLI\)](#)

Afficher ou modifier la configuration d'un référentiel (console)

Vous pouvez consulter les détails de votre référentiel et le mettre à jour à l'aide de la CodeArtifact console.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Repositories, puis choisissez le nom du référentiel que vous souhaitez afficher ou modifier.
3. Développez les détails pour voir ce qui suit :
 - Le domaine du référentiel. Choisissez le nom de domaine pour en savoir plus.
 - Politique en matière de ressources du référentiel. Choisissez Appliquer une politique de référentiel pour en ajouter une.
 - Le nom de ressource Amazon (ARN) du référentiel.
 - Si votre dépôt dispose d'une connexion externe, vous pouvez choisir la connexion pour en savoir plus à ce sujet. Un référentiel ne peut avoir qu'une seule connexion externe. Pour de plus amples informations, veuillez consulter [Connect un CodeArtifact dépôt à un dépôt public](#).
 - Si votre dépôt comporte des référentiels en amont, vous pouvez en choisir un pour en voir les détails. Un référentiel peut comporter jusqu'à 10 référentiels en amont direct. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

 Note

Un référentiel peut avoir une connexion externe ou des référentiels en amont, mais pas les deux.

4. Dans Packages, vous pouvez voir tous les packages disponibles dans ce référentiel. Choisissez un forfait pour en savoir plus à son sujet.
5. Choisissez Afficher les instructions de connexion, puis choisissez un gestionnaire de packages pour savoir comment le configurer CodeArtifact.
6. Choisissez Appliquer la politique de référentiel pour mettre à jour ou ajouter une politique de ressources à votre référentiel. Pour de plus amples informations, veuillez consulter [Politiques de référentiel](#).
7. Choisissez Modifier pour ajouter ou mettre à jour les éléments suivants.
 - Description du référentiel.
 - Balises associées au référentiel.
 - Si votre dépôt dispose d'une connexion externe, vous pouvez modifier le référentiel public auquel il se connecte. Sinon, vous pouvez ajouter un ou plusieurs référentiels existants en tant

que référentiels en amont. Disposez-les dans l'ordre dans lequel vous souhaitez qu'ils soient classés par ordre de priorité CodeArtifact lorsqu'un colis est demandé. Pour de plus amples informations, veuillez consulter [Ordre de priorité du référentiel en amont](#).

Afficher ou modifier la configuration d'un référentiel (AWS CLI)

Pour afficher la configuration actuelle d'un dépôt dans CodeArtifact, utilisez la `describe-repository` commande.

```
aws codeartifact describe-repository --domain my_domain --domain-owner 111122223333 --repository my_repo
```

Exemple de sortie :

```
{  
  "repository": {  
    "name": "my_repo",  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:region-id:111122223333:repository/my_domain/my_repo"  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

Modifier la configuration en amont d'un référentiel

Un référentiel en amont permet à un client du gestionnaire de packages d'accéder aux packages contenus dans plusieurs référentiels à l'aide d'un seul point de terminaison URL. Pour ajouter ou modifier la relation en amont d'un dépôt, utilisez la `update-repository` commande.

```
aws codeartifact update-repository --domain my_domain --domain-owner 111122223333 --repository my_repo \  
  --upstreams repositoryName=my-upstream-repo
```

Exemple de sortie :

```
{
```

```
"repository": {
    "name": "my_repo",
    "administratorAccount": "123456789012",
    "domainName": "my_domain",
    "domainOwner": "111122223333",
    "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/my_repo"
    "upstreams": [
        {
            "repositoryName": "my-upstream-repo"
        }
    ],
    "externalConnections": []
}
}
```

Note

Pour ajouter un référentiel en amont, vous devez être autorisé à effectuer l'AssociateWithDownstreamRepositoryaction sur le référentiel en amont.

Pour supprimer la relation en amont d'un dépôt, utilisez une liste vide comme argument de l'--upstreamsoption.

```
aws codeartifact update-repository --domain my_domain --domain-owner 111122223333 --
repository my_repo --upstreams []
```

Exemple de sortie :

```
{
    "repository": {
        "name": "my_repo",
        "administratorAccount": "123456789012",
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/my_repo"
        "upstreams": [],
        "externalConnections": []
    }
}
```

{

Politiques de référentiel

CodeArtifact utilise des autorisations basées sur les ressources pour contrôler l'accès. Les autorisations basées sur les ressources vous permettent de spécifier qui a accès à un référentiel et quelles actions peuvent être exécutées. Par défaut, seul le propriétaire du référentiel a accès à celui-ci. Vous pouvez appliquer un document de politique permettant aux autres principaux IAM d'accéder à votre référentiel.

Pour plus d'informations, consultez les sections Politiques basées sur les [ressources](#), [Stratégies basées sur l'identité](#) et [Stratégies basées sur les ressources](#).

Créez une politique de ressources pour accorder l'accès en lecture

Une politique de ressources est un fichier texte au format JSON. Le fichier doit spécifier un principal (acteur), une ou plusieurs actions et un effet (AllowouDeny). Par exemple, la politique de ressources suivante accorde au compte l'123456789012 autorisation de télécharger des packages depuis le référentiel.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:ReadFromRepository"  
      ],  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      },  
      "Resource": "*"  
    }  
  ]  
}
```

Comme la politique est évaluée uniquement pour les opérations effectuées par rapport au référentiel auquel elle est attachée, il n'est pas nécessaire de spécifier de ressource. Comme la ressource est implicite, vous pouvez définir la valeur `Resource` sur*. Pour qu'un gestionnaire de packages puisse télécharger un package depuis ce référentiel, une politique de domaine pour l'accès entre comptes doit également être créée. La politique de domaine doit au moins accorder une `codeartifact:GetAuthorizationToken` autorisation au principal. Pour un exemple de politique de domaine complète permettant d'accorder un accès entre comptes, consultez ceci [Exemple de politique de domaine](#).

 Note

L'`codeartifact:ReadFromRepository` action ne peut être utilisée que sur une ressource de référentiel. Vous ne pouvez pas mettre le nom de ressource Amazon (ARN) d'un package en tant `codeartifact:ReadFromRepository` que ressource dans le but d'autoriser l'accès en lecture à un sous-ensemble de packages d'un référentiel. Un principal donné peut lire tous les packages d'un dépôt ou aucun d'entre eux.

Étant donné que la seule action spécifiée dans le référentiel est que ReadFromRepository les utilisateurs et les rôles du compte 1234567890 peuvent télécharger des packages depuis le référentiel. Cependant, ils ne peuvent pas effectuer d'autres actions sur eux (par exemple, répertorier les noms et les versions des packages). Généralement, vous accordez des autorisations dans le cadre de la politique suivante, en plus du ReadFromRepository fait qu'un utilisateur qui télécharge des packages depuis un référentiel doit également interagir avec celui-ci d'une autre manière.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:DescribePackageVersion",  
        "codeartifact:DescribeRepository",  
        "codeartifact:GetPackageVersionReadme",  
        "codeartifact:GetRepositoryEndpoint",  
        "codeartifact>ListPackages",  
        "codeartifact>ListPackageVersions",  
        "codeartifact>ListPackageVersionAssets",  
        "codeartifact:PutPackageVersionReadme",  
        "codeartifact:PutRepositoryEndpoint",  
        "codeartifact:UpdatePackageVersionReadme",  
        "codeartifact:UpdateRepositoryEndpoint"  
      ]  
    }  
  ]  
}
```

```
        "codeartifact>ListPackageVersionDependencies",
        "codeartifactReadFromRepository"
    ],
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
    },
    "Resource": "*"
}
]
```

Définissez une politique

Après avoir créé un document de politique, utilisez la `put-repository-permissions-policy` commande pour le joindre à un référentiel :

```
aws codeartifact put-repository-permissions-policy --domain my_domain --domain-owner 111122223333 \
    --repository my_repo --policy-document file:///PATH/TO/policy.json
```

Lorsque vous appelez `put-repository-permissions-policy`, la politique de ressources du référentiel est ignorée lors de l'évaluation des autorisations. Cela garantit que le propriétaire d'un domaine ne peut pas s'exclure du référentiel, ce qui l'empêcherait de mettre à jour la politique de ressources.

Note

Vous ne pouvez pas autoriser un autre AWS compte à mettre à jour la politique de ressources d'un référentiel à l'aide d'une stratégie de ressources, car la politique de ressources est ignorée lors de l'appel `put-repository-permissions-policy`.

Exemple de sortie :

```
{
    "policy": {
        "resourceArn": "arn:aws:codeartifact:region-id:111122223333:repository/my_domain/my_repo",
        "document": "{ ...policy document content... }",
    }
}
```

```
        "revision": "MQ1yyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxxxx"  
    }  
}
```

La sortie de la commande contient l'Amazon Resource Name (ARN) de la ressource du référentiel, le contenu complet du document de politique et un identifiant de révision. Vous pouvez transmettre l'identifiant de révision à `put-repository-permissions-policy` l'aide de l'`--policy-revision` option. Cela garantit qu'une révision connue du document est remplacée, et non une version plus récente définie par un autre rédacteur.

Lire une politique

Utilisez la `get-repository-permissions-policy` commande pour lire une version existante d'un document de politique. Pour formater la sortie afin de la rendre lisible, utilisez le `--output` et `--query` `policy.document` avec le `json.tool` module Python.

```
aws codeartifact get-repository-permissions-policy --domain my_domain --domain-owner 111122223333 \  
    --repository my_repo --output text --query policy.document | python -m  
    json.tool
```

Exemple de sortie :

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::123456789012:root"  
            },  
            "Action": [  
                "codeartifact:DescribePackageVersion",  
                "codeartifact:DescribeRepository",  
                "codeartifact:GetPackageVersionReadme",  
                "codeartifact:GetRepositoryEndpoint",  
                "codeartifact>ListPackages",  
                "codeartifact>ListPackageVersions",  
                "codeartifact>ListPackageVersionAssets",  
            ]  
        }  
    ]  
}
```

```
        "codeartifact>ListPackageVersionDependencies",
        "codeartifact	ReadFromRepository"
    ],
    "Resource": "*"
}
]
```

Supprimer une politique

Utilisez la `delete-repository-permissions-policy` commande pour supprimer une politique d'un référentiel.

```
aws codeartifact delete-repository-permissions-policy --domain my_domain --domain-owner 111122223333 \
--repository my_repo
```

Le format de sortie est le même que celui de la `get-repository-permissions-policy` commande.

Accorder un accès en lecture aux principaux

Lorsque vous spécifiez l'utilisateur root d'un compte comme principal dans un document de politique, vous accordez l'accès à tous les utilisateurs et rôles de ce compte. Pour limiter l'accès à certains utilisateurs ou rôles, utilisez leur ARN dans la `Principal` section de la politique. Par exemple, utilisez ce qui suit pour accorder un accès en lecture à l'utilisateur IAM bob dans le compte 123456789012.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:ReadFromRepository"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/bob"
            }
        }
    ]
}
```

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

Accorder l'accès en écriture aux packages

L'`codeartifact:PublishPackageVersion` action est utilisée pour contrôler l'autorisation de publier les nouvelles versions d'un package. La ressource utilisée avec cette action doit être un package. Le format du CodeArtifact package ARNs est le suivant.

```
arn:aws:codeartifact:region-id:111122223333:package/my_domain/my_repo/package-format/package-namespace/package-name
```

L'exemple suivant montre l'ARN d'un package npm avec la portée `@parity` et le nom `ui` dans le `my_repo` référentiel du domaine `my_domain`.

```
arn:aws:codeartifact:region-id:111122223333:package/my_domain/my_repo/npm/parity/ui
```

L'ARN d'un package npm sans portée contient la chaîne vide pour le champ d'espace de noms. Par exemple, voici l'ARN d'un package sans portée et dont le nom se trouve `react` dans le `my_repo` référentiel du domaine `my_domain`.

```
arn:aws:codeartifact:region-id:111122223333:package/my_domain/my_repo/npm//react
```

La politique suivante autorise le compte `123456789012` à publier des versions de `@parity/ui` dans le `my_repo` référentiel.

JSON

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

```
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
        },
        "Resource": "arn:aws:codeartifact:us-
east-1:111122223333:package/my_domain/my_repo/npm/parity/ui"
    }
}
```

Important

Pour accorder l'autorisation de publier des versions de Maven et de NuGet package, ajoutez les autorisations suivantes en plus `decodeartifact:PublishPackageVersion`.

1. NuGet: `codeartifact:ReadFromRepository` et spécifiez la ressource du référentiel
2. Maven : `codeartifact:PutPackageMetadata`

Étant donné que cette politique spécifie un domaine et un référentiel dans le cadre de la ressource, elle autorise la publication uniquement lorsqu'ils sont attachés à ce référentiel.

Accorder l'accès en écriture à un dépôt

Vous pouvez utiliser des caractères génériques pour accorder une autorisation d'écriture à tous les packages d'un référentiel. Par exemple, utilisez la politique suivante pour accorder à un compte l'autorisation d'écrire dans tous les packages du `my_repo` référentiel.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:PublishPackageVersion"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            }
        }
    ]
}
```

```
  },
  "Resource": "arn:aws:codeartifact:us-
east-1:111122223333:package/my_domain/my_repo/*"
}
]
```

Interaction entre le référentiel et les politiques de domaine

CodeArtifact prend en charge les politiques de ressources sur les domaines et les référentiels. Les politiques relatives aux ressources sont facultatives. Chaque domaine peut avoir une politique et chaque référentiel du domaine peut avoir sa propre politique de référentiel. Si une politique de domaine et une politique de référentiel sont présentes, les deux sont évaluées pour déterminer si une demande adressée à un CodeArtifact référentiel est autorisée ou refusée. Les politiques de domaine et de référentiel sont évaluées à l'aide des règles suivantes :

- Aucune politique de ressources n'est évaluée lors de l'exécution d'opérations au niveau du compte telles [ListDomains](#)que ou. [ListRepositories](#)
- Aucune politique de référentiel n'est évaluée lors de l'exécution d'opérations au niveau du domaine telles [DescribeDomain](#)que ou. [ListRepositoriesInDomain](#)
- La politique de domaine n'est pas évaluée lors de l'exécution [PutDomainPermissionsPolicy](#). Notez que cette règle empêche les lock-outs.
- La politique de domaine est évaluée lors de l'exécution [PutRepositoryPermissionsPolicy](#), mais la politique de référentiel n'est pas évaluée.
- Un refus explicite dans une politique remplace une autorisation dans une autre politique.
- Une autorisation explicite n'est requise que dans une seule politique de ressources. L'omission d'une action dans une politique de référentiel n'entraîne pas de refus implicite si la politique de domaine autorise l'action.
- Lorsqu'aucune politique de ressources n'autorise une action, le résultat est un refus implicite, sauf si le compte du principal appelant est le propriétaire du domaine ou le compte de l'administrateur du référentiel et qu'une politique basée sur l'identité autorise l'action.

Les politiques de ressources sont facultatives lorsqu'elles sont utilisées pour accorder l'accès dans un scénario de compte unique, où le compte de l'appelant utilisé pour accéder à un référentiel est le même que le compte du propriétaire du domaine et le compte de l'administrateur du référentiel. Des politiques en matière de ressources sont nécessaires pour accorder l'accès dans un scénario multi-

comptes où le compte de l'appelant n'est pas le même que le compte du propriétaire du domaine ou de l'administrateur du référentiel. L'accès entre comptes CodeArtifact suit les règles générales IAM relatives à l'accès entre comptes, décrites dans la section [Déterminer si une demande entre comptes est autorisée](#) dans le guide de l'utilisateur IAM.

- Un titulaire du compte du propriétaire du domaine peut avoir accès à n'importe quel référentiel du domaine par le biais d'une politique basée sur l'identité. Notez que dans ce cas, aucune autorisation explicite n'est requise dans une politique de domaine ou de référentiel.
- Un utilisateur principal du compte du propriétaire du domaine peut se voir accorder l'accès à n'importe quel référentiel par le biais d'une politique de domaine ou de référentiel. Notez que dans ce cas, aucune autorisation explicite n'est requise dans une politique basée sur l'identité.
- Un administrateur principal du compte administrateur du référentiel peut se voir accorder l'accès au référentiel par le biais d'une politique basée sur l'identité. Notez que dans ce cas, aucune autorisation explicite n'est requise dans une politique de domaine ou de référentiel.
- Le principal d'un autre compte n'est autorisé à y accéder que s'il est autorisé par au moins une politique de ressources et au moins une politique basée sur l'identité, aucune politique ne refusant explicitement l'action.

Marquer un dépôt dans CodeArtifact

Les balises sont des paires clé-valeur associées aux ressources AWS. Vous pouvez appliquer des balises à vos référentiels dans CodeArtifact. Pour plus d'informations sur le balisage CodeArtifact des ressources, les cas d'utilisation, les contraintes de clé et de valeur de balise et les types de ressources pris en charge, consultez[Balisage des ressources](#).

Vous pouvez utiliser la CLI pour spécifier des balises lorsque vous créez un référentiel. Vous pouvez utiliser la console ou la CLI pour ajouter ou supprimer des balises et mettre à jour les valeurs des balises dans un référentiel. Vous pouvez ajouter jusqu'à 50 balises à chaque référentiel.

Rubriques

- [Référentiels de balises \(CLI\)](#)
- [Référentiels de balises \(console\)](#)

Référentiels de balises (CLI)

Vous pouvez utiliser la CLI pour gérer les balises du référentiel.

Rubriques

- [Ajouter des balises à un référentiel \(CLI\)](#)
- [Afficher les balises d'un référentiel \(CLI\)](#)
- [Modifier les balises d'un référentiel \(CLI\)](#)
- [Supprimer les balises d'un référentiel \(CLI\)](#)

Ajouter des balises à un référentiel (CLI)

Vous pouvez utiliser la console ou le AWS CLI pour baliser les référentiels.

Pour ajouter une balise à un référentiel lorsque vous le créez, consultez [Création d'un référentiel](#).

Dans ces étapes, nous supposons que vous avez déjà installé une version récente de l' AWS CLI ou que vous avez procédé à une mise à jour vers la version actuelle. Pour plus d'informations, consultez [Installing the AWS Command Line Interface](#) (Installation de).

Depuis le terminal ou la ligne de commande, exécutez la commande `tag-resource`, en spécifiant l'ARN (Amazon Resource Name) du référentiel dans lequel vous souhaitez ajouter des balises ainsi que la clé et la valeur de la balise que vous souhaitez ajouter.

Note

Pour obtenir l'ARN du dépôt, exécutez la `describe-repository` commande suivante :

```
aws codeartifact describe-repository --domain my_domain --repository my_repo --query repository.arn
```

Vous pouvez ajouter plusieurs balises à un référentiel. Par exemple, pour baliser un référentiel nommé *my_repo* dans un domaine *my_domain* doté de deux balises, une clé de balise nommée *key1* avec la valeur de *value1* balise et une clé de balise nommée *key2* avec la valeur de balise de *value2* :

```
aws codeartifact tag-resource --resource-arn arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo --tags key=key1,value=value1 key=key2,value=value2
```

En cas de succès, cette commande n'a aucune sortie.

Afficher les balises d'un référentiel (CLI)

Procédez comme suit pour utiliser le AWS CLI afin d'afficher les AWS balises d'un référentiel. Si aucune balise n'a été ajoutée, la liste renvoyée est vide.

Depuis le terminal ou la ligne de commande, exécutez la commande `list-tags-for-resource`.

Note

Pour obtenir l'ARN du dépôt, exécutez la `describe-repository` commande suivante :

```
aws codeartifact describe-repository --domain my_domain --repository my_repo --query repository.arn
```

Par exemple, pour afficher la liste des clés de balise et des valeurs de balise pour un référentiel nommé *my_repo* dans un domaine *my_domain* portant la valeur `arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo` ARN :

```
aws codeartifact list-tags-for-resource --resource-arn arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo
```

Si elle aboutit, cette commande renvoie des informations similaires à ce qui suit :

```
{  
  "tags": {  
    "key1": "value1",  
    "key2": "value2"  
  }  
}
```

Modifier les balises d'un référentiel (CLI)

Procédez comme suit pour utiliser le AWS CLI pour modifier une balise pour un référentiel. Vous pouvez modifier la valeur d'une clé existante ou ajouter une autre clé.

Sur le terminal ou sur la ligne de commande, exécutez la `tag-resource` commande en spécifiant l'ARN du référentiel dans lequel vous souhaitez mettre à jour une balise, ainsi que la clé et la valeur de la balise.

Note

Pour obtenir l'ARN du dépôt, exécutez la `describe-repository` commande suivante :

```
aws codeartifact describe-repository --domain my_domain --repository my_repo --  
query repository.arn
```

```
aws codeartifact tag-resource --resource-arn arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo --tags key=key1,value=newvalue1
```

En cas de succès, cette commande n'a aucune sortie.

Supprimer les balises d'un référentiel (CLI)

Procédez comme suit pour utiliser le AWS CLI pour supprimer une balise d'un référentiel.

Note

Si vous supprimez un référentiel, toutes les associations de balises sont supprimés du référentiel supprimé. Vous n'avez pas besoin de supprimer les balises avant de supprimer un référentiel.

Sur le terminal ou sur la ligne de commande, exécutez la `untag-resource` commande en spécifiant l'ARN du référentiel dans lequel vous souhaitez supprimer les balises et la clé de balise de la balise que vous souhaitez supprimer.

Note

Pour obtenir l'ARN du dépôt, exécutez la `describe-repository` commande suivante :

```
aws codeartifact describe-repository --domain my_domain --repository my_repo --  
query repository.arn
```

Par exemple, pour supprimer plusieurs balises sur un référentiel nommé *my_repo* dans un domaine nommé *my_domain* avec les clés de balise *key1* et *key2* :

```
aws codeartifact untag-resource --resource-arn arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo --tag-keys key1 key2
```

En cas de succès, cette commande n'a aucune sortie. Après avoir supprimé les balises, vous pouvez afficher les balises restantes dans le référentiel à l'aide de la `list-tags-for-resource` commande.

Référentiels de balises (console)

Vous pouvez utiliser la console ou la CLI pour baliser des ressources.

Rubriques

- [Ajouter des balises à un référentiel \(console\)](#)
- [Afficher les balises d'un référentiel \(console\)](#)
- [Modifier les balises d'un référentiel \(console\)](#)
- [Supprimer des balises d'un référentiel \(console\)](#)

Ajouter des balises à un référentiel (console)

Vous pouvez utiliser la console pour ajouter des balises à un référentiel existant.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Référentiels, choisissez le référentiel auquel vous souhaitez ajouter des balises.
3. Développez la section Détails.
4. Sous Balises du référentiel, s'il n'y a pas de balises sur le référentiel, choisissez Ajouter des balises de référentiel. Si le référentiel contient des balises, choisissez Afficher et modifier les balises du référentiel.
5. Sélectionnez Ajouter une nouvelle balise.
6. Dans les champs Clé et Valeur, entrez le texte de chaque balise que vous souhaitez ajouter. (Le champ Valeur est facultatif.) Par exemple, dans Clé, saisissez **Name**. Dans Value (Valeur), entrez **Test**.

Developer Tools > CodeArtifact > Repositories > reponame > Edit repository

Edit reponame Info

Repository

Repository description - *optional*

1000 character limit

Tags

Tags - *optional*

Key	Value - <i>optional</i>	
<input type="text" value="Name"/> X	<input type="text" value="Test"/> X	Remove

Add new tag

You can add 49 more tags.

▶ **AWS reserved tags**
Resource tags added by other AWS services. These tags cannot be modified.

Upstream repositories - *optional*

Repository name

1. X reponame

Associate upstream repository How to use this input ?

7. (Facultatif) Choisissez Add tag (Ajouter une balise) pour ajouter d'autres lignes et saisir d'autres balises.
 8. Choisissez Mettre à jour le référentiel.

Afficher les balises d'un référentiel (console)

Vous pouvez utiliser la console pour répertorier les balises des référentiels existants.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Référentiels, choisissez le référentiel dans lequel vous souhaitez afficher les balises.
3. Développez la section Détails.
4. Sous Balises du référentiel, choisissez Afficher et modifier les balises du référentiel.

 Note

Si aucune balise n'est ajoutée à ce référentiel, la console affichera Ajouter des balises de référentiel.

Modifier les balises d'un référentiel (console)

Vous pouvez utiliser la console pour modifier les balises qui ont été ajoutées au référentiel.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Référentiels, choisissez le référentiel dans lequel vous souhaitez mettre à jour les balises.
3. Développez la section Détails.
4. Sous Balises du référentiel, choisissez Afficher et modifier les balises du référentiel.

 Note

Si aucune balise n'est ajoutée à ce référentiel, la console affichera Ajouter des balises de référentiel.

5. Dans les champs Clé et Valeur, mettez à jour les valeurs dans chaque champ selon vos besoins. Par exemple, pour la clé **Name**, dans Valeur, remplacez **Test** par **Prod**.
6. Choisissez Mettre à jour le référentiel.

Supprimer des balises d'un référentiel (console)

Vous pouvez utiliser la console pour supprimer des balises des référentiels.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Référentiels, choisissez le référentiel dans lequel vous souhaitez supprimer les balises.
3. Développez la section Détails.
4. Sous Balises du référentiel, choisissez Afficher et modifier les balises du référentiel.

 Note

Si aucune balise n'est ajoutée à ce référentiel, la console affichera Ajouter des balises de référentiel.

5. À côté de la clé et de la valeur de chaque balise que vous souhaitez supprimer, choisissez Supprimer.
6. Choisissez Mettre à jour le référentiel.

Utilisation de référentiels en amont dans CodeArtifact

Un référentiel peut avoir d'autres AWS CodeArtifact référentiels en tant que référentiels en amont. Cela permet à un client du gestionnaire de packages d'accéder aux packages contenus dans plusieurs référentiels à l'aide d'un seul point de terminaison de référentiel.

Vous pouvez ajouter un ou plusieurs référentiels en amont à un AWS CodeArtifact référentiel à l'aide du AWS Management Console SDK. AWS CLI Pour associer un référentiel à un référentiel en amont, vous devez être autorisé à effectuer l'`AssociateWithDownstreamRepository` action sur le référentiel en amont. Pour plus d'informations, consultez [Création d'un référentiel avec un référentiel en amont](#) et [Ajouter ou supprimer des référentiels en amont](#).

Si un dépôt en amont possède une connexion externe à un dépôt public, les référentiels situés en aval de celui-ci peuvent extraire des packages de ce référentiel public. Par exemple, supposons que le référentiel `my_repo` possède un référentiel en amont nommé `upstream` et `upstream` dispose d'une connexion externe à un référentiel `npm` public. Dans ce cas, un gestionnaire de packages connecté `my_repo` peut extraire des packages du référentiel public `npm`. Pour plus d'informations sur la demande de packages à partir de référentiels en amont ou de connexions externes, consultez [Demande d'une version de package avec des référentiels en amont](#) ou [Demande de packages à partir de connexions externes](#).

Rubriques

- [Quelle est la différence entre les référentiels en amont et les connexions externes ?](#)
- [Ajouter ou supprimer des référentiels en amont](#)
- [Connect un CodeArtifact dépôt à un dépôt public](#)
- [Demande d'une version de package avec des référentiels en amont](#)
- [Demande de packages à partir de connexions externes](#)
- [Ordre de priorité du référentiel en amont](#)
- [Comportement des API avec les référentiels en amont](#)

Quelle est la différence entre les référentiels en amont et les connexions externes ?

Dans CodeArtifact, les référentiels en amont et les connexions externes se comportent essentiellement de la même manière, à quelques différences importantes près.

1. Vous pouvez ajouter jusqu'à 10 référentiels en amont à un CodeArtifact référentiel. Vous ne pouvez ajouter qu'une seule connexion externe.
2. Des appels d'API distincts permettent d'ajouter un référentiel en amont ou une connexion externe.
3. Le comportement de rétention des packages est légèrement différent, car les packages demandés auprès des référentiels en amont sont conservés dans ces référentiels. Pour de plus amples informations, veuillez consulter [Rétention des packages dans des référentiels intermédiaires](#).

Ajouter ou supprimer des référentiels en amont

Suivez les étapes décrites dans les sections suivantes pour ajouter ou supprimer des référentiels en amont dans ou depuis un CodeArtifact référentiel. Pour plus d'informations sur les référentiels en amont, consultez [Utilisation de référentiels en amont dans CodeArtifact](#).

Ce guide contient des informations sur la configuration d'autres CodeArtifact référentiels en tant que référentiels en amont. [Pour plus d'informations sur la configuration d'une connexion externe à des référentiels publics tels que npmjs.com, Nuget Gallery, Maven Central ou PyPI, voir Ajouter une connexion externe.](#)

Ajouter ou supprimer des référentiels en amont (console)

Procédez comme indiqué dans la procédure suivante pour ajouter un référentiel en tant que référentiel en amont à l'aide de la CodeArtifact console. Pour plus d'informations sur l'ajout d'un référentiel en amont à l'aide du AWS CLI, consultez [Ajouter ou supprimer des référentiels en amont \(\)AWS CLI](#).

Pour ajouter un référentiel en amont à l'aide de la CodeArtifact console

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domains, puis le nom de domaine qui contient votre référentiel.
3. Choisissez le nom de votre dépôt.
4. Choisissez Modifier.
5. Dans Référentiels en amont, choisissez Associer un référentiel en amont et ajoutez le référentiel que vous souhaitez ajouter en tant que référentiel en amont. Vous ne pouvez ajouter des référentiels que dans le même domaine que les référentiels en amont.

6. Choisissez Mettre à jour le référentiel.

Pour supprimer un référentiel en amont à l'aide de la CodeArtifact console

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domains, puis le nom de domaine qui contient votre référentiel.
3. Choisissez le nom de votre dépôt.
4. Choisissez Modifier.
5. Dans les référentiels en amont, recherchez l'entrée de liste du référentiel en amont que vous souhaitez supprimer et choisissez Dissocier.

Important

Une fois que vous avez supprimé un référentiel en amont d'un CodeArtifact référentiel, les gestionnaires de packages n'auront pas accès aux packages du référentiel en amont ni à aucun de ses référentiels en amont.

6. Choisissez Mettre à jour le référentiel.

Ajouter ou supprimer des référentiels en amont ()AWS CLI

Vous pouvez ajouter ou supprimer les référentiels en amont d'un CodeArtifact dépôt à l'aide du AWS Command Line Interface (AWS CLI). Pour ce faire, utilisez la `update-repository` commande et spécifiez les référentiels en amont à l'aide du `--upstreams` paramètre.

Vous ne pouvez ajouter des référentiels que dans le même domaine que les référentiels en amont.

Pour ajouter des référentiels en amont ()AWS CLI

1. Si ce n'est pas le cas, suivez les étapes décrites [Configuration avec AWS CodeArtifact](#) pour installer et configurer le AWS CLI avec CodeArtifact.
2. Utilisez la `aws codeartifact update-repository` commande avec l'`--upstreams` indicateur pour ajouter des référentiels en amont.

Note

L'appel de la `update-repository` commande remplace les référentiels en amont configurés existants par la liste des référentiels fournie avec l'`--upstreams` indicateur. Si vous souhaitez ajouter des référentiels en amont et conserver les référentiels existants, vous devez inclure les référentiels en amont existants dans l'appel.

L'exemple de commande suivant ajoute deux référentiels en amont à un référentiel nommé `my_repo` qui se trouve dans un domaine nommé `my_domain`. L'ordre des référentiels en amont dans le `--upstreams` paramètre détermine leur priorité de recherche lorsqu'un package est demandé au `my_repo` référentiel. Pour de plus amples informations, veuillez consulter [Ordre de priorité du référentiel en amont](#).

Pour plus d'informations sur la connexion à des référentiels externes publics tels que `npmjs.com` ou `Maven Central`, consultez [Connect un CodeArtifact dépôt à un dépôt public](#)

```
aws codeartifact update-repository \
  --repository my_repo \
  --domain my_domain \
  --domain-owner 111122223333 \
  --upstreams repositoryName=upstream-1 repositoryName=upstream-2
```

La sortie contient les référentiels en amont, comme suit.

```
{
  "repository": {
    "name": "my_repo",
    "administratorAccount": "123456789012",
    "domainName": "my_domain",
    "domainOwner": "111122223333",
    "arn": "arn:aws:codeartifact:us-east-2:111122223333:repository/my_domain/my_repo",
    "upstreams": [
      {
        "repositoryName": "upstream-1"
      },
      {
        "repositoryName": "upstream-2"
      }
    ]
  }
}
```

```
        },
    ],
    "externalConnections": []
}
}
```

Pour supprimer un référentiel en amont (AWS CLI)

1. Si ce n'est pas le cas, suivez les étapes décrites [Configuration avec AWS CodeArtifact](#) pour installer et configurer le AWS CLI avec CodeArtifact.
2. Pour supprimer des référentiels en amont d'un CodeArtifact référentiel, utilisez la `update-repository` commande avec l'`--upstreams` indicateur. La liste des référentiels fournie à la commande sera le nouvel ensemble de référentiels en amont pour le CodeArtifact référentiel. Incluez les référentiels en amont existants que vous souhaitez conserver et omettez les référentiels en amont que vous souhaitez supprimer.

Pour supprimer tous les référentiels en amont d'un référentiel, utilisez la `update-repository` commande et incluez `--upstreams` sans argument. `my_repo` qui suit supprime les référentiels en amont d'un référentiel nommé contenu dans un domaine nommé `my_domain`.

```
aws codeartifact update-repository \
--repository my_repo \
--domain my_domain \
--domain-owner 111122223333 \
--upstreams
```

Le résultat indique que la liste des `upstreams` est vide.

```
{
    "repository": {
        "name": "my_repo",
        "administratorAccount": "123456789012",
        "domainName": "my_domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:us-east-2:111122223333:repository/my_domain/my_repo",
        "upstreams": [],
        "externalConnections": []
    }
}
```

}

Connect un CodeArtifact dépôt à un dépôt public

Vous pouvez ajouter une connexion externe entre un CodeArtifact référentiel et un référentiel public externe tel que <https://npmjs.com> le [référentiel Maven Central](#). Ensuite, lorsque vous demandez un package au CodeArtifact référentiel qui n'est pas déjà présent dans le référentiel, le package peut être récupéré à partir de la connexion externe. Cela permet de consommer les dépendances open source utilisées par votre application.

Dans CodeArtifact, la manière prévue d'utiliser les connexions externes est d'avoir un référentiel par domaine avec une connexion externe à un référentiel public donné. Par exemple, si vous souhaitez vous connecter à npmjs.com, configurez un référentiel de votre domaine avec une connexion externe à npmjs.com et configurez tous les autres référentiels avec une connexion amont. De cette façon, tous les référentiels peuvent utiliser les packages qui ont déjà été récupérés sur npmjs.com, plutôt que de les récupérer et de les stocker à nouveau.

Rubriques

- [Connect à un référentiel externe \(console\)](#)
- [Se connecter à un référentiel externe \(CLI\)](#)
- [Référentiels de connexions externes pris en charge](#)
- [Supprimer une connexion externe \(CLI\)](#)

Connect à un référentiel externe (console)

Lorsque vous utilisez la console pour ajouter une connexion à un référentiel externe, les événements suivants se produisent :

1. Un `-store` dépôt pour le dépôt externe sera créé dans votre CodeArtifact domaine s'il n'en existe pas déjà un. Ces `-store` référentiels se comportent comme des référentiels intermédiaires entre votre dépôt et le référentiel externe et vous permettent de vous connecter à plusieurs référentiels externes.
2. Le `-store` référentiel approprié est ajouté en amont à votre référentiel.

La liste suivante répertorie chaque `-store` référentiel CodeArtifact et le référentiel externe auquel ils se connectent.

1. `cargo-store` est connecté à crates.io.
2. `clojars-store` est connecté au dépôt Clojars.
3. `commonsware-store` est connecté au référentiel CommonsWare Android.
4. `google-android-store` est connecté à Google Android.
5. `gradle-plugins-store` est connecté aux plugins Gradle.
6. `maven-central-store` est connecté au référentiel central Maven.
7. `npm-store` est connecté à npmjs.com.
8. `nuget-store` est connecté à nuget.org.
9. `pypi-store` est connecté à la Python Packaging Authority.
10. `rubygems-store` est connecté à RubyGems .org.

Pour se connecter à un référentiel externe (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domains, puis le nom de domaine qui contient votre référentiel.
3. Choisissez le nom de votre dépôt.
4. Choisissez Modifier.
5. Dans Référentiels en amont, choisissez Associer un référentiel en amont et ajoutez le `-store` référentiel approprié connecté en tant que référentiel en amont.
6. Choisissez Mettre à jour le référentiel.

Une fois le `-store` référentiel ajouté en tant que référentiel en amont, les gestionnaires de packages connectés à votre CodeArtifact référentiel peuvent récupérer les packages depuis le référentiel externe correspondant.

Se connecter à un référentiel externe (CLI)

Vous pouvez utiliser le AWS CLI pour connecter votre CodeArtifact dépôt à un référentiel externe en ajoutant une connexion externe directement au référentiel. Cela permettra aux utilisateurs connectés

au CodeArtifact référentiel, ou à l'un de ses référentiels en aval, de récupérer des packages depuis le référentiel externe configuré. Chaque CodeArtifact dépôt ne peut avoir qu'une seule connexion externe.

Il est recommandé d'avoir un dépôt par domaine avec une connexion externe à un dépôt public donné. Pour connecter d'autres référentiels au référentiel public, ajoutez le référentiel avec la connexion externe en amont. Si vous ou un autre membre de votre domaine avez déjà configuré des connexions externes dans la console, votre domaine possède probablement déjà un `-store` référentiel avec une connexion externe au référentiel public auquel vous souhaitez vous connecter. Pour plus d'informations sur `-store` les référentiels et la connexion à la console, consultez [Connect à un référentiel externe \(console\)](#).

Pour ajouter une connexion externe à un CodeArtifact référentiel (CLI)

- `associate-external-connection` À utiliser pour ajouter une connexion externe. L'exemple suivant connecte un référentiel au registre public npm, npmjs.com. Pour obtenir la liste des référentiels externes pris en charge, consultez [Référentiels de connexions externes pris en charge](#).

```
aws codeartifact associate-external-connection --external-connection public:npmjs \
--domain my_domain --domain-owner 111122223333 --repository my_repo
```

Exemple de sortie :

```
{  
  "repository": {  
    "name": my_repo,  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-  
west-2:111122223333:repository/my_domain/my_repo",  
    "description": "A description of my_repo",  
    "upstreams": [],  
    "externalConnections": [  
      {  
        "externalConnectionName": "public:npmjs",  
        "packageFormat": "npm",  
        "status": "AVAILABLE"  
      }  
    ]  
  }  
}
```

```
    }  
}
```

Après avoir ajouté une connexion externe, consultez [Demande de packages à partir de connexions externes](#) pour plus d'informations sur la demande de packages depuis un référentiel externe avec une connexion externe.

Référentiels de connexions externes pris en charge

CodeArtifact prend en charge une connexion externe aux référentiels publics suivants. Pour utiliser la CodeArtifact CLI afin de spécifier une connexion externe, utilisez la valeur du `--external-connection` paramètre dans la colonne Nom lorsque vous exécutez la `associate-external-connection` commande.

Type de référentiel	Description	Nom
Maven	Référentiel Clojars	public:maven-clojars
Maven	CommonsWare Référentiel Android	public:maven-commonsware
Maven	Référentiel Google Android	public:maven-googleandroid
Maven	Référentiel de plugins Gradle	public:maven-gradleplugins
Maven	Maven Central	public:maven-central
npm	registre public npm	public:npmjs
NuGet	NuGet Galerie	public:nuget-org
Python	Index des packages Python	public:pypi
Ruby	RubyGems.org	public:ruby-gems-org

Type de référentiel	Description	Nom
Rust	Crates.io	public:crates-io

Supprimer une connexion externe (CLI)

Pour supprimer une connexion externe ajoutée à l'aide de la `associate-external-connection` commande contenue dans le AWS CLI, utilisez `disassociate-external-connection`.

```
aws codeartifact disassociate-external-connection --external-connection public:npmjs \
--domain my_domain --domain-owner 111122223333 --repository my_repo
```

Exemple de sortie :

```
{
  "repository": {
    "name": my_repo
    "administratorAccount": "123456789012",
    "domainName": "my_domain",
    "domainOwner": "111122223333",
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo",
    "description": "A description of my_repo",
    "upstreams": [],
    "externalConnections": []
  }
}
```

Demande d'une version de package avec des référentiels en amont

Lorsqu'un client (par exemple, npm) demande une version de package à partir d'un CodeArtifact référentiel nommé *my_repo* contenant plusieurs référentiels en amont, les événements suivants peuvent se produire :

- S'il *my_repo* contient la version du package demandée, il est renvoyé au client.
- S'il *my_repo* ne contient pas la version du package demandée, CodeArtifact recherchez-la dans les *my_repo* référentiels en amont. Si la version du package est trouvée, une référence à celle-ci est *my_repo* copiée et la version du package est renvoyée au client.

- Si `my_repo` ni les référentiels en amont ne contiennent la version du package, une `Not Found` réponse HTTP 404 est renvoyée au client.

Lorsque vous ajoutez des référentiels en amont à l'aide de la `update-repository` commande `create-repository` ou, l'ordre dans lequel ils sont transmis au `--upstreams` paramètre détermine leur priorité lorsqu'une version de package est demandée. Spécifiez les référentiels en amont `--upstreams` dans l'ordre que vous CodeArtifact souhaitez utiliser lorsqu'une version de package est demandée. Pour de plus amples informations, veuillez consulter [Ordre de priorité du référentiel en amont](#).

Le nombre maximum de référentiels en amont autorisés pour un référentiel est de 10. Comme les référentiels en amont direct peuvent également avoir leurs propres référentiels en amont, ils CodeArtifact peuvent rechercher des versions de packages dans plus de 10 référentiels. Le nombre maximum de référentiels consultés CodeArtifact lorsqu'une version de package est demandée est de 25.

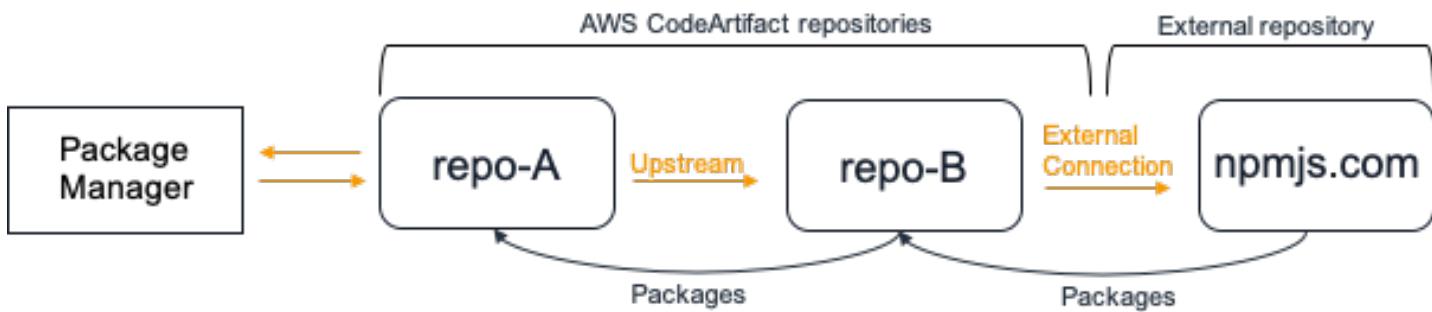
Conservation des packages depuis les référentiels en amont

Si une version de package demandée est trouvée dans un référentiel en amont, une référence à celle-ci est conservée et est toujours disponible dans le référentiel en aval. La version du package conservée n'est affectée par aucun des éléments suivants :

- Suppression du référentiel en amont.
- Déconnexion du référentiel en amont du référentiel en aval.
- Suppression de la version du package du référentiel en amont.
- Modification de la version du package dans le référentiel en amont (par exemple, en y ajoutant une nouvelle ressource).

Récupérez des packages via une relation en amont

Si un CodeArtifact référentiel entretient une relation en amont avec un référentiel doté d'une connexion externe, les demandes de packages ne figurant pas dans le référentiel en amont sont copiées depuis le référentiel externe. Par exemple, considérez la configuration suivante : un référentiel nommé `repo-A` possède un référentiel en amont nommé `repo-B`. `repo-B` dispose d'une connexion externe à <https://npmjs.com>.



S'il npm est configuré pour utiliser le repo-A référentiel, l'exécution `npm install` déclenche la copie des packages depuis et <https://npmjs.com> vers repo-B. Les versions installées sont également intégrées repo-A. L'exemple suivant permet d'installer `lodash`

```

$ npm config get registry
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my-
downstream-repo/
$ npm install lodash
+ lodash@4.17.20
added 1 package from 2 contributors in 6.933s

```

Après l'exécution `npm install`, repo-A contient uniquement la dernière version (`lodash 4.17.20`) car c'est la version qui a été récupérée par npm. repo-A

```

aws codeartifact list-package-versions --repository repo-A --domain my_domain \
--domain-owner 111122223333 --format npm --package lodash

```

Exemple de sortie :

```

{
  "package": "lodash",
  "format": "npm",
  "versions": [
    {
      "version": "4.17.15",
      "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    }
  ]
}

```

Étant donné qu'il `repo-B` possède une connexion externe à <https://npmjs.com>, toutes les versions de package importées <https://npmjs.com> sont stockées dans `repo-B`. Ces versions de package auraient pu être récupérées par n'importe quel dépôt en aval ayant une relation en amont avec `repo-B`.

Le contenu de `repo-B` fournit un moyen de voir tous les packages et versions de packages importés <https://npmjs.com> au fil du temps. Par exemple, pour voir toutes les versions du `lodash` package importées au fil du temps, vous pouvez utiliser `list-package-versions` ce qui suit.

```
aws codeartifact list-package-versions --repository repo-B --domain my_domain \
    --domain-owner 111122223333 --format npm --package lodash --max-results 5
```

Exemple de sortie :

```
{
  "package": "lodash",
  "format": "npm",
  "versions": [
    {
      "version": "0.10.0",
      "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    },
    {
      "version": "0.2.2",
      "revision": "REVISION-2-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    },
    {
      "version": "0.2.0",
      "revision": "REVISION-3-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    },
    {
      "version": "0.2.1",
      "revision": "REVISION-4-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    },
    {
      "version": "0.1.0",
      "revision": "REVISION-5-SAMPLE-6C81EFF7DA55CC",
      "status": "Published"
    }
}
```

```
],
"nextToken": "eyJsaXN0UGFja2FnZVZlcNpb25zVG9rZW4iOiIwLjIuMiJ9"
}
```

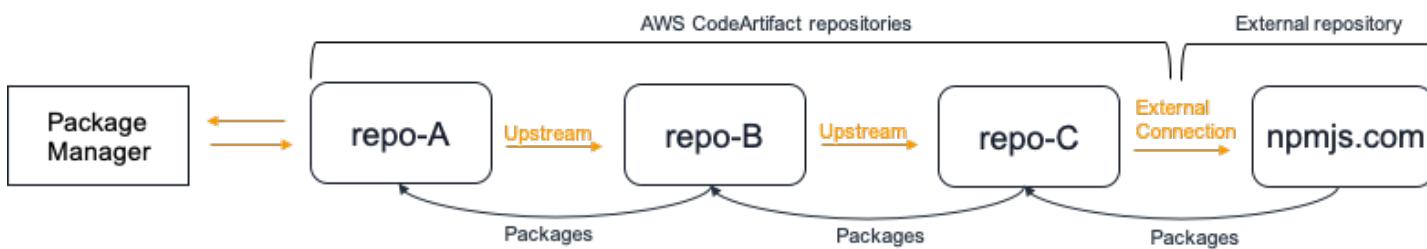
Rétention des packages dans des référentiels intermédiaires

CodeArtifact permet de chaîner des référentiels en amont. Par exemple, `repo-A` peut avoir `repo-B` comme amont et `repo-B` peut avoir `repo-C` comme amont. Cette configuration permet aux versions du package d'être intégrées `repo-B` et `repo-C` disponibles auprès de `repo-A`.



Lorsqu'un gestionnaire de packages se connecte au référentiel `repo-A` et récupère une version de package à partir du référentiel `repo-C`, la version du package n'est pas conservée dans le référentiel `repo-B`. La version du package ne sera conservée que dans le référentiel le plus en aval, dans cet exemple, `repo-A`. Il ne sera conservé dans aucun dépôt intermédiaire. Cela est également vrai pour les chaînes plus longues ; par exemple, s'il y avait quatre référentiels `repo-A`, `repo-B`, `repo-C`, et `repo-D` et qu'un gestionnaire de packages était connecté pour `repo-A` récupérer une version de package `repo-D`, la version du package serait conservée dans ou pas dans `repo-A`, `repo-B` ou `repo-C`.

Le comportement de rétention des packages est similaire lors de l'extraction d'une version de package depuis un référentiel externe, sauf que la version du package est toujours conservée dans le référentiel auquel la connexion externe est attachée. Par exemple, `repo-A` a `repo-B` en amont. `repo-B` possède `repo-C` une connexion amont et `npmjs.com` est `repo-C` également configuré en tant que connexion externe ; voir le schéma suivant.



Si un gestionnaire de packages connecté à `repo-A` demande une version de package, `lodash 4.17.20` par exemple, et que la version du package n'est présente dans aucun des trois référentiels,

elle sera récupérée sur npmjs.com. Lorsque `lodash` 4.17.20 est récupéré, il est conservé car il s'agit du dépôt le plus en aval et **repo-A** **repo-C** car la connexion externe à npmjs.com y est attachée. `lodash` 4.17.20 ne sera pas conservé `repo-B` car il s'agit d'un dépôt intermédiaire.

Demande de packages à partir de connexions externes

Les sections suivantes décrivent comment demander un package à partir d'une connexion externe et le CodeArtifact comportement attendu lors de la demande d'un package.

Rubriques

- [Récupérer des packages depuis une connexion externe](#)
- [Latence de connexion externe](#)
- [CodeArtifact comportement lorsqu'un référentiel externe n'est pas disponible](#)
- [Disponibilité des nouvelles versions du package](#)
- [Importation de versions de packages contenant plusieurs actifs](#)

Récupérer des packages depuis une connexion externe

Pour récupérer des packages depuis une connexion externe une fois que vous les avez ajoutés à votre CodeArtifact dépôt, comme décrit dans [Connect un CodeArtifact dépôt à un dépôt public](#), configurez votre gestionnaire de packages pour qu'il utilise votre référentiel et installe les packages.

Note

Les instructions suivantes utilisent `npm`, pour consulter les instructions de configuration et d'utilisation pour d'autres types de packages [Utilisation CodeArtifact avec Maven](#), [voir Utilisation CodeArtifact avec NuGet](#), ou [Utilisation CodeArtifact avec Python](#).

Pour récupérer des packages depuis une connexion externe

1. Configurez et authentifiez votre gestionnaire de packages auprès de votre CodeArtifact référentiel. Pour `npm`, utilisez la commande `aws codeartifact login` suivante.

```
aws codeartifact login --tool npm --domain my_domain --domain-owner 111122223333 --repository my_repo
```

2. Demandez le package depuis le dépôt public. Pour npm, utilisez la `npm install` commande suivante en `lodash` remplaçant par le package que vous souhaitez installer.

```
npm install lodash
```

3. Une fois le package copié dans votre CodeArtifact dépôt, vous pouvez utiliser les `list-package-versions` commandes `list-packages` et pour l'afficher.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --repository my_repo
```

Exemple de sortie :

```
{  
  "packages": [  
    {  
      "format": "npm",  
      "package": "lodash"  
    }  
  ]  
}
```

La `list-package-versions` commande répertorie toutes les versions du package copiées dans votre CodeArtifact dépôt.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333 --repository my_repo --format npm --package lodash
```

Exemple de sortie :

```
{  
  "defaultDisplayVersion": "1.2.5"  
  "format": "npm",  
  "package": "lodash",  
  "namespace": null,  
  "versions": [  
    {  
      "version": "1.2.5",  
      "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",  
      "status": "Published"  
    }  
  ]  
}
```

```
    }  
]  
}
```

Latence de connexion externe

Lorsque vous récupérez un package depuis un dépôt public à l'aide d'une connexion externe, il y a un délai entre le moment où le package est extrait du référentiel public et celui où il est stocké dans votre CodeArtifact dépôt. Par exemple, supposons que vous ayez installé la version 1.2.5 du package npm « `lodash` » comme décrit dans. [Récupérer des packages depuis une connexion externe](#) Bien que la commande `npm install lodash` ait été exécutée avec succès, la version du package n'apparaît peut-être pas encore dans votre CodeArtifact dépôt. Il faut généralement environ 3 minutes pour que la version du package apparaisse dans votre dépôt, bien que cela puisse parfois prendre plus de temps.

En raison de cette latence, vous avez peut-être réussi à récupérer une version de package, mais vous ne pouvez peut-être pas encore voir la version dans votre référentiel dans la CodeArtifact console ou lorsque vous appelez les opérations `ListPackages` et `ListPackageVersions` API. Une fois la version du package conservée de manière asynchrone, elle sera visible dans la console et via les requêtes d'API.

CodeArtifact comportement lorsqu'un référentiel externe n'est pas disponible

Il arrive parfois qu'un dépôt externe connaisse une panne, ce qui signifie qu'il CodeArtifact ne peut pas récupérer les packages ou que la récupération des packages est beaucoup plus lente que d'habitude. Dans ce cas, les versions de package déjà extraites d'un dépôt externe (par exemple `npmjs.com`) et stockées dans un CodeArtifact référentiel continueront d'être disponibles au téléchargement depuis. Cependant, les packages qui ne sont pas déjà stockés dans ce référentiel CodeArtifact peuvent ne pas être disponibles, même lorsqu'une connexion externe à ce référentiel a été configurée. Par exemple, votre CodeArtifact dépôt peut contenir la version du package npm, `lodash 4.17.19` car c'est ce que vous avez utilisé dans votre application jusqu'à présent. Lorsque vous souhaitez effectuer une mise à niveau vers `4.17.20`, votre CodeArtifact récupérerez normalement cette nouvelle version sur `npmjs.com` et la stockerez dans votre référentiel. Toutefois, si `npmjs.com` est en panne, cette nouvelle version ne sera pas disponible. La seule solution consiste à réessayer ultérieurement une fois que `npmjs.com` sera rétabli.

Les pannes de référentiels externes peuvent également affecter la publication de nouvelles versions de packages sur CodeArtifact. Dans un référentiel avec une connexion externe configurée, n'CodeArtifact autorisera pas la publication d'une version de package déjà présente dans le référentiel externe. Pour de plus amples informations, veuillez consulter [Vue d'ensemble des packages](#). Cependant, dans de rares cas, une panne de référentiel externe peut signifier que CodeArtifact celui-ci ne dispose pas up-to-date d'informations sur les packages et les versions de packages présents dans un référentiel externe. Dans ce cas, CodeArtifact peut autoriser la publication d'une version de package qu'il refuserait normalement.

Disponibilité des nouvelles versions du package

Pour qu'une version de package dans un dépôt public tel que [npmjs.com](#) soit disponible via un CodeArtifact référentiel, elle doit d'abord être ajoutée à un cache de métadonnées de package régional. Ce cache est géré par chaque CodeArtifact AWS région et contient des métadonnées décrivant le contenu des référentiels publics pris en charge. En raison de ce cache, il existe un délai entre le moment où une nouvelle version du package est publiée dans un dépôt public et le moment où elle est disponible sur CodeArtifact. Ce délai varie selon le type de colis.

Pour les packages npm, Python et Nuget, il peut y avoir un délai allant jusqu'à 30 minutes entre le moment où une nouvelle version du package est publiée sur [npmjs.com](#), [pypi.org](#) ou [nuget.org](#) et le moment où elle est disponible pour installation à partir d'un référentiel. CodeArtifact synchronise automatiquement les métadonnées de ces deux référentiels pour s'assurer que le cache est à jour.

Pour les packages Maven, il peut y avoir un délai allant jusqu'à 3 heures entre le moment où une nouvelle version de package est publiée dans un dépôt public et le moment où elle est disponible pour installation à partir d'un CodeArtifact référentiel. CodeArtifact vérifiera les nouvelles versions d'un package au maximum une fois toutes les 3 heures. La première demande pour un nom de package donné après l'expiration de la durée de vie du cache de 3 heures entraînera l'importation de toutes les nouvelles versions de ce package dans le cache régional.

Pour les packages Maven couramment utilisés, les nouvelles versions sont généralement importées toutes les 3 heures, car le taux élevé de demandes signifie que le cache est souvent mis à jour dès que sa durée de vie a expiré. Pour les packages peu utilisés, le cache ne contiendra pas la dernière version tant qu'une version du package n'aura pas été demandée dans un CodeArtifact référentiel. Lors de la première demande, seules les versions précédemment importées seront disponibles CodeArtifact, mais cette demande entraînera la mise à jour du cache. Lors de demandes ultérieures, les nouvelles versions du package seront ajoutées au cache et pourront être téléchargées.

Importation de versions de packages contenant plusieurs actifs

Les packages Maven et Python peuvent avoir plusieurs actifs par version de package. Cela rend l'importation de packages de ces formats plus complexe que celle de npm et de NuGet packages, qui n'ont qu'un seul actif par version de package. Pour une description des actifs importés pour ces types de packages et de la manière dont les actifs récemment ajoutés sont mis à disposition, voir [Demande de packages Python depuis des connexions en amont et externes](#) et [Demande de packages Maven depuis des connexions en amont et externes](#)

Ordre de priorité du référentiel en amont

Lorsque vous demandez une version de package à partir d'un référentiel comportant un ou plusieurs référentiels en amont, leur priorité correspond à l'ordre dans lequel ils ont été répertoriés lors de l'appel de la `update-repository` commande `create-repository` or. Lorsque la version du package demandée est trouvée, la recherche s'arrête, même si la recherche n'a pas été effectuée dans tous les référentiels en amont. Pour de plus amples informations, veuillez consulter [Ajouter ou supprimer des référentiels en amont \(\)AWS CLI](#).

Utilisez la `describe-repository` commande pour voir l'ordre de priorité.

```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-owner 111122223333
```

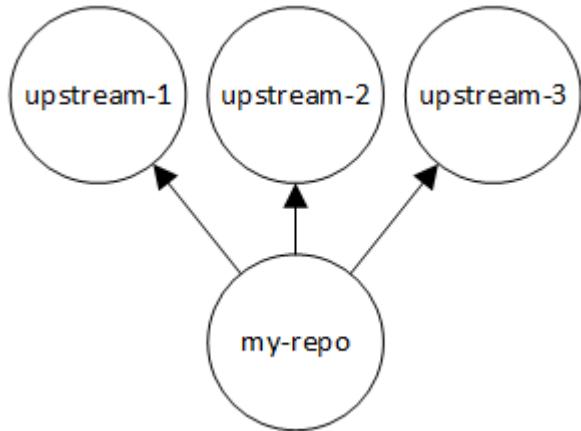
Le résultat peut être le suivant. Cela montre que la priorité du dépôt en amont est `upstream-1` la première, `upstream-2` la deuxième et la `upstream-3` troisième.

```
{  
  "repository": {  
    "name": "my_repo",  
    "administratorAccount": "123456789012",  
    "domainName": "my_domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-  
east-1:111122223333:repository/my_domain/my_repo",  
    "description": "My new repository",  
    "upstreams": [  
      {  
        "repositoryName": "upstream-1"  
      },
```

```
        {
            "repositoryName": "upstream-2"
        },
        {
            "repositoryName": "upstream-3"
        }
    ],
    "externalConnections": []
}
}
```

Exemple d'ordre de priorité simple

Dans le schéma suivant, le `my_repo` référentiel comporte trois référentiels en amont. L'ordre de priorité des référentiels en amont est `upstream-1,upstream-2,upstream-3`.



Une demande de version de package in `my_repo` recherche les référentiels dans l'ordre suivant jusqu'à ce qu'elle soit trouvée ou jusqu'à ce qu'une `Not Found` réponse HTTP 404 soit renvoyée au client :

1. `my_repo`
2. `upstream-1`
3. `upstream-2`
4. `upstream-3`

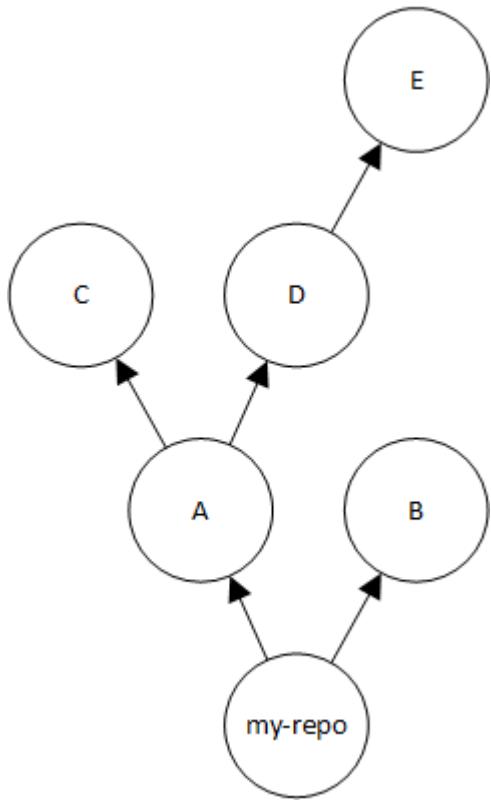
Si la version du package est trouvée, la recherche s'arrête, même si elle n'a pas été recherchée dans tous les référentiels en amont. Par exemple, si la version du package est trouvée dans `upstream-1`, la recherche s'arrête et CodeArtifact ne porte pas sur `upstream-2` ou `upstream-3`.

Lorsque vous utilisez la AWS CLI commande `list-package-versions` pour répertorier les versions du package `my_repo`, elle n'y apparaît qu'en `my_repo`. Il ne répertorie pas les versions des packages dans les référentiels en amont.

Exemple d'ordre de priorité complexe

Si un dépôt en amont possède ses propres référentiels en amont, la même logique est utilisée pour trouver une version de package avant de passer au dépôt en amont suivant. Supposons, par exemple, que votre `my_repo` référentiel comporte deux référentiels en amont, A et B. Si le référentiel A possède des référentiels en amont, une demande de version de package apparaît d'abord dans `my_repoA`, puis dans les référentiels en amont de A, et ainsi de suite.

Dans le schéma suivant, le `my_repo` référentiel contient les référentiels en amont. Le référentiel en amont comporte deux référentiels en amont et en D possède un en amont. Les référentiels en amont situés au même niveau dans le diagramme apparaissent dans leur ordre de priorité, de gauche à droite (le référentiel A a un ordre de priorité plus élevé que le référentiel B, et le référentiel C a un ordre de priorité plus élevé que le référentiel D).



Dans cet exemple, une demande de version de package est consultée dans `my_repo` les référentiels dans l'ordre suivant jusqu'à ce qu'elle soit trouvée ou jusqu'à ce qu'un gestionnaire de packages renvoie une `Not Found` réponse HTTP 404 au client :

1. my_repo
2. A
3. C
4. D
5. E
6. B

Comportement des API avec les référentiels en amont

Lorsque vous appelez certains CodeArtifact APIs référentiels connectés à des référentiels en amont, le comportement peut être différent selon que les packages ou les versions de packages sont stockés dans le référentiel cible ou dans le référentiel amont. Leur comportement APIs est documenté ici.

Pour plus d'informations CodeArtifact APIs, consultez la [référence des CodeArtifact API](#).

La plupart APIs des références à un package ou à une version de package renvoient une `ResourceNotFound` erreur si la version de package spécifiée n'est pas présente dans le référentiel cible. Cela est vrai même si le package ou la version du package est présent dans un référentiel en amont. En fait, les référentiels en amont sont ignorés lors de leur appel. APIs Il s' APIs agit de :

- `DeletePackageVersions`
- `DescribePackageVersion`
- `GetPackageVersionAsset`
- `GetPackageVersionReadme`
- `ListPackages`
- `ListPackageVersionAssets`
- `ListPackageVersionDependencies`
- `ListPackageVersions`
- `UpdatePackageVersionsStatus`

Pour illustrer ce comportement, nous avons deux référentiels : `target-repo` et `upstream-repo`. `target-repo` est vide et a été `upstream-repo` configuré en tant que référentiel en amont. `upstream-repo` contient le package `lodash` npm.

Lorsque l'DescribePackageVersionAPI est appelée `upstream-repo`, qui contient le `lodash` package, nous obtenons le résultat suivant :

```
{  
  "packageVersion": {  
    "format": "npm",  
    "packageName": "lodash",  
    "displayName": "lodash",  
    "version": "4.17.20",  
    "summary": "Lodash modular utilities.",  
    "homePage": "https://lodash.com/",  
    "sourceCodeRepository": "https://github.com/lodash/lodash.git",  
    "publishedTime": "2020-10-14T11:06:10.370000-04:00",  
    "licenses": [  
      {  
        "name": "MIT"  
      }  
    ],  
    "revision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+F1lPs=",  
    "status": "Published"  
  }  
}
```

Lorsque vous appelez la même API `target-repo`, qui est vide mais `upstream-repo` configurée en amont, nous obtenons le résultat suivant :

```
An error occurred (ResourceNotFoundException) when calling the DescribePackageVersion  
operation:  
Package not found in repository. RepoId: repo-id, Package =  
PackageCoordinate{packageType=npm, packageName=lodash},
```

L'CopyPackageVersionsAPI se comporte différemment. Par défaut, CopyPackageVersions l'API copie uniquement les versions de package stockées dans le référentiel cible. Si une version de package est stockée dans le référentiel en amont mais pas dans le référentiel cible, elle ne sera pas copiée. Pour inclure les versions de packages qui sont stockées uniquement dans le référentiel en amont, définissez la valeur de `includeFromUpstream` à `true` dans votre demande d'API.

Pour plus d'informations sur l'CopyPackageVersionsAPI, consultez [Copier des packages entre des référentiels](#).

Utilisation de packages dans CodeArtifact

Les rubriques suivantes expliquent comment effectuer des actions sur des packages à l'aide de la CodeArtifact CLI et de l'API.

Rubriques

- [Vue d'ensemble des packages](#)
- [Lister les noms de packages](#)
- [Lister les versions des packages](#)
- [Répertorier les actifs de la version](#)
- [Télécharger les ressources de la version du package](#)
- [Copier des packages entre des référentiels](#)
- [Supprimer un package ou une version de package](#)
- [Afficher et mettre à jour les détails et les dépendances des versions du package](#)
- [État de version du package de mise à jour](#)
- [Modification des contrôles d'origine des packages](#)

Vue d'ensemble des packages

Un package est un ensemble de logiciels et de métadonnées nécessaires pour résoudre les dépendances et installer le logiciel. Dans CodeArtifact, un package se compose d'un nom de package, d'un espace de [noms](#) facultatif tel que `@types in @types/node`, d'un ensemble de versions de package et de métadonnées au niveau du package telles que des balises npm.

Table des matières

- [Formats de packages pris en charge](#)
- [Publication de packages](#)
 - [Autorisations de publication](#)
 - [Remplacement des actifs du package](#)
 - [Packages privés et référentiels publics](#)
 - [Publication de versions de packages corrigées](#)
 - [Limites de taille des ressources pour la publication](#)

- [Latence de publication](#)
- [État de la version du package](#)
- [Nom du package, version du package et normalisation du nom des actifs](#)

Formats de packages pris en charge

AWS CodeArtifact [prend en charge les formats de package Cargo, generic, Maven, npm, NuGetPyPI, Ruby et Swift.](#)

Publication de packages

Vous pouvez publier de nouvelles versions de n'importe quel [format de package pris en charge](#) CodeArtifact dans un référentiel à l'aide d'outils tels que npm twineMaven,Gradle,nuget,, etdotnet.

Autorisations de publication

Votre utilisateur ou rôle AWS Identity and Access Management (IAM) doit être autorisé à publier dans le référentiel de destination. Les autorisations suivantes sont requises pour publier des packages :

- Cargo : codeartifact:PublishPackageVersion
- générique : codeartifact:PublishPackageVersion
- Maven : codeartifact:PublishPackageVersion et codeartifact:PutPackageMetadata
- npm : codeartifact:PublishPackageVersion
- NuGet: codeartifact:PublishPackageVersion et codeartifact:ReadFromRepository
- Python : codeartifact:PublishPackageVersion
- Rubis : codeartifact:PublishPackageVersion
- Rapide : codeartifact:PublishPackageVersion

Dans la liste d'autorisations précédente, votre politique IAM doit spécifier la package ressource pour les codeartifact:PutPackageMetadata autorisations codeartifact:PublishPackageVersion et. Il doit également spécifier la repository ressource pour l'codeartifact:ReadFromRepository autorisation.

Pour plus d'informations sur les autorisations d' CodeArtifact entrée, consultez [AWS CodeArtifact référence aux autorisations](#).

Remplacement des actifs du package

Vous ne pouvez pas republier une ressource de package qui existe déjà avec un contenu différent. Supposons, par exemple, que vous ayez déjà publié un package Maven avec un actif `mypackage-1.0.jar`. Vous ne pouvez publier à nouveau cette ressource que si la somme de contrôle des anciennes et des nouvelles ressources est identique. Pour republier la même ressource avec un nouveau contenu, supprimez d'abord la version du package à l'aide de la `delete-package-versions` commande. Toute tentative de republication du même nom de ressource avec un contenu différent entraînera une erreur de conflit HTTP 409.

Pour les formats de package qui prennent en charge plusieurs actifs (générique, PyPI et Maven), vous pouvez ajouter de nouveaux actifs portant des noms différents à une version de package existante, en supposant que vous disposez des autorisations requises. Pour les packages génériques, vous pouvez ajouter de nouveaux actifs tant que la version du package est dans l'`Unfinished` état actuel. Étant donné que npm ne prend en charge qu'un seul actif par version de package, pour modifier une version de package publiée de quelque manière que ce soit, vous devez d'abord la supprimer à l'aide `delete-package-versions` de.

Si vous essayez de republier une ressource qui existe déjà (par exemple, `mypackage-1.0.jar`) et que le contenu de la ressource publiée et de la nouvelle ressource est identique, l'opération aboutira car elle est idempotente.

Packages privés et référentiels publics

CodeArtifact ne publie pas les packages stockés dans CodeArtifact des référentiels publics tels que `npmjs.com` ou `Maven Central`. CodeArtifact importe des packages depuis des référentiels publics vers un CodeArtifact référentiel, mais ne déplace jamais les packages dans l'autre sens. Les packages que vous publiez dans CodeArtifact des référentiels restent privés et ne sont disponibles que pour les AWS comptes, les rôles et les utilisateurs auxquels vous avez accordé l'accès.

Publication de versions de packages corrigées

Parfois, vous souhaiterez peut-être publier une version de package modifiée, éventuellement disponible dans un dépôt public. Par exemple, vous avez peut-être découvert un bogue dans une dépendance d'application critique appelée `mydep` 1.1, et vous devez le corriger avant que le fournisseur du package ne puisse examiner et accepter la modification. Comme décrit

précédemment, vous CodeArtifact empêche de publier mydep 1.1 dans votre CodeArtifact référentiel si le référentiel public est accessible depuis votre CodeArtifact référentiel via des référentiels en amont et une connexion externe.

Pour contourner ce problème, publiez la version du package dans un autre CodeArtifact référentiel où le référentiel public n'est pas accessible. Utilisez ensuite l'`copy-package-versions` API pour copier la version corrigée de dans le CodeArtifact référentiel mydep 1.1 à partir duquel vous allez la consommer.

Limites de taille des ressources pour la publication

La taille maximale d'une ressource de package pouvant être publiée est limitée par le quota maximal de taille de fichier de ressource indiqué dans [Quotas dans AWS CodeArtifact](#). Par exemple, vous ne pouvez pas publier une roue Maven JAR ou Python dont la taille est supérieure au quota maximal de votre fichier de ressources actuel. Si vous devez y stocker des actifs plus importants CodeArtifact, demandez une augmentation de quota.

Outre le quota maximal de taille de fichier de ressource, la taille maximale d'une demande de publication pour les packages npm est de 2 Go. Cette limite est indépendante du quota maximal de taille du fichier de ressources et ne peut pas être augmentée avec une augmentation du quota. Dans une demande de publication npm (HTTP PUT), les métadonnées du package et le contenu de l'archive tar du package npm sont regroupés. De ce fait, la taille maximale réelle d'un package npm pouvant être publié varie et dépend de la taille des métadonnées incluses.

Note

Les packages npm publiés sont limités à une taille maximale inférieure à 2 Go.

Latence de publication

Les versions de package publiées dans un CodeArtifact référentiel peuvent souvent être téléchargées en moins d'une seconde. Par exemple, si vous publiez une version de package npm sur CodeArtifact withnpm `publish`, cette version devrait être disponible pour une npm `install` commande en moins d'une seconde. Cependant, la publication peut être incohérente et peut parfois prendre plus de temps. Si vous devez utiliser une version du package immédiatement après la publication, réessayez pour vous assurer que le téléchargement est fiable. Par exemple, après avoir publié la version du package, répétez le téléchargement jusqu'à trois fois si la version du package qui vient d'être publiée n'est pas initialement disponible lors de la première tentative de téléchargement.

Note

L'importation d'une version de package depuis un dépôt public prend généralement plus de temps que la publication. Pour de plus amples informations, veuillez consulter [Latence de connexion externe](#).

État de la version du package

Chaque version de package CodeArtifact possède un statut qui décrit l'état actuel et la disponibilité de la version du package. Vous pouvez modifier l'état de la version du package dans le SDK AWS CLI et. Pour de plus amples informations, veuillez consulter [État de version du package de mise à jour](#).

Les valeurs possibles pour l'état de la version du package sont les suivantes :

- Publié — La version du package a été publiée avec succès et peut être demandée à l'aide d'un gestionnaire de packages. La version du package sera incluse dans les listes de versions de package renvoyées aux gestionnaires de packages, par exemple dans la sortie de `npm view <package-name> versions`. Toutes les ressources de la version du package sont disponibles dans le référentiel.
- Inachevé : le client a chargé une ou plusieurs ressources pour une version de package, mais ne les a pas finalisées en les transférant dans l'Publishedétat. Actuellement, seules les versions génériques et les versions de package Maven peuvent avoir un statut de `Unfinished`. Pour les packages Maven, cela peut se produire lorsque le client télécharge une ou plusieurs ressources pour une version de package mais ne publie pas de `maven-metadata.xml` fichier pour le package qui inclut cette version. Lorsqu'une version de package Maven est inachevée, elle ne sera pas incluse dans les listes de versions renvoyées aux clients et `mvn` ne pourra donc pas être utilisée dans le cadre d'une compilation. gradle Les packages génériques peuvent être délibérément conservés dans `Unfinished` cet état en fournissant l'`unfinished` indicateur lors de l'appel de l'[PublishPackageVersionAPI](#). L'Publishedétat d'un package générique peut être changé en omettant le `unfinished` drapeau ou en appelant l'[UpdatePackageVersionsStatusAPI](#).
- Non répertorié — Les ressources de la version du package peuvent être téléchargées depuis le référentiel, mais la version du package n'est pas incluse dans la liste des versions renvoyées aux gestionnaires de packages. Par exemple, pour un package npm, la sortie de `npm view <package-name> versions` inclura pas la version du package. Cela signifie que la logique de résolution des dépendances de npm ne sélectionnera pas la version du package car celle-

ci n'apparaît pas dans la liste des versions disponibles. Toutefois, si la version du package non répertorié est déjà référencée dans un `npm package-lock.json` fichier, elle peut toujours être téléchargée et installée, par exemple lors de son exécution `npm ci`.

- Archivé — Les ressources de la version du package ne peuvent plus être téléchargées. La version du package ne sera pas incluse dans la liste des versions renvoyées aux gestionnaires de packages. Comme les actifs ne sont pas disponibles, la consommation de la version du package par les clients est bloquée. Si la version de votre application dépend d'une version mise à jour vers Archivé, la version sera interrompue, en supposant que la version du package n'a pas été mise en cache localement. [Vous ne pouvez pas utiliser un gestionnaire de packages ou un outil de génération pour republier une version de package archivée car elle est toujours présente dans le référentiel, mais vous pouvez redéfinir le statut de la version du package sur Non répertorié ou Publié avec l'UpdatePackageVersionsStatus API.](#)
- Supprimé — La version du package n'apparaît pas dans les listes et les ressources ne peuvent pas être téléchargées depuis le référentiel. La principale différence entre Disposé et Archivé est que lorsque le statut est Disposé, les actifs de la version du package seront définitivement supprimés par CodeArtifact. Pour cette raison, vous ne pouvez pas déplacer une version de package du statut Disposé vers Archivé, Non répertorié ou Publié. La version du package ne peut plus être utilisée car les actifs ont été supprimés. Une fois qu'une version du package a été marquée comme supprimée, le stockage des actifs du package ne vous sera plus facturé.

Les versions de package de tous les statuts seront renvoyées par défaut lors d' `list-package-versions` un appel sans `--status` paramètre.

Outre les états listés précédemment, une version de package peut également être supprimée avec l'[DeletePackageVersionsAPI](#). Une fois supprimée, une version de package ne figure plus dans le référentiel et vous pouvez librement republier cette version de package à l'aide d'un gestionnaire de packages ou d'un outil de génération. Après la suppression d'une version de package, le stockage des actifs de cette version de package ne vous sera plus facturé.

Nom du package, version du package et normalisation du nom des actifs

CodeArtifact normalise les noms des packages, les versions des packages et les noms des actifs avant de les stocker, ce qui signifie que les noms ou les versions CodeArtifact peuvent être différents du nom ou de la version fournis lors de la publication du package. Pour plus d'informations sur la façon dont les noms et les versions sont normalisés CodeArtifact pour chaque type de package, consultez la documentation suivante :

- [Normalisation du nom des paquets Python](#)
- [NuGet normalisation du nom du package, de la version et du nom de l'actif](#)

CodeArtifact n'effectue pas de normalisation sur les autres formats de package.

Lister les noms de packages

Utilisez la `list-packages` commande in CodeArtifact pour obtenir une liste de tous les noms de packages d'un référentiel. Cette commande renvoie uniquement les noms des packages, pas les versions.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --  
repository my_repo
```

Exemple de sortie :

```
{  
  "nextToken": "eyJidWNrZXRXJZCI6I...",  
  "packages": [  
    {  
      "package": "acorn",  
      "format": "npm",  
      "originConfiguration": {  
        "restrictions": {  
          "publish": "BLOCK",  
          "upstream": "ALLOW"  
        }  
      },  
      {  
        "package": "acorn-dynamic-import",  
        "format": "npm",  
        "originConfiguration": {  
          "restrictions": {  
            "publish": "BLOCK",  
            "upstream": "ALLOW"  
          }  
        },  
        {  
          "package": "ajv",  
          "format": "npm",  
          "originConfiguration": {  
            "restrictions": {  
              "publish": "BLOCK",  
              "upstream": "ALLOW"  
            }  
          }  
        }  
      }  
    }  
  ]  
}
```

```
        "restrictions": {
            "publish": "BLOCK",
            "upstream": "ALLOW"
        }
    },
    {
        "package": "ajv-keywords",
        "format": "npm",
        "originConfiguration": {
            "restrictions": {
                "publish": "BLOCK",
                "upstream": "ALLOW"
            }
        },
        {
            "package": "anymatch",
            "format": "npm",
            "originConfiguration": {
                "restrictions": {
                    "publish": "BLOCK",
                    "upstream": "ALLOW"
                }
            },
            {
                "package": "ast",
                "namespace": "webassemblyjs",
                "format": "npm",
                "originConfiguration": {
                    "restrictions": {
                        "publish": "BLOCK",
                        "upstream": "ALLOW"
                    }
                }
            }
        }
    ]
}
```

Répertorier les noms des packages npm

Pour répertorier uniquement les noms des packages npm, définissez la valeur de l'--formatoption sur. npm

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --  
repository my_repo \
```

```
--format npm
```

Pour répertorier les packages npm dans un espace de noms (npm scope), utilisez les `--namespace` options et `--format`.

Important

La valeur de l'`--namespace` option ne doit pas inclure le début@. Pour rechercher l'espace de noms@types, définissez la valeur sur `types`

Note

L'`--namespace` option filtre par préfixe d'espace de noms. Tout package npm dont la portée commence par la valeur transmise à l'`--namespace` option sera renvoyé dans la list-packages réponse.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format npm --namespace types
```

Exemple de sortie :

```
{
  "nextToken": "eyJidWNrZXRXJZ...",
  "packages": [
    {
      "package": "3d-bin-packing",
      "namespace": "types",
      "format": "npm"
    },
    {
      "package": "a-big-triangle",
      "namespace": "types",
      "format": "npm"
    },
    {
      "package": "a-big-triangle",
      "namespace": "types",
      "format": "npm"
    }
  ]
}
```

```
        "package": "a11y-dialog",
        "namespace": "types",
        "format": "npm"

    }
]
```

Répertorier les noms des packages Maven

Pour répertorier uniquement les noms des packages Maven, définissez la valeur de l'--formatoption sur maven. Vous devez également spécifier l'ID du groupe Maven dans l'--namespaceoption.

Note

L'--namespaceoption filtre par préfixe d'espace de noms. Tout package npm dont la portée commence par la valeur transmise à l'--namespaceoption sera renvoyé dans la list-packages réponse.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --
repository my_repo \
--format maven --namespace org.apache.commons
```

Exemple de sortie :

```
{
  "nextToken": "eyJidWNrZXRXJZ...",
  "packages": [
    {
      "package": "commons-lang3",
      "namespace": "org.apache.commons",
      "format": "maven"
    },
    {
      "package": "commons-collections4",
      "namespace": "org.apache.commons",
      "format": "maven"
    }
  ]
}
```

```
},
{
  "package": "commons-compress",
  "namespace": "org.apache.commons",
  "format": "maven"

}
]
}
```

Répertorier les noms des paquets Python

Pour répertorier uniquement les noms des packages Python, définissez la valeur de l'--formatoption surpypi.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --
repository my_repo \
--format pipi
```

Filtrer par préfixe de nom de package

Pour renvoyer des packages commençant par une chaîne spécifiée, vous pouvez utiliser l'--package-prefixoption.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --
repository my_repo \
--format npm --package-prefix pat
```

Exemple de sortie :

```
{
  "nextToken": "eyJidWNrZXRXJZ...",
  "packages": [
    {
      "package": "path",
      "format": "npm"

    },
    {
      "package": "pat-test",
      "format": "npm"
    }
  ]
}
```

```
  },
  {
    "package": "patch-math3",
    "format": "npm"
  }
]
```

Combinaisons d'options de recherche prises en charge

Vous pouvez utiliser les `--package-prefix` options `--format--namespace`, et dans n'importe quelle combinaison, sauf qu'`--namespace` elles ne peuvent pas être utilisées seules. La recherche de tous les packages npm dont la portée commence par `@types` nécessite que l'`--format` option soit spécifiée. L'utilisation `--namespace` en elle-même entraîne une erreur.

L'utilisation d'aucune des trois options est également prise en charge par `list-packages` et renverra tous les packages de tous les formats présents dans le référentiel.

Formater la sortie

Vous pouvez utiliser les paramètres disponibles pour toutes les AWS CLI commandes afin de rendre la `list-packages` réponse compacte et plus lisible. Utilisez le `--query` paramètre pour spécifier le format de chaque version de package renvoyée. Utilisez le `--output` paramètre pour formater la réponse en texte brut.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --  
repository my_repo \  
--output text --query 'packages[*].[package]'
```

Exemple de sortie :

```
accepts  
array-flatten  
body-parser  
bytes  
content-disposition  
content-type  
cookie  
cookie-signature
```

Pour plus d'informations, veuillez consulter [Contrôle de la sortie de commande à partir de la AWS CLI](#) dans le AWS Command Line Interface Guide de l'utilisateur.

Valeurs par défaut et autres options

Par défaut, le nombre maximum de résultats renvoyés par `list-packages` est de 100. Vous pouvez modifier cette limite de résultats à l'aide de l'`--max-results` option.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --  
repository my_repo --max-results 20
```

La valeur maximale autorisée `--max-results` est de 1 000. Pour permettre de répertorier les packages dans des référentiels contenant plus de 1 000 packages, `list-packages` prend en charge la pagination à l'aide du `nextToken` champ dans la réponse. Si le nombre de packages dans le référentiel est supérieur à la valeur de `--max-results`, vous pouvez transmettre la valeur de `nextToken` à un autre appel de `list-packages` pour obtenir la page de résultats suivante.

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --  
repository my_repo \  
--next-token r00ABXNyAEdjb...
```

Lister les versions des packages

Utilisez la `list-package-versions` commande in AWS CodeArtifact pour obtenir la liste de toutes les versions d'un nom de package dans un référentiel.

```
aws codeartifact list-package-versions --package kind-of \  
--domain my_domain --domain-owner 111122223333 \  
--repository my_repository --format npm
```

Exemple de sortie :

```
{  
  "defaultDisplayVersion": "1.0.1",  
  "format": "npm",  
  "package": "kind-of",  
  "versions": [  
    {  
      "version": "1.0.1",  
      "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",  
      "displayVersion": "1.0.1",  
      "isDefault": true  
    }  
  ]  
}
```

```
"status": "Published",
"origin": {
    "domainEntryPoint": {
        "externalConnectionName": "public:npmjs"
    },
    "originType": "EXTERNAL"
},
{
    "version": "1.0.0",
    "revision": "REVISION-SAMPLE-2-C752BEEF6D2CFC",
    "status": "Published",
    "origin": {
        "domainEntryPoint": {
            "externalConnectionName": "public:npmjs"
        },
        "originType": "EXTERNAL"
    }
},
{
    "version": "0.1.2",
    "revision": "REVISION-SAMPLE-3-654S65A5C5E1FC",
    "status": "Published",
    "origin": {
        "domainEntryPoint": {
            "externalConnectionName": "public:npmjs"
        },
        "originType": "EXTERNAL"
    }
},
{
    "version": "0.1.1",
    "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"",
    "status": "Published",
    "origin": {
        "domainEntryPoint": {
            "externalConnectionName": "public:npmjs"
        },
        "originType": "EXTERNAL"
    }
},
{
    "version": "0.1.0",
    "revision": "REVISION-SAMPLE-4-AF669139B772FC",
```

```
        "status": "Published",
        "origin": [
            {
                "domainEntryPoint": {
                    "externalConnectionName": "public:npmjs"
                },
                "originType": "EXTERNAL"
            }
        ]
    }
```

Vous pouvez ajouter le `--status` paramètre à l'`list-package-versions` appel pour filtrer les résultats en fonction de l'état de la version du package. Pour plus d'informations sur l'état de la version du package, consultez [État de la version du package](#).

Vous pouvez paginer la réponse à l'`list-package-versions` avec des `--next-token` paramètres `--max-results` et. Pour `--max-results`, spécifiez un entier compris entre 1 et 1 000 pour spécifier le nombre de résultats renvoyés sur une seule page. Sa valeur par défaut est 50. Pour renvoyer les pages suivantes, exécutez `list-package-versions` à nouveau et transmettez la `nextToken` valeur reçue dans la sortie de commande précédente à `--next-token`. Lorsque l'`--next-token` option n'est pas utilisée, la première page de résultats est toujours renvoyée.

La `list-package-versions` commande ne répertorie pas les versions des packages dans les référentiels en amont. Toutefois, les références aux versions de package d'un référentiel en amont qui ont été copiées dans votre référentiel lors d'une demande de version de package sont répertoriées. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

Répertorier les versions du package npm

Pour répertorier toutes les versions de package d'un package npm, définissez la valeur de l'`--format` option sur `npm`

```
aws codeartifact list-package-versions --package my_package --domain my_domain \
--domain-owner 111122223333 --repository my_repo --format npm
```

Pour répertorier les versions du package npm dans un espace de noms spécifique (npm scope), utilisez l'option `--namespace`. La valeur de l'`--namespace` option ne doit pas inclure le début @. Pour rechercher l'espace de noms@`types`, définissez la valeur sur `types`

```
aws codeartifact list-package-versions --package my_package --domain my_domain \  
--domain-owner 111122223333 --repository my_repo --format npm \  
--namespace types
```

Répertorier les versions du package Maven

Pour répertorier toutes les versions d'un package Maven, définissez la valeur de l'--formatoption sur `maven`. Vous devez également spécifier l'ID du groupe Maven dans l'--namespaceoption.

```
aws codeartifact list-package-versions --package my_package --domain my_domain \  
--domain-owner 111122223333 --repository my_repo --format maven \  
--namespace org.apache.commons
```

Trier les versions

`list-package-versions` peut générer des versions triées par ordre décroissant en fonction de l'heure de publication (les versions les plus récentes sont répertoriées en premier). Utilisez le --sort-by paramètre avec une valeur de `PUBLISHED_TIME`, comme suit.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333 \  
--repository my_repository \  
--format npm --package webpack --max-results 5 --sort-by PUBLISHED_TIME
```

Exemple de sortie :

```
{  
  
  "defaultDisplayVersion": "4.41.2",  
  "format": "npm",  
  "package": "webpack",  
  "versions": [  
    {  
      "version": "5.0.0-beta.7",  
      "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",  
      "status": "Published"  
    },  
    {  
      "version": "5.0.0-beta.6",  
      "revision": "REVISION-SAMPLE-2-C752BEEF6D2CFC",  
      "status": "Published"  
    },  
    {  
      "version": "5.0.0-beta.5",  
      "revision": "REVISION-SAMPLE-3-C752BEEF6D2CFC",  
      "status": "Published"  
    },  
    {  
      "version": "5.0.0-beta.4",  
      "revision": "REVISION-SAMPLE-4-C752BEEF6D2CFC",  
      "status": "Published"  
    },  
    {  
      "version": "5.0.0-beta.3",  
      "revision": "REVISION-SAMPLE-5-C752BEEF6D2CFC",  
      "status": "Published"  
    }  
  ]  
}
```

```
{  
  "version": "5.0.0-beta.5",  
  "revision": "REVISION-SAMPLE-3-654S65A5C5E1FC",  
  "status": "Published"  
},  
{  
  "version": "5.0.0-beta.4",  
  "revision": "REVISION-SAMPLE-4-AF669139B772FC",  
  "status": "Published"  
},  
{  
  "version": "5.0.0-beta.3",  
  "revision": "REVISION-SAMPLE-5-C752BEE9B772FC",  
  "status": "Published"  
}  
,  
"nextToken": "eyJsaXN0UGF...."  
}
```

Version d'affichage par défaut

La valeur renvoyée pour `defaultDisplayVersion` dépend du format du package :

- Pour les packages génériques, Maven et PyPI, il s'agit de la version de package la plus récente publiée.
- Pour les packages npm, il s'agit de la version référencée par le `latest` tag. Si la `latest` balise n'est pas définie, il s'agit de la dernière version du package publiée.

Formater la sortie

Vous pouvez utiliser les paramètres disponibles pour toutes les AWS CLI commandes afin de rendre la `list-package-versions` réponse compacte et plus lisible. Utilisez le `--query` paramètre pour spécifier le format de chaque version de package renvoyée. Utilisez le `--output` paramètre pour formater la réponse sous forme de texte brut.

```
aws codeartifact list-package-versions --package my-package-name --domain my_domain --  
domain-owner 111122223333 \  
--repository my_repo --format npm --output text --query 'versions[*].[version]'
```

Exemple de sortie :

0.1.1
0.1.2
0.1.0
3.0.0

Pour plus d'informations, consultez la section [Contrôle de la sortie des commandes AWS CLI dans le guide de AWS Command Line Interface l'utilisateur](#).

Répertorier les actifs de la version

Un actif est un fichier individuel (par exemple, un fichier npm ou un .tgz fichier Maven POM ou JAR) stocké dans CodeArtifact lequel est associé à une version de package. Vous pouvez utiliser la `list-package-version-assets` commande pour répertorier les actifs de chaque version de package.

Exécutez la `list-package-version-assets` commande pour renvoyer les informations suivantes à propos de chaque actif de votre AWS compte et de votre AWS région actuelle :

- Son nom
- Sa taille, en octets.
- Ensemble de valeurs de hachage utilisées pour la validation de la somme de contrôle.

Par exemple, utilisez la commande suivante pour répertorier les actifs du package `Pythonflatten-json`, version `0.1.7`.

```
aws codeartifact list-package-version-assets --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format pypi --package flatten-json \
--package-version 0.1.7
```

Le résultat est présenté ci-dessous :

```
{  
  "format": "pypi",  
  "package": "flatten-json",  
  "version": "0.1.7",  
  "versionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",  
  "assets": [  
    {  
      "name": "flatten_json-0.1.7-py3-none-any.whl",
```

```
        "size": 31520,
        "hashes": {
            "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",
            "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",
            "SHA-256": "43f24850b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
            "SHA-512": "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fc
            ed7bd1e80a0dca9ce320d95f086
            SHA-512"
        },
        {
            "name": "flatten_json-0.1.7.tar.gz",
            "size": 2865,
            "hashes": {
                "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",
                "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",
                "SHA-256": "43f24850b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
                "SHA-512": "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fc
                ed7bd1e80a0dca9ce320d95f086
                SHA-512"
            }
        }
    ]
}
```

Lister les actifs d'un package npm

Un package npm possède toujours un seul actif portant le nom de package.tgz. Pour répertorier les actifs d'un package npm délimité, incluez la portée dans l'--namespaceoption.

```
aws codeartifact list-package-version-assets --domain my_domain --domain-
owner 111122223333 \
--repository my_repo --format npm --package webpack \
--namespace types --package-version 4.9.2
```

Lister les actifs d'un package Maven

Pour répertorier les actifs d'un package Maven, incluez l'espace de noms du package dans l'--namespaceoption. Pour répertorier les actifs du package Maven : commons-cli:commons-cli

```
aws codeartifact list-package-version-assets --domain my_domain --domain-owner 111122223333 \  
--repository my_repo --format maven --package commons-cli \  
--namespace commons-cli --package-version 1.0
```

Télécharger les ressources de la version du package

Un actif est un fichier individuel (par exemple, un fichier npm ou un .tgz fichier Maven POM ou JAR) stocké dans CodeArtifact lequel est associé à une version de package. Vous pouvez télécharger les ressources du package à l'aide de `get-package-version-assets` command. Cela vous permet de récupérer des actifs sans utiliser un client de gestionnaire de packages tel que npm ou pip. Pour télécharger une ressource, vous devez fournir le nom de la ressource, qui peut être obtenu à l'aide de la `list-package-version-assets` commande. Pour plus d'informations, voir [Répertorier les actifs de la version](#). La ressource sera téléchargée sur le stockage local avec un nom de fichier que vous spécifiez.

L'exemple suivant télécharge la *guava-27.1-jre.jar* ressource depuis le package Maven *com.google.guava:guava* avec la version *27.1-jre*.

```
aws codeartifact get-package-version-asset --domain my_domain --domain-owner 111122223333 --repository my_repo \  
--format maven --namespace com.google.guava --package guava --package-version 27.1-jre \  
--asset guava-27.1-jre.jar \  
guava-27.1-jre.jar
```

Dans cet exemple, le nom du fichier a été spécifié *guava-27.1-jre.jar* par le dernier argument de la commande précédente, de sorte que la ressource téléchargée sera nommée *guava-27.1-jre.jar*.

Le résultat de la commande sera :

```
{  
  "assetName": "guava-27.1-jre.jar",  
  "packageVersion": "27.1-jre",  
  "packageVersionRevision": "YGp9ck2tmy03PGSxioclfYzQ0BfTLR9zzhQJtERv62I="  
}
```

Note

Pour télécharger des ressources à partir d'un package npm délimité, incluez la portée dans `--namespace` option. Le `@` symbole doit être omis lors de l'utilisation `--namespace`. Par exemple, si le champ d'application est `@types`, utilisez `--namespace types`.

Le téléchargement de ressources à l'aide de la ressource du package `get-package-version-asset` nécessite une `codeartifact:GetPackageVersionAsset` autorisation sur la ressource du package. Pour plus d'informations sur les politiques d'autorisation basées sur les ressources, consultez la section [Politiques basées sur les ressources dans le Guide de l'utilisateur](#). AWS Identity and Access Management

Copier des packages entre des référentiels

Vous pouvez copier les versions des packages d'un référentiel à un autre dans CodeArtifact. Cela peut être utile pour des scénarios tels que les flux de travail de promotion de packages ou le partage de versions de packages entre des équipes ou des projets. Les référentiels source et de destination doivent se trouver dans le même domaine pour copier les versions des packages.

Autorisations IAM requises pour copier des packages

Pour copier des versions de package CodeArtifact, l'utilisateur appelant doit disposer des autorisations IAM requises et la politique basée sur les ressources attachée aux référentiels source et de destination doit disposer des autorisations requises. Pour plus d'informations sur les politiques d'autorisation et les CodeArtifact référentiels basés sur les ressources, consultez [Politiques de référentiel](#).

L'utilisateur appelant `copy-package-versions` doit avoir l'`ReadFromRepository` autorisation sur le référentiel source et l'`CopyPackageVersions` autorisation sur le référentiel de destination.

Le référentiel source doit disposer de l'`ReadFromRepository` autorisation et le référentiel de destination doit disposer de l'`CopyPackageVersions` autorisation attribuée au compte IAM ou à l'utilisateur qui copie des packages. Les politiques suivantes sont des exemples de politiques de référentiel à ajouter au référentiel source ou au référentiel de destination à l'aide de la `put-repository-permissions-policy` commande. Remplacez `111122223333` par l'identifiant du compte appelant `copy-package-versions`.

Note

L'appel `put-repository-permissions-policy` remplacera la politique de dépôt actuelle, s'il en existe une. Vous pouvez utiliser la `get-repository-permissions-policy` commande pour voir s'il existe une politique. Pour plus d'informations, consultez [Lire une politique](#). Si une politique existe, vous souhaiterez peut-être y ajouter ces autorisations au lieu de la remplacer.

Exemple de politique d'autorisation du référentiel source

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:ReadFromRepository"  
      ],  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333:root"  
      },  
      "Resource": "*"  
    }  
  ]  
}
```

Exemple de politique d'autorisation du référentiel de destination

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:CopyPackageVersions"  
      ]  
    }  
  ]  
}
```

```
        ],
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Resource": "*"
    }
]
}
```

Copier les versions du package

Utilisez la `copy-package-versions` commande in CodeArtifact pour copier une ou plusieurs versions de package d'un référentiel source vers un référentiel de destination dans le même domaine. L'exemple suivant copiera les versions 6.0.2 et 4.0.0 d'un package npm nommé `my-package` du `my_repo` référentiel vers le référentiel `repo-2`

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333
--source-repository my_repo \
--destination-repository repo-2 --package my-package --format npm \
--versions 6.0.2 4.0.0
```

Vous pouvez copier plusieurs versions du même nom de package en une seule opération. Pour copier des versions de différents noms de packages, vous devez appeler chacun `copy-package-versions` d'eux.

La commande précédente produira le résultat suivant, en supposant que les deux versions puissent être copiées avec succès.

```
{
    "successfulVersions": {
        "6.0.2": {
            "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        "4.0.0": {
            "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        }
    },
    "failedVersions": {}
}
```

}

Copier un package depuis des référentiels en amont

Normalement, `copy-package-versions` ne recherche dans le référentiel spécifié par l'`--source-repository` option que les versions à copier. Toutefois, vous pouvez copier des versions à la fois du référentiel source et de ses référentiels en amont en utilisant l'`--include-from-upstream` option. Si vous utilisez le CodeArtifact SDK, appelez l'`CopyPackageVersionsAPI` avec le `includeFromUpstream` paramètre défini sur `true`. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

Copier un package npm délimité

Pour copier une version de package npm dans une portée, utilisez l'`--namespace` option pour spécifier la portée. Par exemple, pour copier le package`@types/react`, utilisez`--namespace types`. Le `@` symbole doit être omis lors de l'utilisation`--namespace`.

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
--source-repository repo-1 \
--destination-repository repo-2 --format npm --namespace types \
--package react --versions 0.12.2
```

Copier les versions du package Maven

Pour copier des versions de package Maven entre des référentiels, spécifiez le package à copier en transmettant l'ID du groupe Maven avec l'`--namespace` option et l'ArtifactID Maven avec l'option. `--name` Par exemple, pour copier une seule version de `com.google.guava:guava` :

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
--source-repository my_repo --destination-repository repo-2 --format maven -- \
namespace com.google.guava \
--package guava --versions 27.1-jre
```

Si la version du package est copiée avec succès, le résultat sera similaire à ce qui suit.

```
{  
  "successfulVersions": {  
    "27.1-jre": {  
      "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",  
      "version": "27.1-jre"  
    }  
  }  
}
```

```
        "status": "Published"
    }
},
"failedVersions": {}
}
```

Versions qui n'existent pas dans le référentiel source

Si vous spécifiez une version qui n'existe pas dans le référentiel source, la copie échouera. Si certaines versions existent dans le référentiel source et d'autres non, toutes les versions ne seront pas copiées. Dans l'exemple suivant, la version 0.2.0 du package `array-unique` npm est présente dans le référentiel source, mais pas la version 5.6.7 :

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
    --source-repository my_repo --destination-repository repo-2 --format npm \
    --package array-unique --versions 0.2.0 5.6.7
```

Dans ce scénario, le résultat sera similaire au suivant.

```
{
    "successfulVersions": {},
    "failedVersions": {
        "0.2.0": {
            "errorCode": "SKIPPED",
            "errorMessage": "Version 0.2.0 was skipped"
        },
        "5.6.7": {
            "errorCode": "NOT_FOUND",
            "errorMessage": "Could not find version 5.6.7"
        }
    }
}
```

Le code `SKIPPED` d'erreur est utilisé pour indiquer que la version n'a pas été copiée dans le référentiel de destination car aucune autre version n'a pu être copiée.

Versions qui existent déjà dans le référentiel de destination

Lorsqu'une version de package est copiée dans un référentiel où elle existe déjà, CodeArtifact compare ses actifs de package et les métadonnées au niveau de la version du package dans les deux référentiels.

Si les actifs et les métadonnées de la version du package sont identiques dans les référentiels source et de destination, aucune copie n'est effectuée mais l'opération est considérée comme réussie. Cela signifie que `copy-package-versions` est idempotent. Dans ce cas, la version qui était déjà présente dans les référentiels source et de destination ne sera pas répertoriée dans la sortie `decopy-package-versions`.

Dans l'exemple suivant, deux versions du package npm `array-unique` sont présentes dans le référentiel `repo-1` source. La version `0.2.1` est également présente dans le référentiel de destination `dest-repo` et la version `0.2.0` ne l'est pas.

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \  
    --source-repository my_repo --destination-repository repo-2 --format npm --  
    package array-unique \  
    --versions 0.2.1 0.2.0
```

Dans ce scénario, le résultat sera similaire au suivant.

```
{  
  "successfulVersions": {  
    "0.2.0": {  
      "revision": "Yad+B1QcBq2kdEVrx1E1vSfHJVh8Pr61hBUkoWPGWX0=",  
      "status": "Published"  
    }  
  },  
  "failedVersions": {}  
}
```

La version `0.2.0` est répertoriée `successfulVersions` car elle a été copiée avec succès du référentiel source vers le référentiel de destination. La version `0.2.1` n'apparaît pas dans la sortie car elle était déjà présente dans le référentiel de destination.

Si la version du package, les ressources ou les métadonnées diffèrent dans les référentiels source et de destination, l'opération de copie échouera. Vous pouvez utiliser le `--allow-overwrite` paramètre pour forcer un remplacement.

Si certaines versions existent dans le référentiel de destination et d'autres non, toutes les versions ne seront pas copiées. Dans l'exemple suivant, la version `0.3.2` du package `array-unique` npm est présente à la fois dans les référentiels source et de destination, mais le contenu de la version du package est différent. La version `0.2.1` est présente dans le référentiel source mais pas dans le référentiel de destination.

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \  
    --source-repository my_repo --destination-repository repo-2 --format npm --  
    package array-unique \  
    --versions 0.3.2 0.2.1
```

Dans ce scénario, le résultat sera similaire au suivant.

```
{  
    "successfulVersions": {},  
    "failedVersions": {  
        "0.2.1": {  
            "errorCode": "SKIPPED",  
            "errorMessage": "Version 0.2.1 was skipped"  
        },  
        "0.3.2": {  
            "errorCode": "ALREADY_EXISTS",  
            "errorMessage": "Version 0.3.2 already exists"  
        }  
    }  
}
```

La version 0.2.1 est marquée comme SKIPPED car elle n'a pas été copiée dans le référentiel de destination. Il n'a pas été copié car la copie de la version 0.3.2 a échoué car elle était déjà présente dans le référentiel de destination, mais pas identique dans les référentiels source et de destination.

Spécification d'une révision de version de package

Une révision de version de package est une chaîne qui spécifie un ensemble spécifique de ressources et de métadonnées pour une version de package. Vous pouvez spécifier une révision de version de package pour copier les versions de package dans un état spécifique. Pour spécifier une révision de version de package, utilisez le `--version-revisions` paramètre pour transmettre une ou plusieurs versions de package séparées par des virgules et les paires de révisions de version de package à la `copy-package-versions` commande.

Note

Vous devez spécifier le `--versions` ou le `--version-revisions` paramètre avec `copy-package-versions`. Vous ne pouvez pas spécifier les deux.

L'exemple suivant ne copiera la version 0.3.2 du package que *my-package* si elle est présente dans le référentiel source avec la révision REVISION-1-SAMPLE-6C81EFF7DA55CC de la version du package.

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
--source-repository repo-1 \
--destination-repository repo-2 --format npm --namespace my-namespace \
--package my-package --version-revisions 0.3.2=REVISION-1-SAMPLE-6C81EFF7DA55CC
```

L'exemple suivant copie deux versions du package *my-package*, 0.3.2 et 0.3.13. La copie ne réussira que si, dans le référentiel source, la version 0.3.2 de *my-package* contient une révision REVISION-1-SAMPLE-6C81EFF7DA55CC et la version 0.3.13 une révision REVISION-2-SAMPLE-55C752BEE772FC

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
--source-repository repo-1 \
--destination-repository repo-2 --format npm --namespace my-namespace \
--package my-package --version-revisions 0.3.2=REVISION-1-SAMPLE-6C81EFF7DA55CC,0.3.13=REVISION-2-SAMPLE-55C752BEE772FC
```

Pour rechercher les révisions d'une version de package, utilisez la `list-package-versions` commande `describe-package-version` ou.

Pour plus d'informations, veuillez consulter les sections [Révision de la version du package](#) et [CopyPackageVersion](#) (français non garanti) de la Référence d'API CodeArtifact .

Copier les packages npm

Pour plus d'informations sur `copy-package-versions` le comportement avec les packages npm, consultez les [balises npm et l'CopyPackageVersionsAPI](#).

Supprimer un package ou une version de package

Vous pouvez supprimer une ou plusieurs versions de package à la fois à l'aide de la `delete-package-versions` commande. Pour supprimer complètement un package d'un référentiel, y compris toutes les versions et configurations associées, utilisez la `delete-package` commande. Un package peut exister dans un référentiel sans aucune version de package. Cela peut se produire lorsque toutes les versions sont supprimées à l'aide de la `delete-package-versions` commande,

ou si le package a été créé sans aucune version à l'aide de l'opération `put-package-origin-configuration` API (voir [Modification des contrôles d'origine des packages](#)).

Rubriques

- [Suppression d'un package \(AWS CLI\)](#)
- [Suppression d'un package \(console\)](#)
- [Supprimer une version de package \(AWS CLI\)](#)
- [Suppression d'une version de package \(console\)](#)
- [Supprimer un package npm ou une version de package](#)
- [Supprimer un package Maven ou une version de package](#)
- [Bonnes pratiques pour supprimer des packages ou des versions de packages](#)

Suppression d'un package (AWS CLI)

Vous pouvez supprimer un package, y compris toutes ses versions et configurations, à l'aide de la `delete-package` commande. L'exemple suivant supprime le package PyPI `my-package` nommé dans le `my_repo` dépôt du domaine : `my_domain`

```
aws codeartifact delete-package --domain my_domain --domain-owner 111122223333 \  
--repository my_repo --format pypi \  
--package my-package
```

Exemple de sortie :

```
{  
  "deletedPackage": {  
    "format": "pypi",  
    "originConfiguration": {  
      "restrictions": {  
        "publish": "ALLOW",  
        "upstream": "BLOCK"  
      }  
    },  
    "package": "my-package"  
  }  
}
```

Vous pouvez confirmer que le package a été supprimé en exécutant `describe-package` le même nom de package :

```
aws codeartifact describe-package --domain my_domain --domain-owner 111122223333 \  
--repository my_repo --format pypi --package my-package
```

Suppression d'un package (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le panneau de navigation, choisissez Référentiels.
3. Choisissez le référentiel dans lequel vous souhaitez supprimer un package.
4. Choisissez le Package que vous souhaitez supprimer.
5. Choisissez Supprimer le package.

Supprimer une version de package (AWS CLI)

Vous pouvez supprimer une ou plusieurs versions de package à la fois à l'aide de la `delete-package-versions` commande. L'exemple suivant supprime les versions *4.0.0*, *4.0.1*, et *5.0.0* du package PyPI *my-package* nommé dans *my_repo* le domaine : *my_domain*

```
aws codeartifact delete-package-versions --domain my_domain --domain-owner 111122223333 \  
--repository my_repo --format pypi \  
--package my-package --versions 4.0.0 4.0.1 5.0.0
```

Exemple de sortie :

```
{  
  "successfulVersions": {  
    "4.0.0": {  
      "revision": "oxwwYC9dDeuBoCt6+PDSwL60MZ7rXeiXy44BM32Iawo=",  
      "status": "Deleted"  
    },  
    "4.0.1": {  
      "revision": "byaaQR748wrsdBaT+PDSwL60MZ7rXeiBKM0551aqWmo=",  
      "status": "Deleted"  
    },  
    "5.0.0": {  
      "revision": "byaaQR748wrsdBaT+PDSwL60MZ7rXeiBKM0551aqWmo=",  
      "status": "Deleted"  
    }  
  }  
}
```

```
        "revision": "yubm34QWeST345ts+ASeioPI354rXeiSWr734PotwRw=",
        "status": "Deleted"
    }
},
"failedVersions": {}
}
```

Vous pouvez confirmer que les versions ont été supprimées en exécutant `list-package-versions` le même nom de package :

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format pypi --package my-package
```

Suppression d'une version de package (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le panneau de navigation, choisissez Référentiels.
3. Choisissez le référentiel dans lequel vous souhaitez supprimer les versions du package.
4. Choisissez le Package dont vous souhaitez supprimer les versions.
5. Sélectionnez la version du package que vous souhaitez supprimer.
6. Sélectionnez Delete (Supprimer).

 Note

Dans la console, vous ne pouvez supprimer qu'une seule version de package à la fois. Pour en supprimer plusieurs à la fois, utilisez la CLI.

Supprimer un package npm ou une version de package

Pour supprimer un package npm ou des versions de package individuelles, définissez l'--formatoption sur `npm`. Pour supprimer une version de package dans un package npm délimité, utilisez l'--namespaceoption pour spécifier la portée. Par exemple, pour supprimer le package`@types/react`, utilisez`--namespace types`. Omettez le `@` symbole lors de l'utilisation`--namespace`.

```
aws codeartifact delete-package-versions --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format npm --namespace types \
--package react --versions 0.12.2
```

Pour supprimer le package@types/react, y compris toutes ses versions :

```
aws codeartifact delete-package --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format npm --namespace types \
--package react
```

Supprimer un package Maven ou une version de package

Pour supprimer un package Maven ou des versions de package individuelles, définissez l'--formatoption sur maven et spécifiez le package à supprimer en transmettant l'ID du groupe Maven avec l'--namespaceoption et le Maven ArtifactID avec l'option. --name Par exemple, voici comment supprimer une seule version de com.google.guava :

```
aws codeartifact delete-package-versions --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format maven --namespace com.google.guava \
--package guava --versions 27.1-jre
```

L'exemple suivant montre comment supprimer le package com.google.guava, y compris toutes ses versions :

```
aws codeartifact delete-package --domain my_domain --domain-owner 111122223333 \
--repository my_repo --format maven --namespace com.google.guava \
--package guava
```

Bonnes pratiques pour supprimer des packages ou des versions de packages

Si vous devez supprimer une version du package, il est recommandé de créer un référentiel pour stocker une copie de sauvegarde de la version du package que vous souhaitez supprimer. Vous pouvez le faire en appelant d'abord copy-package-versions le dépôt de sauvegarde :

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333 \
--source-repository my_repo \
```

```
--destination-repository repo-2 --package my-package --format npm \
--versions 6.0.2 4.0.0
```

Une fois que vous avez copié la version du package, vous pouvez faire appel `delete-package-versions` au package ou à la version du package que vous souhaitez supprimer.

```
aws codeartifact delete-package-versions --domain my_domain --domain-owner 111122223333
 \
--repository my_repo --format pypi \
--package my-package --versions 4.0.0 4.0.1 5.0.0
```

Afficher et mettre à jour les détails et les dépendances des versions du package

Vous pouvez consulter les informations relatives à la version d'un package, y compris les dépendances, dans CodeArtifact. Vous pouvez également mettre à jour le statut d'une version de package. Pour plus d'informations sur l'état de la version du package, consultez [État de la version du package](#).

Afficher les détails de la version du package

Utilisez la `describe-package-version` commande pour afficher les détails relatifs aux versions des packages. Les détails de la version du package sont extraits d'un package lorsqu'il est publié sur CodeArtifact. Les détails des différents packages varient et dépendent de leurs formats et de la quantité d'informations que leurs auteurs y ont ajoutées.

La plupart des informations contenues dans le résultat de la `describe-package-version` commande dépendent du format du package. Par exemple, `describe-package-version` extrait les informations d'un package npm de son `package.json` fichier. La révision est créée par CodeArtifact. Pour de plus amples informations, veuillez consulter [Spécification d'une révision de version de package](#).

Deux versions de package portant le même nom peuvent se trouver dans le même référentiel si elles se trouvent chacune dans des espaces de noms différents. Utilisez le `--namespace` paramètre facultatif pour spécifier un espace de noms. Pour plus d'informations, consultez [Afficher les détails de la version du package npm](#) ou [Afficher les détails de la version du package Maven](#).

L'exemple suivant renvoie des détails sur la version `1.9.0` d'un package Python nommé `pyhamcrest` qui se trouve dans le `my_repo` référentiel.

```
aws codeartifact describe-package-version --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format pypi --package pyhamcrest --package-version 1.9.0
```

Le résultat peut ressembler à ce qui suit.

```
{  
  "format": "pypi",  
  "package": "PyHamcrest",  
  "displayName": "PyHamcrest",  
  "version": "1.9.0",  
  "summary": "Hamcrest framework for matcher objects",  
  "homePage": "https://github.com/hamcrest/PyHamcrest",  
  "publishedTime": 1566002944.273,  
  "licenses": [  
    {  
      "id": "license-id",  
      "name": "license-name"  
    }  
  ],  
  "revision": "REVISION-SAMPLE-55C752BEE9B772FC"  
}
```

Note

CodeArtifact récupère les détails de la version du package, tels que la page d'accueil du package ou les informations de licence du package, à partir des métadonnées fournies par l'auteur du package. Si l'une de ces informations dépasse 400 Ko, ce qui correspond à la taille limite des éléments DynamoDB CodeArtifact, vous ne pourrez pas traiter ces données et il est possible que vous ne voyiez pas ces informations sur la console ou dans la réponse de `describe-package-version`. Par exemple, un package python tel que <https://pypi.org/project/rapyd-sdk/> possède un très grand champ de licence, ces informations ne seront donc pas traitées par CodeArtifact.

Afficher les détails de la version du package npm

Pour afficher les détails d'une version de package npm, définissez la valeur de l'`--format` option sur **npm**. Vous pouvez éventuellement inclure l'espace de noms de version du package (npm scope).

dans l'--namespaceoption. La valeur de l'--namespaceoption ne doit pas inclure le début@. Pour rechercher l'espace de noms@types, définissez la valeur sur. **types**

Ce qui suit renvoie des détails sur 4.41.5 la version d'un package npm nommé webpack dans le @types scope.

```
aws codeartifact describe-package-version --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format npm --package webpack --namespace types --package-version 4.41.5
```

Le résultat peut ressembler à ce qui suit.

```
{
  "format": "npm",
  "namespace": "types",
  "package": "webpack",
  "displayName": "webpack",
  "version": "4.41.5",
  "summary": "Packs CommonJS/AMD modules for the browser. Allows ... further output omitted for brevity",
  "homePage": "https://github.com/webpack/webpack",
  "sourceCodeRepository": "https://github.com/webpack/webpack.git",
  "publishedTime": 1577481261.09,
  "licenses": [
    {
      "id": "license-id",
      "name": "license-name"
    }
  ],
  "revision": "REVISION-SAMPLE-55C752BEE9B772FC",
  "status": "Published",
  "origin": {
    "domainEntryPoint": {
      "externalConnectionName": "public:npmjs"
    },
    "originType": "EXTERNAL"
  }
}
```

Afficher les détails de la version du package Maven

Pour afficher les détails d'une version de package Maven, définissez la valeur de l'--formatoption sur maven et incluez l'espace de noms de version du package dans l'--namespaceoption.

L'exemple suivant renvoie des détails sur 1.2 la version d'un package Maven nommé commons-rng-client-api qui se trouve dans l'espace de org.apache.commons noms et dans le my_repo référentiel.

```
aws codeartifact describe-package-version --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format maven --namespace org.apache.commons --package commons-rng-client-api --
package-version 1.2
```

Le résultat peut ressembler à ce qui suit.

```
{  
  "format": "maven",  
  "namespace": "org.apache.commons",  
  "package": "commons-rng-client-api",  
  "displayName": "Apache Commons RNG Client API",  
  "version": "1.2",  
  "summary": "API for client code that uses random numbers generators.",  
  "publishedTime": 1567920624.849,  
  "licenses": [],  
  "revision": "REVISION-SAMPLE-55C752BEE9B772FC"  
}
```

Note

CodeArtifact n'extrait pas les informations détaillées sur la version du package des fichiers POM parents. Les métadonnées d'une version de package donnée incluront uniquement des informations dans le POM pour cette version de package exacte, et non pour le POM parent ou tout autre POM référencé de manière transitive à l'aide de la parent balise POM. Cela signifie que la sortie de describe-package-version omettra les métadonnées (telles que les informations de licence) pour les versions de package Maven qui reposent sur une parent référence pour contenir ces métadonnées.

Afficher les dépendances entre les versions du package

Utilisez la `list-package-version-dependencies` commande pour obtenir la liste des dépendances d'une version de package. La commande suivante répertorie les dépendances d'un package npm nommé `my-package`, version `4.41.5`, dans le `my_repo` référentiel, dans le `my_domain` domaine.

```
aws codeartifact list-package-version-dependencies --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format npm --package my-package --package-version 4.41.5
```

Le résultat peut ressembler à ce qui suit.

```
{
  "dependencies": [
    {
      "namespace": "webassemblyjs",
      "package": "ast",
      "dependencyType": "regular",
      "versionRequirement": "1.8.5"
    },
    {
      "namespace": "webassemblyjs",
      "package": "helper-module-context",
      "dependencyType": "regular",
      "versionRequirement": "1.8.5"
    },
    {
      "namespace": "webassemblyjs",
      "package": "wasm-edit",
      "dependencyType": "regular",
      "versionRequirement": "1.8.5"
    }
  ],
  "versionRevision": "REVISION-SAMPLE-55C752BEE9B772FC"
}
```

Pour connaître la plage de valeurs prises en charge pour le champ `DependencyType`, consultez le type de [PackageDependency](#) données dans l'CodeArtifact API.

Afficher le fichier readme de la version du package

Certains formats de package, tels que npm, incluent un README fichier. Utilisez le `get-package-version-readme` pour obtenir le README fichier d'une version de package. La commande suivante renvoie le README fichier d'un package npm nommé `my-package`, version `4.41.5`, dans le `my_repo` référentiel, dans le `my_domain` domaine.

Note

CodeArtifact ne prend pas en charge l'affichage de fichiers readme à partir de packages génériques ou Maven.

```
aws codeartifact get-package-version-readme --domain my_domain --domain-owner 111122223333 --repository my_repo \
--format npm --package my-package --package-version 4.41.5
```

Le résultat peut ressembler à ce qui suit.

```
{  
  "format": "npm",  
  "package": "my-package",  
  "version": "4.41.5",  
  "readme": "<div align="center"><a href="https://github.com/webpack/webpack"> ... more content ...</a>",  
  "versionRevision": "REVISION-SAMPLE-55C752BEE9B772FC"  
}
```

État de version du package de mise à jour

Chaque version de package CodeArtifact possède un statut qui décrit l'état actuel et la disponibilité de la version du package. Vous pouvez modifier l'état de la version du package à l'aide de la console AWS CLI et de la console.

Note

Pour plus d'informations sur l'état de la version du package, y compris une liste des statuts disponibles, consultez [État de la version du package](#).

Mettre à jour le statut de version du package

La définition du statut d'une version de package permet de contrôler la manière dont une version de package peut être utilisée sans la supprimer complètement du référentiel. Par exemple, lorsqu'une version de package a le statut `Unlisted`, elle peut toujours être téléchargée normalement, mais elle n'apparaîtra pas dans les listes de versions de package renvoyées à des commandes telles qu'en `npm view`. L'[UpdatePackageVersionsStatus API](#) permet de définir l'état de version de plusieurs versions d'un même package en un seul appel d'API. Pour une description des différents statuts, voir [Vue d'ensemble des packages](#).

Utilisez la `update-package-versions-status` commande pour modifier le statut d'une version de package en `Published`, `Unlisted`, ou `Archived`. Pour connaître les autorisations IAM requises pour utiliser la commande, consultez [Autorisations IAM requises pour mettre à jour l'état de la version d'un package](#). L'exemple suivant définit le statut de la version 4.1.0 du package `chalk` npm sur.

`Archived`

```
aws codeartifact update-package-versions-status --domain my_domain
  --domain-owner 111122223333 --repository my_repo --format npm --package chalk
  --versions 4.1.0 --target-status Archived
```

Exemple de sortie :

```
{ "successfulVersions": {
    "4.1.0": {
        "revision": "+0z8skWbwY3k8M6SrNIqNj6bVH/ax+CxvkJx+No5j8I=",
        "status": "Archived"
    }
},
"failedVersions": {}}
```

Cet exemple utilise un package npm, mais la commande fonctionne de la même manière pour les autres formats. Plusieurs versions peuvent être déplacées vers le même statut cible à l'aide d'une seule commande, voir l'exemple suivant.

```
aws codeartifact update-package-versions-status --domain my_domain
  --domain-owner 111122223333 --repository my_repo --format npm --package chalk
  --versions 4.1.0 4.1.1 --target-status Archived
```

Exemple de sortie :

```
{  
  "successfulVersions": {  
    "4.1.0": {  
      "revision": "25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ4=",  
      "status": "Archived"  
    },  
    "4.1.1": {  
      "revision": "+0z8skWbwY3k8M6SrNIqNj6bVH/ax+CxvkJx+No5j8I=",  
      "status": "Archived"  
    }  
  "failedVersions": {}  
}
```

Notez qu'une fois publiée, une version de package ne peut pas être replacée dans cet Unfinished état. Ce statut n'est donc pas autorisé en tant que valeur pour le --target-status paramètre. Pour déplacer la version du package vers l'Disposedétat, utilisez plutôt la dispose-package-versions commande comme décrit ci-dessous.

Autorisations IAM requises pour mettre à jour l'état de la version d'un package

update-package-versions-status Pour demander un package, vous devez avoir l'codeartifact:UpdatePackageVersionsStatus autorisation d'accéder à la ressource du package. Cela signifie que vous pouvez accorder l'autorisation update-package-versions-status d'appeler pour chaque forfait. Par exemple, une politique IAM qui accorde l'autorisation d'appeler update-package-versions-status le package npm *chalk* inclurait une déclaration comme celle-ci.

```
{  
  "Action": [  
    "codeartifact:UpdatePackageVersionsStatus"  
  ],  
  "Effect": "Allow",  

```

Mise à jour de l'état d'un package npm délimité

Pour mettre à jour l'état de la version du package d'une version de package npm avec une portée, utilisez le `--namespace` paramètre. Par exemple, pour annuler la liste de la version 8.0.0 de `@nestjs/core`, utilisez la commande suivante.

```
aws codeartifact update-package-versions-status --domain my_domain
--domain-owner 111122223333 --repository my_repo --format npm --namespace nestjs
--package core --versions 8.0.0 --target-status Unlisted
```

Mettre à jour le statut d'un package Maven

Les packages Maven ont toujours un identifiant de groupe, appelé espace de noms dans. CodeArtifact Utilisez le `--namespace` paramètre pour spécifier l'ID du groupe Maven lors de l'appel `update-package-versions-status`. Par exemple, pour archiver la version 2.13.1 du package `Mavenorg.apache.logging.log4j:log4j`, utilisez la commande suivante.

```
aws codeartifact update-package-versions-status --domain my_domain
--domain-owner 111122223333 --repository my_repo --format maven
--namespace org.apache.logging.log4j --package log4j
--versions 2.13.1 --target-status Archived
```

Spécification d'une révision de version de package

Une révision de version de package est une chaîne qui spécifie un ensemble spécifique de ressources et de métadonnées pour une version de package. Vous pouvez spécifier une révision de version de package pour mettre à jour le statut des versions de package qui se trouvent dans un état spécifique. Pour spécifier une révision de version de package, utilisez le `--version-revisions` paramètre pour transmettre une ou plusieurs versions de package séparées par des virgules et les paires de révisions de version de package. L'état d'une version de package ne sera mis à jour que si la révision actuelle de la version de package correspond à la valeur spécifiée.

Note

Le `--versions` paramètre doit également être défini lors de son `--version-revisions` utilisation.

```
aws codeartifact update-package-versions-status --domain my_domain
  --domain-owner 111122223333 --repository my_repo --format npm --package chalk
  --version-revisions "4.1.0=25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8bzVMJ4="
  --versions 4.1.0 --target-status Archived
```

Pour mettre à jour plusieurs versions à l'aide d'une seule commande, transmettez aux options une liste séparée par des virgules de paires de versions et de révisions de `--version-revisions` version. L'exemple de commande suivant définit deux paires différentes de version de package et de révision de version de package.

```
aws codeartifact update-package-versions-status --domain my_domain
  --domain-owner 111122223333 --repository my_repo --format npm
  --package chalk
  --version-revisions "4.1.0=25/UjBleHs1DZewk+zozoeqH/
R80Rc9gL1P8bzVMJ4=, 4.0.0=E3lhBp0R0bRTut4pkjV5c1AQGkgSA70xti16hMMzelc="
  --versions 4.1.0 4.0.0 --target-status Published
```

Exemple de sortie :

```
{
  "successfulVersions": {
    "4.0.0": {
      "revision": "E3lhBp0R0bRTut4pkjV5c1AQGkgSA70xti16hMMzelc=",
      "status": "Published"
    },
    "4.1.0": {
      "revision": "25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8bzVMJ4=",
      "status": "Published"
    }
  },
  "failedVersions": {}
}
```

Lors de la mise à jour de plusieurs versions de package, les versions transmises `--version-revisions` doivent être les mêmes que celles transmises `--versions`. Si une révision n'est pas spécifiée correctement, le statut de cette version ne sera pas mis à jour.

Utilisation du paramètre d'état attendu

La `update-package-versions-status` commande fournit le `--expected-status` paramètre qui permet de spécifier l'état actuel attendu d'une version de package. Si le statut actuel ne

correspond pas à la valeur transmise--expected-status, le statut de cette version du package ne sera pas mis à jour.

Par exemple, dans *my_repo*, les versions 4.0.0 et 4.1.0 du package npm ont chalk actuellement un statut de Published. Un appel indiquant un statut attendu de Unlisted échouera à mettre à jour les deux versions du package en raison de l'incompatibilité des statuts. `update-package-versions-status`

```
aws codeartifact update-package-versions-status --domain my_domain
--domain-owner 111122223333 --repository my_repo --format npm --package chalk
--versions 4.1.0 4.0.0 --target-status Archived --expected-status Unlisted
```

Exemple de sortie :

```
{  
  "successfulVersions": {},  
  "failedVersions": {  
    "4.0.0": {  
      "errorCode": "MISMATCHED_STATUS",  
      "errorMessage": "current status: Published, expected status: Unlisted"  
    },  
    "4.1.0": {  
      "errorCode": "MISMATCHED_STATUS",  
      "errorMessage": "current status: Published, expected status: Unlisted"  
    }  
  }  
}
```

Erreurs liées aux différentes versions de package

Il existe plusieurs raisons pour lesquelles le statut d'une version de package ne sera pas mis à jour lors de l'appel `update-package-versions-status`. Par exemple, la révision de la version du package a peut-être été spécifiée de manière incorrecte ou le statut attendu ne correspond pas à l'état actuel. Dans ces cas, la version sera incluse dans la `failedVersions` carte dans la réponse de l'API. Si une version échoue, les autres versions spécifiées dans le même appel à `update-package-versions-status` peuvent être ignorées et leur statut n'est pas mis à jour. Ces versions seront également incluses dans la `failedVersions` carte avec un `errorCode` de `SKIPPED`.

Dans l'implémentation actuelle de `update-package-versions-status`, si le statut d'une ou de plusieurs versions ne peut pas être modifié, toutes les autres versions seront ignorées. En d'autres

termes, soit toutes les versions sont mises à jour avec succès, soit aucune version n'est mise à jour. Ce comportement n'est pas garanti dans le contrat d'API ; à l'avenir, certaines versions pourraient réussir tandis que d'autres échoueront en un seul appel à `update-package-versions-status`.

L'exemple de commande suivant inclut un échec de mise à jour de l'état de version causé par une incompatibilité de version de package. Cet échec de mise à jour entraîne l'annulation d'un autre appel de mise à jour de l'état de la version.

```
aws codeartifact update-package-versions-status --domain my_domain
  --domain-owner 111122223333 --repository my_repo
  --format npm --package chalk
  --version-revisions "4.1.0=25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ=, 4.0.0=E3lhBp0R0bRTut4pkjV5c1AQGkgSA70xtiL6hMMze1c="
  --versions 4.1.0 4.0.0 --target-status Archived
```

Exemple de sortie :

```
{  
  "successfulVersions": {},  
  "failedVersions": {  
    "4.0.0": {  
      "errorCode": "SKIPPED",  
      "errorMessage": "version 4.0.0 is skipped"  
    },  
    "4.1.0": {  
      "errorCode": "MISMATCHED_REVISION",  
      "errorMessage": "current revision: 25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ4=, expected revision: 25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ="  
    }  
  }  
}
```

Élimination des versions du package

Le statut `Disposed` du package a un comportement similaire à celui du `packageArchived`, sauf que les actifs du package seront définitivement supprimés, de CodeArtifact sorte que le compte du propriétaire du domaine ne sera plus facturé pour le stockage des actifs. Pour plus d'informations sur le statut de chaque version de package, consultez [État de la version du package](#). Pour modifier le statut d'une version de package en `Disposed`, utilisez la `dispose-package-versions` commande. Cette fonctionnalité est distincte du `update-package-versions-status` fait que

la suppression d'une version de package n'est pas réversible. Dans la mesure où les actifs du package seront supprimés, le statut de la version ne peut pas être redéfini sur ArchivedUnlisted, ouPublished. La seule action qui peut être entreprise sur une version de package qui a été supprimée est de la supprimer à l'aide de la `delete-package-versions` commande.

Pour que l'appel soit `dispose-package-versions` réussi, le principal IAM appelant doit avoir l'`codeartifact:DisposePackageVersions` autorisation d'accéder à la ressource du package.

Le comportement de la `dispose-package-versions` commande est similaire à `update-package-versions-status` celui des `--expected-status` options `--version-revisions` et décrites dans les sections sur la [révision de la version](#) et sur le [statut attendu](#). Par exemple, la commande suivante tente de supprimer une version de package mais échoue en raison d'un état attendu qui ne correspond pas.

```
aws codeartifact dispose-package-versions --domain my_domain --domain-owner 111122223333 --repository my_repo --format npm --package chalk --versions 4.0.0 --expected-status Unlisted
```

Exemple de sortie :

```
{  
  "successfulVersions": {},  
  "failedVersions": {  
    "4.0.0": {  
      "errorCode": "MISMATCHED_STATUS",  
      "errorMessage": "current status: Published, expected status: Unlisted"  
    }  
  }  
}
```

Si la même commande est exécutée à nouveau avec un `--expected-status` de `Published`, la destruction réussira.

```
aws codeartifact dispose-package-versions --domain my_domain --domain-owner 111122223333 --repository my_repo --format npm --package chalk --versions 4.0.0 --expected-status Published
```

Exemple de sortie :

```
{  
  "successfulVersions": {  
    "4.0.0": {  
      "revision": "E3lhBp0R0bRTut4pkjV5c1AQGkgSA70xti16hMMzelc=",  
      "status": "Disposed"  
    }  
  },  
  "failedVersions": {}  
}
```

Modification des contrôles d'origine des packages

Dans AWS CodeArtifact, les versions de package peuvent être ajoutées à un référentiel en les publiant directement, en les extrayant d'un référentiel en amont ou en les ingérant depuis un référentiel public externe. Le fait d'autoriser l'ajout de versions d'un package à la fois par publication directe et par ingestion à partir de référentiels publics vous rend vulnérable aux attaques de substitution de dépendances. Pour de plus amples informations, veuillez consulter [Attaques de substitution de la dépendance](#). Pour vous protéger contre une attaque de substitution de dépendance, vous pouvez configurer les contrôles d'origine des packages sur un package dans un référentiel afin de limiter la manière dont les versions de ce package peuvent être ajoutées au référentiel.

La configuration des contrôles d'origine des packages doit être envisagée par toute équipe qui souhaite autoriser les nouvelles versions de différents packages à provenir à la fois de sources internes, telles que la publication directe, et de sources externes, telles que des référentiels publics. Par défaut, les contrôles d'origine des packages sont configurés en fonction de la manière dont la première version d'un package est ajoutée au référentiel. Pour plus d'informations sur les paramètres de contrôle de l'origine des packages et leurs valeurs par défaut, consultez [Paramètres de contrôle de l'origine des packages](#).

Pour supprimer l'enregistrement du package après avoir utilisé l'opération `put-package-origin-configuration` API, utilisez `delete-package` (voir [Supprimer un package ou une version de package](#)).

Scénarios courants de contrôle d'accès aux packages

Cette section inclut certains scénarios courants lorsqu'une version de package est ajoutée à un CodeArtifact référentiel. Les paramètres de contrôle de l'origine des packages seront définis pour les nouveaux packages en fonction de la manière dont la première version du package est ajoutée.

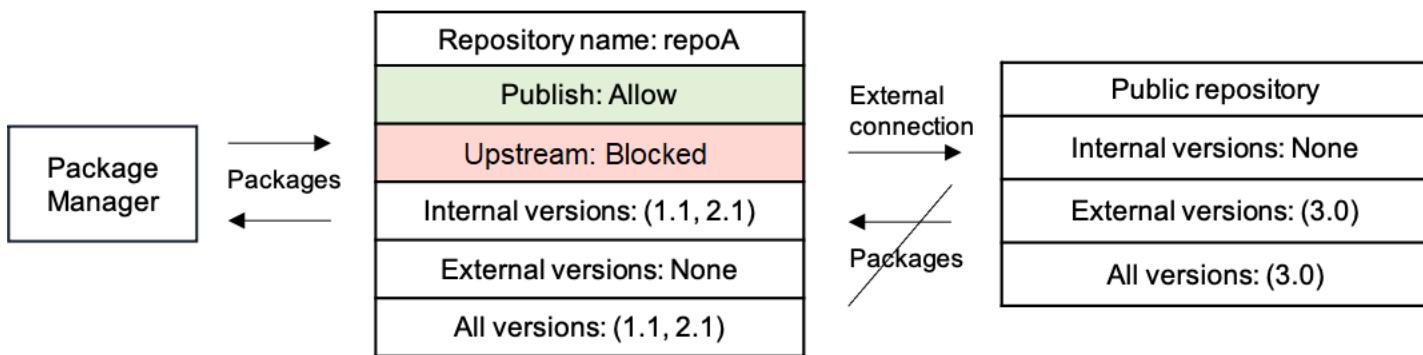
Dans les scénarios suivants, un package interne est un package publié directement depuis un gestionnaire de packages vers votre référentiel, tel qu'un package que vous ou votre équipe créez et gérez. Un package externe est un package existant dans un dépôt public qui peut être ingéré dans votre référentiel via une connexion externe.

Une version de package externe est publiée pour un package interne existant

Dans ce scénario, considérez un package interne, `PackageA`. Votre équipe publie la première version du package `PackageA` dans un CodeArtifact référentiel. Comme il s'agit de la première version de

package pour ce package, les paramètres de contrôle de l'origine du package sont automatiquement définis sur Publier : Autoriser et Upstream : Bloquer. Une fois que le package existe dans votre dépôt, un package portant le même nom est publié dans un dépôt public connecté à votre CodeArtifact dépôt. Il peut s'agir d'une tentative d'attaque de substitution de dépendance contre le package interne ou d'une simple coïncidence. Quoi qu'il en soit, les contrôles d'origine des packages sont configurés pour bloquer l'ingestion de la nouvelle version externe afin de se protéger contre une attaque potentielle.

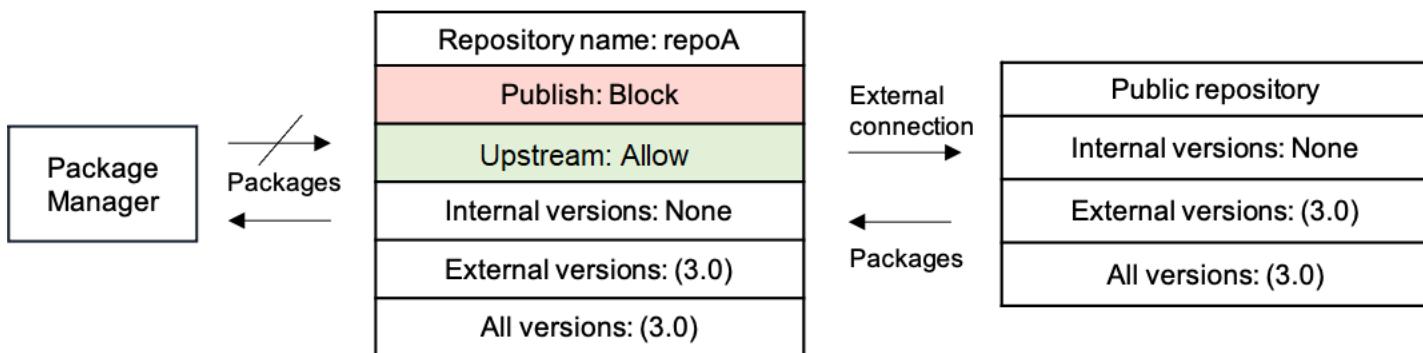
Dans l'image suivante, RePoA est votre CodeArtifact dépôt avec une connexion externe à un dépôt public. Votre dépôt contient les versions 1.1 et 2.1 de PackageA, mais la version 3.0 est publiée dans le dépôt public. Normalement, RePoA ingère la version 3.0 une fois que le package a été demandé par un gestionnaire de packages. L'ingestion de packages étant définie sur Bloquer, la version 3.0 n'est pas ingérée dans votre CodeArtifact référentiel et n'est pas disponible pour les gestionnaires de packages qui y sont connectés.



Une version de package interne est publiée pour un package externe existant

Dans ce scénario, un package, PackageB, existe en externe dans un référentiel public que vous avez connecté à votre référentiel. Lorsqu'un gestionnaire de packages connecté à votre référentiel demande PackageB, la version du package est ingérée dans votre référentiel depuis le référentiel public. Comme il s'agit de la première version de package de PackageB ajoutée à votre référentiel, les paramètres d'origine du package sont configurés sur Publish : BLOCK et Upstream : ALLOW. Plus tard, vous essayez de publier une version portant le même nom de package dans le référentiel. Soit vous ne connaissez pas le package public et vous essayez de publier un package indépendant sous le même nom, soit vous essayez de publier une version corrigée, soit vous essayez de publier directement la version exacte du package qui existe déjà en externe. CodeArtifact rejettéra la version que vous essayez de publier, mais vous permettra d'annuler explicitement le rejet et de publier la version si nécessaire.

Dans l'image suivante, RePoA est votre CodeArtifact dépôt avec une connexion externe à un dépôt public. Votre dépôt contient la version 3.0 qu'il a ingérée depuis le dépôt public. Vous souhaitez publier la version 1.1 dans votre dépôt. Normalement, vous pouvez publier la version 1.2 sur RePoA, mais comme la publication est définie sur Bloquer, la version 1.2 ne peut pas être publiée.



Publication d'une version de package corrigée d'un package externe existant

Dans ce scénario, un package, PackageB, existe en externe dans un référentiel public que vous avez connecté à votre référentiel. Lorsqu'un gestionnaire de packages connecté à votre référentiel demande PackageB, la version du package est ingérée dans votre référentiel depuis le référentiel public. Comme il s'agit de la première version de package de PackageB ajoutée à votre référentiel, les paramètres d'origine du package sont configurés sur Publish : BLOCK et Upstream : ALLOW. Votre équipe décide qu'elle doit publier les versions patchées de ce package dans le référentiel. Pour pouvoir publier directement les versions des packages, votre équipe modifie les paramètres de contrôle de l'origine des packages en Publish : ALLOW et Upstream : BLOCK. Les versions de ce package peuvent désormais être publiées directement dans votre dépôt et ingérées à partir de référentiels publics. Une fois que votre équipe a publié les versions du package corrigées, elle rétablit les paramètres d'origine du package sur Publish : BLOCK et Upstream : ALLOW.

Paramètres de contrôle de l'origine des packages

Grâce aux contrôles de l'origine des packages, vous pouvez configurer la manière dont les versions des packages peuvent être ajoutées à un référentiel. Les listes suivantes incluent les paramètres et valeurs de contrôle d'origine des packages disponibles.

Note

Les paramètres et valeurs disponibles sont différents lors de la configuration des contrôles d'origine sur les groupes de packages. Pour de plus amples informations, veuillez consulter [Contrôles d'origine des groupes de packages](#).

Publish

Ce paramètre permet de configurer si les versions des packages peuvent être publiées directement dans le référentiel à l'aide de gestionnaires de packages ou d'outils similaires.

- AUTORISER : les versions du package peuvent être publiées directement.
- BLOCK : Les versions du package ne peuvent pas être publiées directement.

En amont

Ce paramètre définit si les versions des packages peuvent être ingérées à partir de référentiels publics externes ou conservées à partir de référentiels en amont à la demande d'un gestionnaire de packages.

- AUTORISER : Toute version de package peut être conservée à partir d'autres CodeArtifact référentiels configurés en tant que référentiels en amont ou ingérée à partir d'une source publique via une connexion externe.
- BLOCAGE : Les versions de package ne peuvent pas être conservées à partir d'autres CodeArtifact référentiels configurés comme référentiels en amont ou ingérées à partir d'une source publique avec une connexion externe.

Paramètres de contrôle de l'origine des packages par défaut

Les paramètres de contrôle d'origine des packages par défaut sont configurés en fonction des paramètres de contrôle d'origine du groupe de packages associé au package. Pour plus d'informations sur les groupes de packages et les contrôles d'origine des groupes de packages, consultez [Utilisation de groupes de packages dans CodeArtifact](#) et [Contrôles d'origine des groupes de packages](#).

Si un package est associé à un groupe de packages avec des paramètres de restriction ALLOW correspondant à chaque type de restriction, les contrôles d'origine du package par défaut pour un package seront basés sur la manière dont la première version de ce package est ajoutée au référentiel.

- Si la première version du package est publiée directement par un gestionnaire de packages, les paramètres seront Publish : ALLOW et Upstream : BLOCK.
- Si la première version du package est ingérée à partir d'une source publique, les paramètres seront Publish : BLOCK et Upstream : ALLOW.

Note

Les packages qui existaient dans CodeArtifact des référentiels avant mai 2022 environ seront dotés de contrôles d'origine par défaut tels que Publish : ALLOW et Upstream : ALLOW. Les contrôles d'origine des packages doivent être définis manuellement pour ces packages. Les valeurs par défaut actuelles ont été définies sur les nouveaux packages depuis lors et ont commencé à être appliquées lorsque la fonctionnalité a été lancée le 14 juillet 2022. Pour plus d'informations sur la définition des contrôles d'origine des packages, consultez [Modification des contrôles d'origine des packages](#).

Sinon, si un package est associé à un groupe de packages dont au moins un paramètre de restriction est BLOCK ou ALLOW_SPECIFIC_REPOSITORIES, les paramètres de contrôle d'origine par défaut pour ce package seront définis sur Publish : ALLOW et Upstream : ALLOW.

Comment les contrôles d'origine des packages interagissent avec les contrôles d'origine des groupes de packages

Étant donné que les packages ont des paramètres de contrôle d'origine et que les groupes de packages associés ont des paramètres de contrôle d'origine, il est important de comprendre comment ces deux paramètres différents interagissent les uns avec les autres.

L'interaction entre les deux paramètres est qu'un paramètre de l'emporte BLOCK toujours sur un paramètre de ALLOW. Le tableau suivant répertorie quelques exemples de configurations et leurs paramètres de contrôle d'origine effectifs.

Paramètre de contrôle de l'origine du package	Paramètre de contrôle de l'origine du groupe de packages	Paramètre de contrôle d'origine efficace
PUBLIER : AUTORISER	PUBLIER : AUTORISER	PUBLIER : AUTORISER
EN AMONT : AUTORISER	EN AMONT : AUTORISER	EN AMONT : AUTORISER
PUBLIER : BLOQUER	PUBLIER : AUTORISER	PUBLIER : BLOQUER
EN AMONT : AUTORISER	EN AMONT : AUTORISER	EN AMONT : AUTORISER
PUBLIER : AUTORISER	PUBLIER : AUTORISER	PUBLIER : AUTORISER

Paramètre de contrôle de l'origine du package	Paramètre de contrôle de l'origine du groupe de packages	Paramètre de contrôle d'origine efficace
EN AMONT : AUTORISER	EN AMONT : BLOC	EN AMONT : BLOC

Cela signifie qu'un package dont les paramètres d'origine sont Publish : ALLOW et Upstream : ALLOW s'en remet effectivement aux paramètres de contrôle d'origine du groupe de packages associé.

Modification des contrôles d'origine des packages

Les contrôles d'origine des packages sont configurés automatiquement en fonction de la manière dont la première version d'un package est ajoutée au référentiel. Pour plus d'informations, voir [Paramètres de contrôle de l'origine des packages par défaut](#). Pour ajouter ou modifier des contrôles d'origine de package pour un package dans un CodeArtifact référentiel, effectuez les étapes de la procédure suivante.

Pour ajouter ou modifier les contrôles d'origine des packages (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez **Repositories**, puis le référentiel qui contient le package que vous souhaitez modifier.
3. Dans le tableau **Packages**, recherchez et sélectionnez le package que vous souhaitez modifier.
4. Sur la page récapitulative du package, dans les commandes **Origin**, choisissez **Modifier**.
5. Dans **Modifier les contrôles d'origine**, choisissez les contrôles d'origine du package que vous souhaitez définir pour ce package. Les deux paramètres de contrôle de l'origine du package, Publish et Upstream, doivent être définis en même temps.
 - Pour autoriser la publication directe des versions du package, dans **Publier**, sélectionnez **Autoriser**. Pour bloquer la publication des versions du package, choisissez **Bloquer**.
 - Pour autoriser l'ingestion de packages provenant de référentiels externes et l'extraction de packages depuis des référentiels en amont, dans **Sources en amont**, sélectionnez **Autoriser**. Pour bloquer toute ingestion et extraction de versions de packages depuis des référentiels externes et en amont, choisissez **Bloquer**.

Pour ajouter ou modifier les contrôles d'origine des packages (AWS CLI)

1. Si ce n'est pas le cas, configurez le AWS CLI en suivant les étapes décrites dans [Configuration avec AWS CodeArtifact](#).
2. Utilisez la `put-package-origin-configuration` commande pour ajouter ou modifier les contrôles d'origine des packages. Remplacez les champs suivants :
 - *my_domain* Remplacez-le par le CodeArtifact domaine qui contient le package que vous souhaitez mettre à jour.
 - *my_repo* Remplacez-le par le CodeArtifact référentiel qui contient le package que vous souhaitez mettre à jour.
 - *npm* Remplacez-le par le format du package que vous souhaitez mettre à jour.
 - *my_package* Remplacez-le par le nom du package que vous souhaitez mettre à jour.
 - Remplacez *ALLOW* et par *BLOCK* les paramètres de contrôle de l'origine des packages souhaités.

```
aws codeartifact put-package-origin-configuration --domain my_domain \  
--repository my_repo --format npm --package my_package \  
--restrictions publish=ALLOW,upstream=BLOCK
```

Publication et référentiels en amont

CodeArtifact n'autorise pas la publication de versions de packages présentes dans des référentiels en amont ou des référentiels publics accessibles. Supposons, par exemple, que vous souhaitez publier un package `com.mycompany.mypackage:1.0` Maven dans un référentiel et que vous `myrepo` disposiez `myrepo` d'un référentiel en amont avec une connexion externe à Maven Central. Envisagez les scénarios suivants.

1. Les paramètres de contrôle de l'origine du package `com.mycompany.mypackage` sont Publish : `ALLOW` et Upstream : `ALLOW`. S'il `com.mycompany.mypackage:1.0` est présent dans le référentiel en amont ou dans Maven Central, CodeArtifact rejette toute tentative de publication dans ce référentiel `myrepo` avec une erreur de conflit 409. Vous pouvez toujours publier une version différente, telle que `com.mycompany.mypackage:1.1`.
2. Les paramètres de contrôle de l'origine du package `com.mycompany.mypackage` sont Publish : `ALLOW` et Upstream : `BLOCK`. Vous pouvez publier dans votre référentiel n'importe quelle version

de com.mycompany.mypackage qui n'existe pas encore car les versions des packages ne sont pas accessibles.

3. Les paramètres de contrôle de l'origine du package com.mycompany.mypackage sont Publish : BLOCK et Upstream : ALLOW. Vous ne pouvez publier aucune version de package directement dans votre référentiel.

Utilisation de groupes de packages dans CodeArtifact

Les groupes de packages peuvent être utilisés pour appliquer une configuration à plusieurs packages qui correspondent à un modèle défini en utilisant le format du package, l'espace de noms du package et le nom du package. Vous pouvez utiliser des groupes de packages pour configurer plus facilement les contrôles d'origine des packages pour plusieurs packages. Les contrôles d'origine des packages sont utilisés pour bloquer ou autoriser l'ingestion ou la publication de nouvelles versions de packages, ce qui protège les utilisateurs contre les actions malveillantes connues sous le nom d'attaques de substitution de dépendances.

Chaque domaine contient CodeArtifact automatiquement un groupe de packages racine. Ce groupe de packages racine contient tous les packages et permet par défaut aux versions de packages d'entrer dans les référentiels du domaine à partir de tous les types d'origine. /* Le groupe de packages racine peut être modifié, mais ne peut pas être supprimé.

La fonctionnalité de configuration des groupes de packages fonctionne finalement de manière cohérente lors de la création d'un nouveau groupe de packages ou de la suppression d'un groupe de packages existant. Cela signifie que lors de la création ou de la suppression d'un groupe de packages, les contrôles d'origine seront appliqués aux packages associés attendus, mais avec un certain retard en raison du comportement cohérent éventuel. Le temps nécessaire pour atteindre une cohérence éventuelle dépend du nombre de groupes de packages dans le domaine ainsi que du nombre de packages dans le domaine. Il peut y avoir une brève période pendant laquelle les contrôles d'origine ne sont pas immédiatement reflétés sur les packages associés après la création ou la suppression d'un groupe de packages.

En outre, les mises à jour des contrôles d'origine des groupes de packages entrent en vigueur presque immédiatement. Contrairement à la création ou à la suppression de groupes de packages, les modifications apportées aux contrôles d'origine d'un groupe de packages existant sont répercutées sur les packages associés sans le même délai.

Ces rubriques contiennent des informations sur les groupes de packages dans AWS CodeArtifact.

Rubriques

- [Création d'un groupe de packages](#)
- [Afficher ou modifier un groupe de packages](#)
- [Supprimer un groupe de packages](#)

- [Contrôles d'origine des groupes de packages](#)
- [Syntaxe de définition du groupe de packages et comportement de correspondance](#)
- [Marquer un groupe de packages dans CodeArtifact](#)

Création d'un groupe de packages

Vous pouvez créer un groupe de packages à l'aide de la CodeArtifact console, du AWS Command Line Interface (AWS CLI) ou CloudFormation. Pour plus d'informations sur la gestion des groupes de CodeArtifact packages avec CloudFormation, consultez[Création de CodeArtifact ressources avec AWS CloudFormation](#).

Création d'un groupe de packages (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domains, puis le domaine dans lequel vous souhaitez créer un groupe de packages.
3. Choisissez Groupes de packages, puis choisissez Créer un groupe de packages.
4. Dans Définition du groupe de packages, entrez la définition du groupe de packages pour votre groupe de packages. La définition du groupe de packages détermine quels packages sont associés au groupe. Vous pouvez saisir la définition du groupe de packages manuellement avec du texte, ou vous pouvez utiliser le mode visuel pour effectuer des sélections et la définition du groupe de packages sera créée automatiquement.
5. Pour utiliser le mode visuel afin de créer la définition du groupe de packages :
 - a. Choisissez Visual pour passer en mode visuel.
 - b. Dans Format du package, choisissez le format des packages à associer à ce groupe.
 - c. Dans Namespace (Scope), choisissez les critères d'espace de noms auxquels vous souhaitez répondre.
 - Égal : correspond exactement à l'espace de noms spécifié. Si cette option est sélectionnée, entrez l'espace de noms correspondant.
 - Vide : associe les packages sans espace de noms.
 - Commence par un mot : fait correspondre les espaces de noms commençant par un mot spécifique. Si cette option est sélectionnée, entrez le mot de préfixe correspondant.

Pour plus d'informations sur les mots et les limites des mots, voir [Mots, limites de mots et correspondance de préfixes](#).

- Tout : Faites correspondre les packages dans tous les espaces de noms.
- d. Si l'option Egale, Blank ou Commence par un mot est sélectionnée, dans Nom du package, choisissez les critères de nom du package auxquels vous souhaitez répondre.
- Exactement égal : correspond exactement au nom du package spécifié. Si c'est le cas, entrez le nom du package correspondant.
 - Commence par le préfixe : fait correspondre les packages qui commencent par le préfixe spécifié.
 - Commence par un mot : fait correspondre les packages qui commencent par un mot spécifique. Si cette option est sélectionnée, entrez le mot de préfixe correspondant.
- Pour plus d'informations sur les mots et les limites des mots, voir [Mots, limites de mots et correspondance de préfixes](#).
- Tous : Faites correspondre tous les packages.
- e. Choisissez Next pour revoir la définition.
6. Pour saisir la définition du groupe de packages avec du texte :
- a. Choisissez Texte pour passer en mode texte.
 - b. Dans Définition du groupe de packages, entrez la définition du groupe de packages. Pour plus d'informations sur la syntaxe de définition des groupes de packages, consultez [Syntaxe de définition du groupe de packages et comportement de correspondance](#).
 - c. Choisissez Next pour revoir la définition.
7. Dans Réviser la définition, passez en revue les packages qui seront inclus dans le nouveau groupe de packages en fonction de la définition fournie précédemment. Après avoir passé en revue, choisissez Next.
8. Dans Informations sur le groupe de packages, ajoutez éventuellement une description et une adresse e-mail de contact pour le groupe de packages. Choisissez Suivant.
9. Dans Contrôles d'origine des packages, configurez les contrôles d'origine à appliquer aux packages du groupe. Pour plus d'informations sur les contrôles d'origine des groupes de packages, consultez [Contrôles d'origine des groupes de packages](#).
10. Choisissez Créer un groupe de packages.

Création d'un groupe de packages (AWS CLI)

Utilisez la `create-package-group` commande pour créer un groupe de packages dans votre domaine. Pour l'--package-group option, entrez la définition du groupe de packages qui détermine les packages associés au groupe. Pour plus d'informations sur la syntaxe de définition des groupes de packages, consultez [Syntaxe de définition du groupe de packages et comportement de correspondance](#).

Si ce n'est pas le cas, configurez le AWS CLI en suivant les étapes décrites dans [Configuration avec AWS CodeArtifact](#).

```
aws codeartifact create-package-group \
  --domain my_domain \
  --package-group '/nuget/*' \
  --domain-owner 111122223333 \
  --contact-info contact@email.com \
  --description "a new package group" \
  --tags key=key1,value=value1
```

Afficher ou modifier un groupe de packages

Vous pouvez consulter la liste de tous les groupes de packages, afficher les détails d'un groupe de packages spécifique ou modifier les détails ou la configuration d'un groupe de packages à l'aide de la CodeArtifact console ou du AWS Command Line Interface (AWS CLI).

Afficher ou modifier un groupe de packages (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, sélectionnez Domaines, puis choisissez le domaine qui contient le groupe de packages que vous souhaitez afficher ou modifier.
3. Choisissez Groupes de packages, puis choisissez le groupe de packages que vous souhaitez afficher ou modifier.
4. Dans Détails, consultez les informations relatives au groupe de packages, notamment son groupe parent, sa description, son ARN, son adresse e-mail de contact et les contrôles d'origine des packages.
5. Dans Sous-groupes, consultez la liste des groupes de packages dont ce groupe est le groupe parent. Les groupes de packages de cette liste peuvent hériter des paramètres de ce groupe.

de packages. Pour de plus amples informations, veuillez consulter [Hiérarchie des groupes de packages et spécificité du modèle](#).

6. Dans Packages, visualisez les packages appartenant à ce groupe de packages en fonction de la définition du groupe de packages. Dans la colonne Force, vous pouvez voir la force de l'association de packages. Pour de plus amples informations, veuillez consulter [Hiérarchie des groupes de packages et spécificité du modèle](#).
7. Pour modifier les informations du groupe de packages, choisissez Modifier le groupe de packages.
 - a. Dans Informations, mettez à jour la description ou les coordonnées du groupe de packages. Vous ne pouvez pas modifier la définition d'un groupe de packages.
 - b. Dans Contrôles d'origine des groupes de packages, mettez à jour les paramètres de contrôle d'origine du groupe de packages, qui déterminent comment les packages associés peuvent entrer dans les référentiels du domaine. Pour de plus amples informations, veuillez consulter [Contrôles d'origine des groupes de packages](#).

Afficher ou modifier un groupe de packages (AWS CLI)

Utilisez les commandes suivantes pour afficher ou modifier des groupes de packages avec le AWS CLI. Si ce n'est pas le cas, configurez le AWS CLI en suivant les étapes décrites dans[Configuration avec AWS CodeArtifact](#).

Pour afficher tous les groupes de packages d'un domaine, utilisez la `list-package-groups` commande.

```
aws codeartifact list-package-groups \
  --domain my_domain \
  --domain-owner 111122223333
```

Pour afficher les détails d'un groupe de packages, utilisez la `describe-package-group` commande. Pour plus d'informations sur les définitions de groupes de packages, consultez[Syntaxe et exemples de définition de groupes de packages](#).

```
aws codeartifact describe-package-group \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group '/nuget/*'
```

Pour afficher les groupes de packages enfants d'un groupe de packages, utilisez la `list-sub-package-groups` commande.

```
aws codeartifact list-sub-package-groups \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group '/nuget/*' \
```

Pour afficher le groupe de packages associé à un package, utilisez la `get-associated-package-group` commande. Vous devez utiliser le nom de package et l'espace de noms normalisés pour les NuGet formats de package Python et Swift. Pour plus d'informations sur la façon dont les noms de packages et les espaces de noms sont normalisés, consultez la [NuGet documentation de normalisation des noms Python](#) et [Swift](#).

```
aws codeartifact get-associated-package-group \
  --domain my_domain \
  --domain-owner 111122223333 \
  --format npm \
  --package packageName \
  --namespace scope \
```

Pour modifier un groupe de packages, utilisez la `update-package-group` commande. Cette commande est utilisée pour mettre à jour les coordonnées ou la description d'un groupe de packages. Pour plus d'informations sur les paramètres de contrôle d'origine des groupes de packages, ainsi que sur leur ajout ou leur modification, consultez [Contrôles d'origine des groupes de packages](#). Pour plus d'informations sur les définitions de groupes de packages, voir [Syntaxe et exemples de définition de groupes de packages](#)

```
aws codeartifact update-package-group \
  --domain my_domain \
  --package-group '/nuget/*' \
  --domain-owner 111122223333 \
  --contact-info contact@email.com \
  --description "updated package group description" \
```

Supprimer un groupe de packages

Vous pouvez supprimer un groupe de packages à l'aide de la CodeArtifact console ou du AWS Command Line Interface (AWS CLI).

Notez le comportement suivant lors de la suppression de groupes de packages :

- Le groupe de packages racine ne peut pas être supprimé. /*
- Les packages et les versions de packages associés à ce groupe de packages ne sont pas supprimés.
- Lorsqu'un groupe de packages est supprimé, les groupes de packages enfants directs deviennent les enfants du groupe de packages parent direct du groupe de packages. Par conséquent, si l'un des groupes d'enfants hérite des paramètres du parent, ces paramètres peuvent changer.

Supprimer un groupe de packages (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, sélectionnez Domaines, puis choisissez le domaine qui contient le groupe de packages que vous souhaitez afficher ou modifier.
3. Choisissez Package groups.
4. Choisissez le groupe de packages que vous souhaitez supprimer, puis cliquez sur Supprimer.
5. Entrez Supprimer dans le champ et choisissez Supprimer.

Supprimer un groupe de packages (AWS CLI)

Pour supprimer un groupe de packages, utilisez la `delete-package-group` commande.

```
aws codeartifact delete-package-group \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group '/nuget/*'
```

Contrôles d'origine des groupes de packages

Les contrôles d'origine des packages sont utilisés pour configurer la manière dont les versions de package peuvent entrer dans un domaine. Vous pouvez configurer des contrôles d'origine sur un groupe de packages afin de configurer la manière dont les versions de chaque package associé au groupe de packages peuvent entrer dans les référentiels spécifiés du domaine.

Les paramètres de contrôle de l'origine des groupes de packages sont les suivants :

- **Paramètres de restriction:** Ces paramètres définissent si les packages peuvent entrer dans un référentiel CodeArtifact depuis la publication, depuis des flux internes ou depuis des référentiels publics externes.
- **Listes de référentiels autorisés:** Chaque paramètre de restriction peut être défini pour autoriser des référentiels spécifiques. Si un paramètre de restriction est défini pour autoriser des référentiels spécifiques, cette restriction sera associée à une liste de référentiels autorisés correspondante.

Note

Les paramètres de contrôle d'origine pour les groupes de packages sont légèrement différents des paramètres de contrôle d'origine pour les packages individuels. Pour plus d'informations sur les paramètres de contrôle d'origine des packages, consultez [Paramètres de contrôle de l'origine des packages](#).

Paramètres de restriction

Les paramètres de restriction des paramètres de contrôle d'origine d'un groupe de packages déterminent la manière dont les packages associés à ce groupe peuvent entrer dans les référentiels du domaine.

PUBLISH

Le PUBLISH paramètre définit si les versions de package peuvent être publiées directement dans n'importe quel référentiel du domaine à l'aide de gestionnaires de packages ou d'outils similaires.

- AUTORISER : les versions des packages peuvent être publiées directement dans tous les référentiels.
- BLOCK : Les versions des packages ne peuvent être publiées directement dans aucun référentiel.
- ALLOW_SPECIFIC_REPOSITORIES : Les versions des packages ne peuvent être publiées directement que dans les référentiels spécifiés dans la liste des référentiels autorisés pour la publication.
- INHERIT : le PUBLISH paramètre est hérité du premier groupe de packages parent avec un paramètre qui ne l'est pas INHERIT.

EXTERNAL_AMONT

Le EXTERNAL_UPSTREAM paramètre définit si les versions des packages peuvent être ingérées à partir de référentiels publics externes à la demande d'un gestionnaire de packages. Pour obtenir la liste des référentiels externes pris en charge, consultez [Référentiels de connexions externes pris en charge](#).

- AUTORISER : Toute version de package peut être ingérée dans tous les référentiels à partir d'une source publique avec une connexion externe.
- BLOCK : Les versions du package ne peuvent être ingérées dans aucun référentiel à partir d'une source publique dotée d'une connexion externe.
- ALLOW_SPECIFIC_REPOSITORIES : Les versions des packages peuvent uniquement être ingérées à partir d'une source publique dans les référentiels spécifiés dans la liste des référentiels autorisés pour les flux ascendants externes.
- INHERIT : le EXTERNAL_UPSTREAM paramètre est hérité du premier groupe de packages parent avec un paramètre qui ne l'est pas INHERIT.

INTERNE_AMONT

Le INTERNAL_UPSTREAM paramètre définit si les versions des packages peuvent être conservées à partir des référentiels internes en amont du même CodeArtifact domaine à la demande d'un gestionnaire de packages.

- AUTORISER : Toute version de package peut être conservée à partir d'autres CodeArtifact référentiels configurés en tant que référentiels en amont.
- BLOCK : Les versions des packages ne peuvent pas être conservées à partir d'autres CodeArtifact référentiels configurés comme référentiels en amont.
- ALLOW_SPECIFIC_REPOSITORIES : Les versions des packages ne peuvent être conservées que depuis CodeArtifact d'autres référentiels configurés comme référentiels en amont vers des référentiels spécifiés dans la liste des référentiels autorisés pour les référentiels internes en amont.
- INHERIT : le INTERNAL_UPSTREAM paramètre est hérité du premier groupe de packages parent avec un paramètre qui ne l'est pas INHERIT.

Listes de référentiels autorisés

Lorsqu'un paramètre de restriction est configuré comme `su ALLOW_SPECIFIC_REPOSITORIES`, le groupe de packages contient une liste de référentiels autorisés qui contient la liste des référentiels autorisés pour ce paramètre de restriction. Par conséquent, un groupe de packages contient de 0 à 3 listes de référentiels autorisés, une pour chaque paramètre configuré comme `ALLOW_SPECIFIC_REPOSITORIES`.

Lorsque vous ajoutez un dépôt à la liste des référentiels autorisés d'un groupe de packages, vous devez spécifier à quelle liste de référentiels autorisés l'ajouter.

Les listes de référentiels autorisés possibles sont les suivantes :

- `EXTERNAL_UPSTREAM`: autorisez ou bloquez l'ingestion de versions de packages provenant de référentiels externes dans le référentiel ajouté.
- `INTERNAL_UPSTREAM`: autorisez ou bloquez l'extraction de versions de packages depuis un autre CodeArtifact référentiel dans le référentiel ajouté.
- `PUBLISH`: autorisez ou bloquez la publication directe des versions de packages depuis les gestionnaires de packages vers le référentiel ajouté.

Modification des paramètres de contrôle d'origine des groupes de packages

Pour ajouter ou modifier des contrôles d'origine pour un groupe de packages, effectuez les étapes de la procédure suivante. Pour plus d'informations sur les paramètres de contrôle d'origine des groupes de packages, reportez-vous [Paramètres de restriction](#) aux sections et [Listes de référentiels autorisés](#).

Pour ajouter ou modifier des contrôles d'origine de groupes de packages (CLI)

1. Si ce n'est pas le cas, configurez le AWS CLI en suivant les étapes décrites dans [Configuration avec AWS CodeArtifact](#).
2. Utilisez la `update-package-group-origin-configuration` commande pour ajouter ou modifier les contrôles d'origine des packages.
 - Pour `--domain`, entrez le CodeArtifact domaine qui contient le groupe de packages que vous souhaitez mettre à jour.
 - Pour `--domain-owner`, entrez le numéro de compte du propriétaire du domaine.
 - Pour `--package-group`, entrez le groupe de packages que vous souhaitez mettre à jour.

- Pour `--restrictions`, entrez des paires clé-valeur qui représentent les restrictions de contrôle d'origine.
- Pour `--add-allowed-repositories`, entrez un objet JSON contenant le type de restriction et le nom du référentiel à ajouter à la liste des référentiels autorisés correspondante pour la restriction.
- Pour `--remove-allowed-repositories`, entrez un objet JSON contenant le type de restriction et le nom du référentiel à supprimer de la liste des référentiels autorisés correspondant à la restriction.

```
aws codeartifact update-package-group-origin-configuration \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group '/nuget/*' \
  --restrictions INTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES \
  --add-allowed-repositories
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo \
  --remove-allowed-repositories
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2
```

L'exemple suivant ajoute plusieurs restrictions et plusieurs référentiels dans une seule commande.

```
aws codeartifact update-package-group-origin-configuration \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group '/nuget/*' \
  --
restrictions PUBLISH=BLOCK,EXTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES,INTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES \
  \
  --add-allowed-repositories
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2 \
  --remove-allowed-repositories
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2
```

Exemples de configuration du contrôle d'origine des groupes de packages

Les exemples suivants présentent les configurations de contrôle de l'origine des packages pour les scénarios courants de gestion des packages.

Autoriser la publication de packages avec des noms privés, mais pas leur ingestion

Ce scénario est probablement courant dans le domaine de la gestion des packages :

- Autorisez la publication des packages avec des noms privés dans les référentiels de votre domaine à partir des gestionnaires de packages, et empêchez leur ingestion dans les référentiels de votre domaine à partir de référentiels publics externes.
- Autorisez l'ingestion de tous les autres packages dans les référentiels de votre domaine à partir de référentiels publics externes, et bloquez leur publication dans les référentiels de votre domaine à partir des gestionnaires de packages.

Pour ce faire, vous devez configurer un groupe de packages avec un modèle incluant le ou les noms privés et les paramètres d'origine de PUBLISH : ALLOW, EXTERNAL_UPSTREAM : BLOCK et INTERNAL_UPSTREAM : ALLOW. Cela permettra de garantir que les packages avec des noms privés peuvent être publiés directement, mais ne peuvent pas être ingérés à partir de référentiels externes.

Les AWS CLI commandes suivantes créent et configurent un groupe de packages avec des paramètres de restriction d'origine correspondant au comportement souhaité :

Pour créer le groupe de packages :

```
aws codeartifact create-package-group \
  --domain my_domain \
  --package-group /npm/space/anycompany~ \
  --domain-owner 111122223333 \
  --contact-info contact@email.com | URL \
  --description "my package group"
```

Pour mettre à jour la configuration d'origine du groupe de packages :

```
aws codeartifact update-package-group-origin-configuration \
  --domain my_domain \
  --domain-owner 111122223333 \
```

```
--package-group '/npm/space/anycompany~' \
--restrictions PUBLISH=ALLOW,EXTERNAL_UPSTREAM=BLOCK,INTERNAL_UPSTREAM=ALLOW
```

Autoriser l'ingestion à partir de référentiels externes via un seul référentiel

Dans ce scénario, votre domaine possède plusieurs référentiels. Parmi ces référentiels, `repoA` dispose d'une connexion en amont à `repoB`, qui possède une connexion externe au référentiel `publicnpmjs.com`, comme indiqué :

```
repoA --> repoB --> npmjs.com
```

Vous souhaitez autoriser l'ingestion de packages provenant d'un groupe de packages spécifique, `/npm/space/anycompany~` de destination à `repoA`, mais uniquement de destination `npmjs.com` en direction `repoB`. Vous souhaitez également bloquer l'ingestion de packages associés au groupe de packages dans tout autre référentiel de votre domaine et bloquer la publication directe de packages avec les gestionnaires de packages. Pour ce faire, vous devez créer et configurer le groupe de packages comme suit :

Paramètres de restriction d'origine pour PUBLISH : BLOCK, EXTERNAL_UPSTREAM : ALLOW_SPECIFIC_REPOSITORIES, et INTERNAL_UPSTREAM : ALLOW_SPECIFIC_REPOSITORIES.

`repoA` et `repoB` ajoutés à la liste des référentiels autorisés appropriée :

- `repoA` devrait être ajouté à la INTERNAL_UPSTREAM liste, car il obtiendra les packages depuis son amont interne, `repoB`.
- `repoB` doit être ajouté à la EXTERNAL_UPSTREAM liste, car il obtiendra les packages du référentiel externe, `npmjs.com`.

Les AWS CLI commandes suivantes créent et configurent un groupe de packages avec des paramètres de restriction d'origine correspondant au comportement souhaité :

Pour créer le groupe de packages :

```
aws codeartifact create-package-group \
--domain my_domain \
--package-group /npm/space/anycompany~ \
--domain-owner 111122223333 \
--contact-info contact@email.com | URL \
```

```
--description "my package group"
```

Pour mettre à jour la configuration d'origine du groupe de packages :

```
aws codeartifact update-package-group-origin-configuration \
  --domain my_domain \
  --domain-owner 111122223333 \
  --package-group /npm/space/anycompany~ \
  --
  restrictions PUBLISH=BLOCK,EXTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES,INTERNAL_UPSTREAM=ALLOW_ALL
  \
  --add-allowed-repositories
  originRestrictionType=INTERNAL_UPSTREAM,repositoryName=repoA
  originRestrictionType=EXTERNAL_UPSTREAM,repositoryName=repoB
```

Comment les paramètres de contrôle d'origine des groupes de packages interagissent avec les paramètres de contrôle de l'origine des packages

Étant donné que les packages ont des paramètres de contrôle d'origine et que les groupes de packages associés ont des paramètres de contrôle d'origine, il est important de comprendre comment ces deux paramètres différents interagissent les uns avec les autres. Pour plus d'informations sur l'interaction entre les paramètres, consultez [Comment les contrôles d'origine des packages interagissent avec les contrôles d'origine des groupes de packages](#).

Syntaxe de définition du groupe de packages et comportement de correspondance

Cette rubrique contient des informations sur la définition des groupes de packages, le comportement de correspondance des modèles, la force des associations de packages et la hiérarchie des groupes de packages.

Table des matières

- [Syntaxe et exemples de définition de groupes de packages](#)
 - [Définition et normalisation des groupes de packages](#)
 - [Espaces de noms dans les définitions de groupes de packages](#)
- [Hiérarchie des groupes de packages et spécificité du modèle](#)
- [Mots, limites de mots et correspondance de préfixes](#)

- [Sensibilité à la casse](#)
- [Match fort et faible](#)
- [Variations supplémentaires](#)

Syntaxe et exemples de définition de groupes de packages

La syntaxe du modèle pour définir les groupes de packages suit de près le formatage des chemins de package. Le chemin d'un package est créé à partir des coordonnées d'un package (format, espace de noms et nom) en ajoutant une barre oblique au début et en séparant chacun des composants par une barre oblique. Par exemple, le chemin du package npm nommé anycompany-ui-components dans l'espace de noms est/npm/space/anycompany-ui-components.

Un modèle de groupe de packages suit la même structure qu'un chemin de package, sauf que les composants qui ne sont pas spécifiés dans le cadre de la définition du groupe sont omis et que le modèle se termine par un suffixe. Le suffixe inclus détermine le comportement correspondant du modèle, comme suit :

- Un \$ suffixe correspondra aux coordonnées complètes du colis.
- Un ~ suffixe correspondra à un préfixe.
- Un * suffixe correspondra à toutes les valeurs du composant défini précédemment.

Voici des exemples de modèles pour chacune des combinaisons autorisées :

1. Tous les formats de package : /*
2. Un format de package spécifique : /npm/*
3. Format du package et préfixe d'espace de noms : /maven/com.anycompany~
4. Format du package et espace de noms : /npm/space/*
5. Format du package, espace de noms et préfixe de nom : /npm/space/anycompany-ui~
6. Format, espace de noms et nom du package : /maven/org.apache.logging.log4j/log4j-core\$

Comme le montrent les exemples ci-dessus, le ~ suffixe est ajouté à la fin d'un espace de noms ou d'un nom pour représenter une correspondance de préfixe et * vient après une barre oblique lorsqu'il est utilisé pour correspondre à toutes les valeurs du composant suivant dans le chemin (tous les formats, tous les espaces de noms ou tous les noms).

Définition et normalisation des groupes de packages

CodeArtifact normalise les NuGet noms de packages Python et Swift, et normalise les espaces de noms de packages Swift avant de les stocker. CodeArtifact utilise ces noms normalisés lors de la mise en correspondance de packages avec des définitions de groupes de packages. Par conséquent, les groupes de packages contenant un espace de noms ou un nom dans ces formats doivent utiliser l'espace de noms et le nom normalisés. Pour plus d'informations sur la façon dont les noms de packages et les espaces de noms sont normalisés, consultez la [NuGet documentation de normalisation des noms Python](#) et [Swift](#).

Espaces de noms dans les définitions de groupes de packages

Pour les packages ou les formats de packages sans espace de noms (Python et NuGet), les groupes de packages ne doivent pas contenir d'espace de noms. La définition du groupe de packages pour ces groupes de packages contient une section d'espace de noms vide. Par exemple, le chemin du package Python nommé requests est /python//requests.

Pour les packages ou les formats de package dotés d'un espace de noms (Maven, generic et Swift), l'espace de noms doit être inclus si le nom du package est inclus. Pour le format de package Swift, l'espace de noms de package normalisé sera utilisé. Pour plus d'informations sur la façon dont les espaces de noms des packages Swift sont normalisés, consultez [Normalisation rapide du nom du package et de l'espace de noms](#)

Hiérarchie des groupes de packages et spécificité du modèle

Les packages « inclus » ou « associés à » un groupe de packages sont des packages dont le chemin de package correspond au modèle du groupe mais ne correspond pas au modèle d'un groupe plus spécifique. Par exemple, étant donné les groupes de packages /npm/* et /npm/space/*, le chemin du package /npm//react est associé au premier groupe (/npm/*) tandis que /npm/space/aui.components et /npm/space/amplify-ui-core sont associés au second groupe (. /npm/space/*). Même si un package peut correspondre à plusieurs groupes, chaque package n'est associé qu'à un seul groupe, la correspondance la plus spécifique, et seule la configuration de ce groupe s'applique au package.

Lorsqu'un chemin de package correspond à plusieurs modèles, le modèle « plus spécifique » peut être considéré comme le modèle correspondant le plus long. Sinon, le modèle le plus spécifique est celui qui correspond à un sous-ensemble approprié des packages correspondant au modèle le moins spécifique. Dans notre exemple précédent, chaque package correspondant correspond /npm/space/* également /npm/*, mais l'inverse n'est pas vrai, ce qui rend /npm/space/* le modèle

plus spécifique car il s'agit d'un sous-ensemble approprié de `/npm/*`. Comme un groupe est un sous-ensemble d'un autre groupe, il crée une hiérarchie dans laquelle `/npm/space/*` se trouve un sous-groupe du groupe parent. `/npm/*`

Bien que seule la configuration du groupe de packages le plus spécifique s'applique à un package, ce groupe peut être configuré pour hériter de la configuration de son groupe parent.

Mots, limites de mots et correspondance de préfixes

Avant de discuter de la correspondance des préfixes, définissons quelques termes clés :

- Un mot est une lettre ou un chiffre suivi de zéro ou de plusieurs lettres, chiffres ou caractères de marque (tels que des accents, des trémas, etc.).
- La limite d'un mot se trouve à la fin d'un mot, lorsqu'un caractère autre qu'un mot est atteint. Les caractères autres que les mots sont des caractères de ponctuation tels que `.`, `-`, et `_`.

Plus précisément, le modèle regex pour un mot est `[\p{L}\p{N}][\p{L}\p{N}\p{M}]*`, qui peut être décomposé comme suit :

- `\p{L}` représente n'importe quelle lettre.
- `\p{N}` représente un nombre quelconque.
- `\p{M}` représente n'importe quel caractère de marque, tel que les accents, les trémas, etc.

Par conséquent, `[\p{L}\p{N}]` représente un chiffre ou une lettre, et `[\p{L}\p{N}\p{M}]*` représente zéro ou plusieurs lettres, chiffres ou caractères de marque et une limite de mot se trouve à la fin de chaque correspondance de ce modèle d'expression régulière.

Note

La correspondance des limites des mots est basée sur cette définition d'un « mot ». Il n'est pas basé sur des mots définis dans un dictionnaire, ou CameCase. Par exemple, il n'y a pas de limite de mots dans `oneword` ou `OneWord`.

Maintenant que le mot et la limite des mots sont définis, nous pouvons les utiliser pour décrire la correspondance des préfixes dans CodeArtifact. Pour indiquer une correspondance de préfixe sur la limite d'un mot, un caractère correspondant (`~`) est utilisé après un caractère de mot. Par exemple,

le modèle `/npm/space/foo~` correspond aux chemins du package `/npm/space/foo` et `/npm/space/foo-bar`, mais pas à `/npm/space/food` ou `/npm/space/foot`.

Un caractère générique (*) doit être utilisé plutôt que `~` lorsque vous suivez un caractère autre qu'un mot, comme dans le modèle. `/npm/*`

Sensibilité à la casse

Les définitions de groupes de packages font la distinction majuscules/majuscules, ce qui signifie que les modèles qui ne diffèrent que par majuscules peuvent exister en tant que groupes de packages distincts. Par exemple, un utilisateur peut créer des groupes de packages distincts avec les modèles `/npm//AsyncStorage$`/`/npm//asyncStorage$`, et `/npm//asyncstorage$` pour les trois packages distincts qui existent dans le registre public npm : `AsyncStorage`, `AsyncStorage`, `asyncstorage` qui ne diffèrent qu'au cas par cas.

Bien que le cas soit important, associez CodeArtifact toujours les packages à un groupe de packages si le package présente une variation du modèle qui diffère au cas par cas. Si un utilisateur crée le groupe de `/npm//AsyncStorage$` packages sans créer les deux autres groupes indiqués ci-dessus, toutes les variantes majuscules du nom `AsyncStorage`, y compris `AsyncStorage` et `asyncstorage`, seront associées au groupe de packages. Mais, comme décrit dans la section suivante [Match fort et faible](#), ces variations seront traitées différemment de `AsyncStorage` qui correspond exactement au modèle.

Match fort et faible

Les informations de la section précédente indiquent que [Sensibilité à la casse](#) les groupes de packages distinguent les majuscules et minuscules, puis expliquent qu'ils ne le font pas. Cela est CodeArtifact dû au fait que les définitions des groupes de packages reposent sur un concept de correspondance forte (ou correspondance exacte) et de correspondance faible (ou correspondance de variation). Une bonne correspondance se produit lorsque l'emballage correspond exactement au modèle, sans aucune variation. Une faible correspondance se produit lorsque le colis correspond à une variante du modèle, par exemple une majuscule différente. Un comportement de correspondance faible empêche les packages qui sont des variantes du modèle d'un groupe de packages de se transformer en un groupe de packages plus général. Lorsqu'un package est une variante (correspondance faible) du modèle du groupe correspondant le plus spécifique, le package est associé au groupe mais le package est bloqué au lieu d'appliquer la configuration de contrôle d'origine du groupe, empêchant ainsi toute nouvelle version du package d'être extraite des flux ascendants ou publiée. Ce comportement réduit le risque d'attaques de la

chaîne d'approvisionnement résultant de la confusion des dépendances entre des packages portant des noms presque identiques.

Pour illustrer le comportement de faible correspondance, supposons que le groupe de packages `/npm/*` autorise l'ingestion et bloque la publication. Un groupe de packages plus spécifique est configuré pour bloquer l'ingestion et autoriser la publication. `/npm//anycompany-spicy-client$` Le package nommé `anycompany-spicy-client` correspond parfaitement au groupe de packages, ce qui permet de publier des versions de package et bloque l'ingestion de versions de package. La seule case du nom du package qui peut être publiée est `anycompany-spicy-client` celle qui correspond parfaitement au modèle de définition du package. La publication d'une variante différente, telle que `AnyCompany-spicy-client`, est bloquée en raison d'une faible correspondance. Plus important encore, le groupe de packages bloque l'ingestion de toutes les variantes majuscules, et pas seulement du nom en minuscules utilisé dans le modèle, réduisant ainsi le risque d'attaque par confusion des dépendances.

Variations supplémentaires

Outre les différences majuscules/minuscules, la faible correspondance ignore également les différences entre les séquences de tirets-, de points ., de traits de soulignement _ et de caractères confuses (tels que les caractères d'apparence similaire issus d'alphabets distincts). Lors de la normalisation utilisée pour les correspondances faibles, CodeArtifact effectue le pliage en majuscules (comme lors de la conversion en minuscules), remplace les séquences de tirets, de points et de soulignements par un seul point et normalise les caractères susceptibles de confusion.

La correspondance faible traite les tirets, les points et les traits de soulignement comme équivalents, mais ne les ignore pas complètement. Cela signifie que `foo-bar`, `foo.bar`, `foo.. bar` et `foo_bar` sont tous des équivalents à faible correspondance, mais pas `foobar`. Bien que plusieurs référentiels publics mettent en œuvre des mesures pour empêcher ces types de variations, la protection fournie par les référentiels publics ne rend pas cette fonctionnalité des groupes de packages inutile. Par exemple, les référentiels publics tels que le registre public `npm` n'empêcheront les nouvelles variantes du package nommé `my-package` que si `my-package` y est déjà publié. Si `my-package` est un package interne et qu'un groupe de packages `/npm//my-package$` est créé pour autoriser la publication et bloquer l'ingestion, vous ne voudrez probablement pas publier `mon-package` dans le registre public `npm` afin d'empêcher une variante telle que `my.package` d'être autorisée.

Bien que certains formats de package tels que Maven traitent ces caractères différemment (Maven les traite . comme un séparateur de hiérarchie d'espaces de noms, mais pas - ou_), des éléments tels que `com.act-on` peuvent toujours être confondus avec `com.act.on`.

Note

Notez que chaque fois que plusieurs variantes sont associées à un groupe de packages, un administrateur peut créer un nouveau groupe de packages pour une variante spécifique afin de configurer un comportement différent pour cette variante.

Marquer un groupe de packages dans CodeArtifact

Les balises sont des paires clé-valeur associées aux ressources AWS. Vous pouvez appliquer des balises à vos groupes de packages dans CodeArtifact. Pour plus d'informations sur le balisage CodeArtifact des ressources, les cas d'utilisation, les contraintes de clé et de valeur de balise et les types de ressources pris en charge, consultez [Balisage des ressources](#).

Vous pouvez utiliser la CLI pour spécifier des balises lorsque vous créez un groupe de packages ou lorsque vous ajoutez, supprimez ou mettez à jour la valeur des balises d'un groupe de packages existant.

Groupes de packages de balises (CLI)

Vous pouvez utiliser la CLI pour gérer les balises de groupes de packages.

Si ce n'est pas le cas, configurez le AWS CLI en suivant les étapes décrites dans [Configuration avec AWS CodeArtifact](#).

Tip

Pour ajouter des balises, vous devez fournir le nom de ressource Amazon (ARN) du groupe de packages. Pour obtenir l'ARN du groupe de packages, exécutez la `describe-package-group` commande suivante :

```
aws codeartifact describe-package-group \
  --domain my_domain \
  --package-group /npm/scope/anycompany~ \
  --query packageGroup.arn
```

Rubriques

- [Ajouter des balises à un groupe de packages \(CLI\)](#)
- [Afficher les balises d'un groupe de packages \(CLI\)](#)
- [Modifier les balises d'un groupe de packages \(CLI\)](#)
- [Supprimer les balises d'un groupe de packages \(CLI\)](#)

Ajouter des balises à un groupe de packages (CLI)

Vous pouvez ajouter des balises aux groupes de packages lors de leur création ou à un groupe de packages existant. Pour plus d'informations sur l'ajout de balises à un groupe de packages lors de sa création, consultez[Création d'un groupe de packages](#).

Pour ajouter une balise à un groupe de packages existant à l' AWS CLI aide du terminal ou de la ligne de commande, exécutez la tag-resource commande en spécifiant le nom de ressource Amazon (ARN) du groupe de packages dans lequel vous souhaitez ajouter des balises, ainsi que la clé et la valeur de la balise que vous souhaitez ajouter. Pour plus d'informations sur le groupe de packages ARNs, consultez[Groupe de packages ARNs](#).

Vous pouvez ajouter plusieurs balises à un groupe de packages. Par exemple, pour étiqueter un groupe de packages */npm/scope/anycompany~* avec deux balises, une clé de balise nommée *key1* avec la valeur de *value1* balise et une clé de balise nommée *key2* avec la valeur de balise de *value2* :

```
aws codeartifact tag-resource \
  --resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
  group/my_domain/npm/scope/anycompany~ \
  --tags key=key1,value=value1 key=key2,value=value2
```

En cas de succès, cette commande n'a aucune sortie.

Afficher les balises d'un groupe de packages (CLI)

Procédez comme suit pour utiliser le AWS CLI afin d'afficher les AWS balises d'un groupe de packages. Si aucune balise n'a été ajoutée, la liste renvoyée est vide.

Sur le terminal ou sur la ligne de commande, exécutez la list-tags-for-resource commande avec l'Amazon Resource Name (ARN) du groupe de packages. Pour plus d'informations sur le groupe de packages ARNs, consultez[Groupe de packages ARNs](#).

Par exemple, pour afficher la liste des clés de balise et des valeurs de balise pour un groupe de packages, `/npm/scope/anycompany~` nommé avec une valeur ARN de `arn:aws:codeartifact:us-west-2:123456789012:package-group/my_domain/npm/ scope/anycompany~`

```
aws codeartifact list-tags-for-resource \
--resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
group/my_domain/npm/scope/anycompany~
```

Si elle aboutit, cette commande renvoie des informations similaires à ce qui suit :

```
{
  "tags": {
    "key1": "value1",
    "key2": "value2"
  }
}
```

Modifier les balises d'un groupe de packages (CLI)

Procédez comme suit pour utiliser le AWS CLI pour modifier une balise pour un groupe de packages. Vous pouvez modifier la valeur d'une clé existante ou ajouter une autre clé. Vous pouvez également supprimer des balises d'un groupe de packages, comme indiqué dans la section suivante.

Sur le terminal ou sur la ligne de commande, exécutez la tag-resource commande en spécifiant l'ARN du groupe de packages dans lequel vous souhaitez mettre à jour une balise, ainsi que la clé et la valeur de la balise. Pour plus d'informations sur le groupe de packages ARNs, consultez[Groupe de packages ARNs](#).

```
aws codeartifact tag-resource \
--resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
group/my_domain/npm/scope/anycompany~ \
--tags key=key1,value=newvalue1
```

En cas de succès, cette commande n'a aucune sortie.

Supprimer les balises d'un groupe de packages (CLI)

Procédez comme suit pour utiliser le AWS CLI pour supprimer une balise d'un groupe de packages.

Note

Si vous supprimez un groupe de packages, toutes les associations de balises sont supprimées du groupe de packages supprimé. Il n'est pas nécessaire de supprimer les balises avant de supprimer un groupe de packages.

Sur le terminal ou sur la ligne de commande, exécutez la `untag-resource` commande en spécifiant l'ARN du groupe de packages dans lequel vous souhaitez supprimer les balises et la clé de balise de la balise que vous souhaitez supprimer. Pour plus d'informations sur le groupe de packages ARNs, consultez [Groupe de packages ARNs](#).

Par exemple, pour supprimer plusieurs balises d'un groupe de packages `/npm/scope/anycompany~`, à l'aide des clés de balise `key1` et `key2` :

```
aws codeartifact untag-resource \
  --resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
  group/my_domain/npm/scope/anycompany~ \
  --tag-keys key1 key2
```

En cas de succès, cette commande n'a aucune sortie. Après avoir supprimé les balises, vous pouvez afficher les balises restantes du groupe de packages à l'aide de la `list-tags-for-resource` commande.

Travailler avec des domaines dans CodeArtifact

CodeArtifact les domaines facilitent la gestion de plusieurs référentiels au sein d'une organisation.

Vous pouvez utiliser un domaine pour appliquer des autorisations à de nombreux référentiels appartenant à différents comptes AWS. Un actif n'est stocké qu'une seule fois dans un domaine, même s'il est disponible dans plusieurs référentiels.

Bien que vous puissiez avoir plusieurs domaines, nous recommandons un seul domaine de production contenant tous les artefacts publiés afin que vos équipes de développement puissent trouver et partager des packages. Vous pouvez utiliser un deuxième domaine de préproduction pour tester les modifications apportées à la configuration du domaine de production.

Ces rubriques décrivent comment utiliser la CodeArtifact console AWS CLI, et comment CloudFormation créer ou configurer des CodeArtifact domaines.

Rubriques

- [Vue d'ensemble du domaine](#)
- [Création d'un domaine](#)
- [Supprimer un domaine](#)
- [Politiques de domaine](#)
- [Marquer un domaine dans CodeArtifact](#)

Vue d'ensemble du domaine

Lorsque vous travaillez avec CodeArtifact, les domaines sont utiles pour les raisons suivantes :

- Stockage dédupliqué : un actif ne doit être stocké qu'une seule fois dans un domaine, même s'il est disponible dans 1 ou 1 000 référentiels. Cela signifie que vous ne payez qu'une seule fois pour le stockage.
- Copie rapide : lorsque vous extrayez des packages d'un CodeArtifact référentiel en amont vers un référentiel en aval ou que vous utilisez l'[CopyPackageVersions API](#), seuls les enregistrements de métadonnées doivent être mis à jour. Aucune ressource n'est copiée. Cela permet de configurer rapidement un nouveau référentiel à des fins de test ou de test. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).

- Partage aisement entre les référentiels et les équipes : tous les actifs et métadonnées d'un domaine sont chiffrés à l'aide d'une seule clé AWS KMS key (clé KMS). Il n'est pas nécessaire de gérer une clé pour chaque référentiel ni d'accorder à plusieurs comptes l'accès à une seule clé.
- Appliquer une politique à plusieurs référentiels : l'administrateur du domaine peut appliquer une politique à l'ensemble du domaine. Cela inclut la restriction des comptes ayant accès aux référentiels du domaine et des personnes autorisées à configurer les connexions aux référentiels publics à utiliser comme sources de packages. Pour plus d'informations, consultez la section [Politiques de domaine](#).
- Noms de référentiels uniques : le domaine fournit un espace de noms pour les référentiels. Les noms de référentiels doivent uniquement être uniques au sein du domaine. Vous devez utiliser des noms significatifs et faciles à comprendre.

Les noms de domaine doivent être uniques au sein d'un compte.

Vous ne pouvez pas créer de référentiel sans domaine. Lorsque vous utilisez l'[CreateRepositoryAPI](#) pour créer un référentiel, vous devez spécifier un nom de domaine. Vous ne pouvez pas déplacer un dépôt d'un domaine vers un autre.

Un référentiel peut appartenir au même AWS compte qui possède le domaine, ou à un autre compte. Si les comptes propriétaires sont différents, le compte propriétaire du référentiel doit être `CreateRepository` autorisé à accéder à la ressource du domaine. Vous pouvez le faire en ajoutant une politique de ressources au domaine à l'aide de la [PutDomainPermissionsPolicy](#) commande.

Bien qu'une organisation puisse avoir plusieurs domaines, il est recommandé de disposer d'un seul domaine de production contenant tous les artefacts publiés afin que les équipes de développement puissent trouver et partager des packages au sein de leur organisation. Un deuxième domaine de pré-production peut être utile pour tester les modifications apportées à la configuration du domaine de production.

Domaines multi-comptes

Les noms de domaine doivent uniquement être uniques au sein d'un compte, ce qui signifie que plusieurs domaines peuvent porter le même nom au sein d'une même région. Pour cette raison, si vous souhaitez accéder à un domaine appartenant à un compte auprès duquel vous n'êtes pas authentifié, vous devez fournir l'ID du propriétaire du domaine ainsi que le nom du domaine dans la CLI et dans la console. Consultez les exemples de CLI suivants.

Accédez à un domaine appartenant à un compte sur lequel vous êtes authentifié :

Lorsque vous accédez à un domaine dans le compte auprès duquel vous êtes authentifié, il vous suffit de spécifier le nom de domaine. L'exemple suivant répertorie les packages présents dans le *my_repo* référentiel du *my_domain* domaine appartenant à votre compte.

```
aws codeartifact list-packages --domain my_domain --repository my_repo
```

Accédez à un domaine appartenant à un compte pour lequel vous n'êtes pas authentifié :

Lorsque vous accédez à un domaine appartenant à un compte sur lequel vous n'êtes pas authentifié, vous devez spécifier le propriétaire du domaine ainsi que le nom de domaine. L'exemple suivant répertorie les packages du *other-repo* référentiel du *other-domain* domaine appartenant à un compte auprès duquel vous n'êtes pas authentifié. Notez l'ajout du `--domain-owner` paramètre.

```
aws codeartifact list-packages --domain other-domain --domain-owner 111122223333 --repository other-repo
```

Types de AWS KMS clés pris en charge dans CodeArtifact

CodeArtifact ne prend en charge que les [clés KMS symétriques](#). Vous ne pouvez pas utiliser de [clé KMS asymétrique](#) pour chiffrer vos CodeArtifact domaines. Pour plus d'informations, consultez la section [Identification des clés KMS symétriques et asymétriques](#). Pour savoir comment créer une nouvelle clé gérée par le client, consultez la section [Création de clés KMS de chiffrement symétriques](#) dans le guide du AWS Key Management Service développeur.

CodeArtifact prend en charge les stockages de clés AWS KMS externes (XKS). Vous êtes responsable de la disponibilité, de la durabilité et de la latence des opérations clés avec les clés XKS, ce qui peut affecter la disponibilité, la durabilité et la latence. Voici quelques exemples des effets de l'utilisation des touches XKS avec CodeArtifact :

- Étant donné que chaque actif d'un package demandé et toutes ses dépendances sont soumis à une latence de déchiffrement, la latence de compilation peut être considérablement augmentée en augmentant la latence des opérations XKS.
- Étant donné que tous les actifs sont chiffrés CodeArtifact, la perte des éléments clés XKS entraînera la perte de tous les actifs associés au domaine utilisant la clé XKS.

Pour plus d'informations sur les clés XKS, consultez la section Stockages de [clés externes](#) dans le manuel du AWS Key Management Service développeur.

Création d'un domaine

Vous pouvez créer un domaine à l'aide de la CodeArtifact console, du AWS Command Line Interface (AWS CLI) ou CloudFormation. Lorsque vous créez un domaine, celui-ci ne contient aucun référentiel. Pour de plus amples informations, veuillez consulter [Création d'un référentiel](#). Pour plus d'informations sur la gestion CodeArtifact des domaines avec CloudFormation, consultez[Création de CodeArtifact ressources avec AWS CloudFormation](#).

Rubriques

- [Création d'un domaine \(console\)](#)
- [Création d'un domaine \(AWS CLI\)](#)
- [Exemple de politique AWS KMS clé](#)

Création d'un domaine (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domains, puis Create domain.
3. Dans Nom, saisissez le nom de votre domaine.
4. Développez Additional configuration (Configuration supplémentaire).
5. Utilisez une AWS KMS key (clé KMS) pour chiffrer tous les actifs de votre domaine. Vous pouvez utiliser une clé KMS AWS gérée ou une clé KMS que vous gérez. Pour plus d'informations sur les types de clés KMS pris en charge dans CodeArtifact, consultez[Types de AWS KMS clés pris en charge dans CodeArtifact](#).
 - Choisissez la clé gérée par AWS si vous souhaitez utiliser la clé par défaut Clé gérée par AWS.
 - Choisissez Clé gérée par le client si vous souhaitez utiliser une clé KMS que vous gérez. Pour utiliser une clé KMS que vous gérez, dans ARN de la clé gérée par le client, recherchez et choisissez la clé KMS.

Pour plus d'informations, consultez [Clé gérée par AWS](#) la section « [Clé gérée par le client](#) » dans le guide du AWS Key Management Service développeur.

6. Choisissez Create domain (Créer un domaine).

Création d'un domaine (AWS CLI)

Pour créer un domaine avec le AWS CLI, utilisez la `create-domain` commande. Vous devez utiliser une AWS KMS key (clé KMS) pour chiffrer tous les actifs de votre domaine. Vous pouvez utiliser une clé KMS AWS gérée ou une clé KMS que vous gérez. Si vous utilisez une clé KMS AWS gérée, n'utilisez pas le `--encryption-key` paramètre.

Pour plus d'informations sur les types de clés KMS pris en charge dans CodeArtifact, consultez [Types de AWS KMS clés pris en charge dans CodeArtifact](#). Pour plus d'informations sur les clés KMS, consultez [Clé gérée par AWS](#) la section « [Clé gérée par le client](#) » dans le Guide du AWS Key Management Service développeur.

```
aws codeartifact create-domain --domain my_domain
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre nouveau domaine.

```
{  
  "domain": {  
    "name": "my_domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my_domain",  
    "status": "Active",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0,  
    "createdTime": "2020-10-12T16:51:18.039000-04:00"  
  }  
}
```

Si vous utilisez une clé KMS que vous gérez, incluez son Amazon Resource Name (ARN) dans le `--encryption-key` paramètre.

```
aws codeartifact create-domain --domain my_domain --encryption-key arn:aws:kms:us-west-2:111122223333:key/your-kms-key
```

Les données au format JSON apparaissent dans la sortie avec des informations sur votre nouveau domaine.

```
{
```

```
"domain": {  
    "name": "my_domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my_domain",  
    "status": "Active",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0,  
    "createdTime": "2020-10-12T16:51:18.039000-04:00"  
}  
}  
}
```

Créer un domaine avec des balises

Pour créer un domaine avec des balises, ajoutez le `--tags` paramètre à votre `create-domain` commande.

```
aws codeartifact create-domain --domain my_domain --tags key=k1,value=v1  
key=k2,value=v2
```

Exemple de politique AWS KMS clé

Lorsque vous créez un domaine dans CodeArtifact, vous utilisez une clé KMS pour chiffrer tous les actifs du domaine. Vous pouvez choisir une clé KMS AWS gérée ou une clé gérée par le client que vous gérez. Pour plus d'informations sur les clés KMS, consultez le [guide du AWS Key Management Service développeur](#).

Pour utiliser une clé gérée par le client, votre clé KMS doit être associée à une politique de clé autorisant l'accès à CodeArtifact. Une politique clé est une politique de ressources pour une AWS KMS clé et constitue le principal moyen de contrôler l'accès aux clés KMS. Chaque clé KMS doit avoir exactement une politique de clé. Les instructions dans la politique de clé déterminent qui a l'autorisation d'utiliser la clé KMS et la façon dont ces personnes peuvent l'utiliser.

L'exemple de déclaration de politique clé suivant permet AWS CodeArtifact de créer des autorisations et d'afficher les détails clés au nom des utilisateurs autorisés. Cette déclaration de politique limite l'autorisation d' CodeArtifact agir au nom de l'ID de compte spécifié en utilisant les clés de kms : CallerAccount condition kms : ViaService et. Il accorde également toutes les AWS KMS autorisations à l'utilisateur root IAM, afin que la clé puisse être gérée après sa création.

JSON

```
{  
    "Version": "2012-10-17",  
    "Id": "key-consolepolicy-3",  
    "Statement": [  
        {  
            "Sid": "Allow access through AWS CodeArtifact for all principals in  
            the account that are authorized to use CodeArtifact",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "kms:CreateGrant",  
                "kms:DescribeKey"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "kms:CallerAccount": "111122223333",  
                    "kms:ViaService": "codeartifact.us-west-2.amazonaws.com"  
                }  
            }  
        },  
        {  
            "Sid": "Enable IAM User Permissions",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::111122223333:root"  
            },  
            "Action": "kms:*",  
            "Resource": "*"  
        }  
    ]  
}
```

Supprimer un domaine

Vous pouvez supprimer un domaine à l'aide de la CodeArtifact console ou du AWS Command Line Interface (AWS CLI).

Rubriques

- [Restrictions relatives à la suppression de domaines](#)
- [Supprimer un domaine \(console\)](#)
- [Supprimer un domaine \(AWS CLI\)](#)

Restrictions relatives à la suppression de domaines

Normalement, vous ne pouvez pas supprimer un domaine contenant des référentiels. Avant de supprimer le domaine, vous devez d'abord supprimer ses référentiels. Pour de plus amples informations, veuillez consulter [Supprimer un dépôt](#).

Toutefois, si vous CodeArtifact n'avez plus accès à la clé KMS du domaine, vous pouvez supprimer le domaine même s'il contient toujours des référentiels. Cette situation se produira si vous supprimez la clé KMS du domaine ou si vous révoquez l'[autorisation KMS](#) CodeArtifact utilisée pour accéder à la clé. Dans cet état, vous ne pouvez pas accéder aux référentiels du domaine ni aux packages qui y sont stockés. Il est également impossible de répertorier et de supprimer des référentiels lorsque vous CodeArtifact ne pouvez pas accéder à la clé KMS du domaine. Pour cette raison, la suppression du domaine ne permet pas de vérifier si le domaine contient des référentiels lorsque la clé KMS du domaine est inaccessible.

Note

Lorsqu'un domaine contenant encore des référentiels est supprimé, CodeArtifact il supprime les référentiels de manière asynchrone dans les 15 minutes. Une fois le domaine supprimé, les référentiels seront toujours visibles dans la CodeArtifact console et dans le résultat de la `list-repositories` commande jusqu'à ce que le nettoyage automatique des référentiels soit effectué.

Supprimer un domaine (console)

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Domaines, puis le domaine que vous souhaitez supprimer.
3. Sélectionnez Delete (Supprimer).

Supprimer un domaine (AWS CLI)

Utilisez la `delete-domain` commande pour supprimer un domaine.

```
aws codeartifact delete-domain --domain my_domain --domain-owner 111122223333
```

Les données au format JSON apparaissent dans la sortie avec des informations sur le domaine supprimé.

```
{  
  "domain": {  
    "name": "my_domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my_domain",  
    "status": "Active",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0,  
    "createdTime": "2020-10-12T16:51:18.039000-04:00"  
  }  
}
```

Politiques de domaine

CodeArtifact prend en charge l'utilisation d'autorisations basées sur les ressources pour contrôler l'accès. Les autorisations basées sur les ressources vous permettent de spécifier qui a accès à une ressource et quelles actions ils peuvent effectuer sur celle-ci. Par défaut, seul le compte AWS propriétaire du domaine peut créer des référentiels dans le domaine et y accéder. Vous pouvez appliquer un document de politique à un domaine pour permettre à d'autres principaux IAM d'y accéder.

Pour plus d'informations, consultez [les sections Politiques et autorisations et Stratégies basées sur l'identité et Stratégies basées sur les ressources](#).

Rubriques

- [Activer l'accès multicompte à un domaine](#)
- [Exemple de politique de domaine](#)
- [Exemple de politique de domaine avec AWS Organizations](#)

- [Définissez une politique de domaine](#)
- [Lire une politique de domaine](#)
- [Supprimer une politique de domaine](#)

Activer l'accès multicompte à un domaine

Une politique de ressources est un fichier texte au format JSON. Le fichier doit spécifier un principal (acteur), une ou plusieurs actions et un effet (AllowouDeny). Pour créer un référentiel dans un domaine appartenant à un autre compte, le principal doit CreateRepository obtenir l'autorisation sur la ressource du domaine.

Par exemple, la politique de ressources suivante accorde au compte 123456789012 l'autorisation de créer un référentiel dans le domaine.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:CreateRepository"  
      ],  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      },  
      "Resource": "*"  
    }  
  ]  
}
```

Pour autoriser la création de référentiels avec des balises, vous devez inclure l'codeartifact:TagResource autorisation. Cela permettra également au compte d'ajouter des balises au domaine et à tous les référentiels qu'il contient.

La politique de domaine est évaluée pour toutes les opérations effectuées sur le domaine et pour toutes les ressources du domaine. Cela signifie que la politique de domaine peut être

utilisée pour appliquer des autorisations aux référentiels et aux packages du domaine. Lorsque l'Resourceélément est défini sur*, l'instruction s'applique à toutes les ressources du domaine. Par exemple, si la politique ci-dessus est également incluse `codeartifact:DescribeRepository` dans la liste des actions IAM autorisées, elle autorisera l'appel à tous `DescribeRepository` les référentiels du domaine. Une politique de domaine peut être utilisée pour appliquer des autorisations à des ressources spécifiques du domaine en utilisant une ressource spécifique ARNs dans l'Resourceélément.

Note

Les politiques de domaine et de référentiel peuvent être utilisées pour configurer les autorisations. Lorsque les deux politiques sont présentes, les deux politiques sont évaluées et une action est autorisée si l'une ou l'autre politique l'autorise. Pour de plus amples informations, veuillez consulter [Interaction entre le référentiel et les politiques de domaine](#).

Pour accéder aux packages d'un domaine appartenant à un autre compte, un principal doit `GetAuthorizationToken` obtenir l'autorisation sur la ressource du domaine. Cela permet au propriétaire du domaine de contrôler les comptes autorisés à lire le contenu des référentiels du domaine.

Par exemple, la politique de ressources suivante accorde au compte l'123456789012 autorisation de récupérer un jeton d'authentification pour n'importe quel référentiel du domaine.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:GetAuthorizationToken"  
      ],  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      },  
      "Resource": "*"  
    }  
  ]  
}
```

```
]  
}
```

Note

Un principal qui souhaite récupérer des packages depuis un point de terminaison du référentiel doit `ReadFromRepository` obtenir l'autorisation sur la ressource du référentiel en plus de l'`GetAuthorizationToken` autorisation sur le domaine. De même, un directeur qui souhaite publier des packages sur un point de terminaison du référentiel doit `PublishPackageVersion` obtenir l'autorisation en plus de `GetAuthorizationToken`. Pour plus d'informations sur les `PublishPackageVersion` autorisations `ReadFromRepository` et, consultez la section [Politiques du référentiel](#).

Exemple de politique de domaine

Lorsque plusieurs comptes utilisent un domaine, les comptes doivent bénéficier d'un ensemble d'autorisations de base pour permettre l'utilisation complète du domaine. La politique de ressources suivante répertorie un ensemble d'autorisations qui permettent l'utilisation complète du domaine.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "BasicDomainPolicy",  
      "Action": [  
        "codeartifact:GetDomainPermissionsPolicy",  
        "codeartifact>ListRepositoriesInDomain",  
        "codeartifact:GetAuthorizationToken",  
        "codeartifact:DescribeDomain",  
        "codeartifact>CreateRepository"  
      ],  
      "Effect": "Allow",  
      "Resource": "*",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      }  
    }  
  ]  
}
```

```
    }  
]  
}
```

Note

Il n'est pas nécessaire de créer une politique de domaine si un domaine et tous ses référentiels appartiennent à un seul compte et ne doivent être utilisés qu'à partir de ce compte.

Exemple de politique de domaine avec AWS Organizations

Vous pouvez utiliser la clé de `aws:PrincipalOrgID` condition pour accorder l'accès à un CodeArtifact domaine à partir de tous les comptes de votre organisation, comme suit.

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Sid": "DomainPolicyForOrganization",  
        "Effect": "Allow",  
        "Principal": "*",  
        "Action": [  
            "codeartifact:GetDomainPermissionsPolicy",  
            "codeartifact>ListRepositoriesInDomain",  
            "codeartifact:GetAuthorizationToken",  
            "codeartifact:DescribeDomain",  
            "codeartifact>CreateRepository"  
        ],  
        "Resource": "*",  
        "Condition": {  
            "StringEquals": { "aws:PrincipalOrgID": ["o-xxxxxxxxxxxx"] }  
        }  
    }  
}
```

Pour plus d'informations sur l'utilisation de la clé de aws :PrincipalOrgID condition, consultez [AWS Global Condition Context Keys](#) dans le guide de l'utilisateur IAM.

Définissez une politique de domaine

Vous pouvez utiliser la `put-domain-permissions-policy` commande pour associer une politique à un domaine.

```
aws codeartifact put-domain-permissions-policy --domain my_domain --domain-owner 111122223333 \  
--policy-document file://</PATH/TO/policy.json>
```

Lorsque vous appelez `put-domains-permissions-policy`, la politique de ressources du domaine est ignorée lors de l'évaluation des autorisations. Cela garantit que le propriétaire d'un domaine ne peut pas s'empêcher d'accéder au domaine, ce qui l'empêcherait de mettre à jour la politique de ressources.

Note

Vous ne pouvez pas autoriser un autre AWS compte à mettre à jour la politique de ressources d'un domaine à l'aide d'une stratégie de ressources, car la politique de ressources est ignorée lors de l'appel `put-domain-permissions-policy`.

Exemple de sortie :

```
{  
  "policy": {  
    "resourceArn": "arn:aws:codeartifact:region-id:111122223333:domain/my_domain",  
    "document": "{ ...policy document content... }",  
    "revision": "MQ1yyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxxx"  
  }  
}
```

La sortie de la commande contient l'Amazon Resource Name (ARN) de la ressource de domaine, le contenu complet du document de politique et un identifiant de révision. L'identifiant de révision peut être transmis à `put-domain-permissions-policy` l'aide de l'`--policy-revision` option. Cela garantit qu'une révision connue du document est remplacée, et non une version plus récente définie par un autre rédacteur.

Lire une politique de domaine

Pour lire une version existante d'un document de politique, utilisez la `get-domain-permissions-policy` commande. Pour formater la sortie à des fins de lisibilité, utilisez le module `--output` et `--query` `policy.document` avec le `json.tool` module Python, comme suit.

```
aws codeartifact get-domain-permissions-policy --domain my_domain --domain-owner 111122223333 \
  --output text --query policy.document | python -m json.tool
```

Exemple de sortie :

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "BasicDomainPolicy",  
      "Action": [  
        "codeartifact:GetDomainPermissionsPolicy",  
        "codeartifact>ListRepositoriesInDomain",  
        "codeartifact:GetAuthorizationToken",  
        "codeartifact>CreateRepository"  
      ],  
      "Effect": "Allow",  
      "Resource": "*",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333:root"  
      }  
    }  
  ]  
}
```

Supprimer une politique de domaine

Utilisez la `delete-domain-permissions-policy` commande pour supprimer une politique d'un domaine.

```
aws codeartifact delete-domain-permissions-policy --domain my_domain --domain-owner 111122223333
```

Le format de sortie est identique à celui des `delete-domain-permissions-policy` commandes `get-domain-permissions-policy` et.

Marquer un domaine dans CodeArtifact

Les balises sont des paires clé-valeur associées aux ressources AWS. Vous pouvez appliquer des balises à vos domaines dans CodeArtifact. Pour plus d'informations sur le balisage CodeArtifact des ressources, les cas d'utilisation, les contraintes de clé et de valeur de balise et les types de ressources pris en charge, consultez[Balisage des ressources](#).

Vous pouvez utiliser la CLI pour spécifier des balises lorsque vous créez un domaine. Vous pouvez utiliser la console ou la CLI pour ajouter ou supprimer des balises et mettre à jour les valeurs des balises dans un domaine. Vous pouvez ajouter jusqu'à 50 balises à chaque domaine.

Rubriques

- [Domaines de balises \(CLI\)](#)
- [Domaines de balises \(console\)](#)

Domaines de balises (CLI)

Vous pouvez utiliser la CLI pour gérer les balises de domaine.

Rubriques

- [Ajouter des balises à un domaine \(CLI\)](#)
- [Afficher les balises d'un domaine \(CLI\)](#)
- [Modifier les balises d'un domaine \(CLI\)](#)
- [Supprimer les balises d'un domaine \(CLI\)](#)

Ajouter des balises à un domaine (CLI)

Vous pouvez utiliser la console ou le AWS CLI pour baliser des domaines.

Pour ajouter une balise à un domaine lorsque vous le créez, consultez[Création d'un référentiel](#).

Dans ces étapes, nous supposons que vous avez déjà installé une version récente de l' AWS CLI ou que vous avez procédé à une mise à jour vers la version actuelle. Pour plus d'informations, consultez [Installing the AWS Command Line Interface](#) (Installation de).

Sur le terminal ou sur la ligne de commande, exécutez la tag-resource commande en spécifiant le nom de ressource Amazon (ARN) du domaine dans lequel vous souhaitez ajouter des balises, ainsi que la clé et la valeur de la balise que vous souhaitez ajouter.

Note

Pour obtenir l'ARN du domaine, exécutez la describe-domain commande suivante :

```
aws codeartifact describe-domain --domain my_domain --query domain.arn
```

Vous pouvez ajouter plusieurs balises à un domaine. Par exemple, pour baliser un domaine nommé *my_domain* avec deux balises, une clé de balise nommée *key1* avec la valeur de *value1* balise et une clé de balise nommée *key2* avec la valeur de balise de *value2* :

```
aws codeartifact tag-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain --tags key=key1,value=value1 key=key2,value=value2
```

En cas de succès, cette commande n'a aucune sortie.

Afficher les balises d'un domaine (CLI)

Procédez comme suit pour utiliser le AWS CLI afin d'afficher les AWS balises d'un domaine. Si aucune balise n'a été ajoutée, la liste renvoyée est vide.

Sur le terminal ou sur la ligne de commande, exécutez la list-tags-for-resource commande avec l'Amazon Resource Name (ARN) du domaine.

Note

Pour obtenir l'ARN du domaine, exécutez la describe-domain commande suivante :

```
aws codeartifact describe-domain --domain my_domain --query domain.arn
```

Par exemple, pour afficher la liste des clés de balise et des valeurs de balise pour un domaine nommé *my_domain* avec la valeur `arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain` ARN :

```
aws codeartifact list-tags-for-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain
```

Si elle aboutit, cette commande renvoie des informations similaires à ce qui suit :

```
{  
  "tags": {  
    "key1": "value1",  
    "key2": "value2"  
  }  
}
```

Modifier les balises d'un domaine (CLI)

Procédez comme suit pour utiliser le AWS CLI pour modifier une balise pour un domaine. Vous pouvez modifier la valeur d'une clé existante ou ajouter une autre clé. Vous pouvez également supprimer des balises d'un domaine, comme indiqué dans la section suivante.

Sur le terminal ou sur la ligne de commande, exécutez la `tag-resource` commande en spécifiant l'ARN du domaine dans lequel vous souhaitez mettre à jour une balise, ainsi que la clé et la valeur de la balise :

Note

Pour obtenir l'ARN du domaine, exécutez la `describe-domain` commande suivante :

```
aws codeartifact describe-domain --domain my_domain --query domain.arn
```

```
aws codeartifact tag-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain --tags key=key1,value=newvalue1
```

En cas de succès, cette commande n'a aucune sortie.

Supprimer les balises d'un domaine (CLI)

Procédez comme suit pour utiliser le AWS CLI pour supprimer une balise d'un domaine.

Note

Si vous supprimez un domaine, toutes les associations de balises sont supprimées du domaine supprimé. Il n'est pas nécessaire de supprimer les balises avant de supprimer un domaine.

Sur le terminal ou sur la ligne de commande, exécutez la `untag-resource` commande en spécifiant l'ARN du domaine dans lequel vous souhaitez supprimer les balises et la clé de balise de la balise que vous souhaitez supprimer.

Note

Pour obtenir l'ARN du domaine, exécutez la `describe-domain` commande suivante :

```
aws codeartifact describe-domain --domain my_domain --query domain.arn
```

Par exemple, pour supprimer plusieurs balises sur un domaine nommé à l'*mydomain* aide des clés de balise *key1* et *key2* :

```
aws codeartifact untag-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain --tag-keys key1 key2
```

En cas de succès, cette commande n'a aucune sortie. Après avoir supprimé les balises, vous pouvez afficher les balises restantes sur le domaine à l'aide de la `list-tags-for-resource` commande.

Domaines de balises (console)

Vous pouvez utiliser la console ou la CLI pour baliser des ressources.

Rubriques

- [Ajouter des balises à un domaine \(console\)](#)

- [Afficher les balises d'un domaine \(console\)](#)
- [Modifier les balises d'un domaine \(console\)](#)
- [Supprimer les balises d'un domaine \(console\)](#)

Ajouter des balises à un domaine (console)

Vous pouvez utiliser la console pour ajouter des balises à un domaine existant.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Domaines, choisissez le domaine auquel vous souhaitez ajouter des balises.
3. Développez la section Détails.
4. Sous Balises de domaine, choisissez Ajouter des balises de domaine s'il n'y en a aucune sur le domaine, ou choisissez Afficher et modifier les balises de domaine s'il y en a.
5. Sélectionnez Ajouter une nouvelle balise.
6. Dans les champs Clé et Valeur, entrez le texte de chaque balise que vous souhaitez ajouter. (Le champ Valeur est facultatif.) Par exemple, dans Clé, saisissez **Name**. Dans Value (Valeur), entrez **Test**.

The screenshot shows the 'Edit domainname' page in the AWS CodeArtifact console. The URL in the browser is: Developer Tools > CodeArtifact > Domains > domainname > Edit domain. The page title is 'Edit domainname' with an 'Info' link. A 'Tags' section is highlighted, showing a table for adding new tags. The table has 'Key' and 'Value' fields, both with search icons and 'X' buttons. A 'Remove' button is also present. Below the table is a button labeled 'Add new tag'. A note says 'You can add 49 more tags.' At the bottom of the 'Tags' section is a section for 'AWS reserved tags' with a note: 'Resource tags added by other AWS services. These tags cannot be modified.' At the bottom right of the page are 'Cancel' and 'Update domain' buttons.

7. (Facultatif) Choisissez Add tag (Ajouter une balise) pour ajouter d'autres lignes et saisir d'autres balises.
8. Choisissez Mettre à jour le domaine.

Afficher les balises d'un domaine (console)

Vous pouvez utiliser la console pour répertorier les balises des domaines existants.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Domaines, choisissez le domaine dans lequel vous souhaitez afficher les balises.
3. Développez la section Détails.
4. Sous Balises de domaine, choisissez Afficher et modifier les balises de domaine.

 Note

Si aucune balise n'est ajoutée à ce domaine, la console affichera Ajouter des balises de domaine.

Modifier les balises d'un domaine (console)

Vous pouvez utiliser la console pour modifier les balises qui ont été ajoutées au domaine.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Domaines, choisissez le domaine dans lequel vous souhaitez mettre à jour les balises.
3. Développez la section Détails.
4. Sous Balises de domaine, choisissez Afficher et modifier les balises de domaine.

 Note

Si aucune balise n'est ajoutée à ce domaine, la console affichera Ajouter des balises de domaine.

5. Dans les champs Clé et Valeur, mettez à jour les valeurs dans chaque champ selon vos besoins. Par exemple, pour la clé **Name**, dans Valeur, remplacez **Test** par **Prod**.
6. Choisissez Mettre à jour le domaine.

Supprimer les balises d'un domaine (console)

Vous pouvez utiliser la console pour supprimer des balises de domaines.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Sur la page Domaines, choisissez le domaine dans lequel vous souhaitez supprimer les balises.
3. Développez la section Détails.
4. Sous Balises de domaine, choisissez Afficher et modifier les balises de domaine.

 Note

Si aucune balise n'est ajoutée à ce domaine, la console affichera Ajouter des balises de domaine.

5. À côté de la clé et de la valeur de chaque balise que vous souhaitez supprimer, choisissez Supprimer.
6. Choisissez Mettre à jour le domaine.

Utilisation CodeArtifact avec Cargo

Ces rubriques décrivent comment utiliser Cargo, le gestionnaire de paquets de Rust, avec CodeArtifact.

Note

CodeArtifact ne prend en charge que Cargo 1.74.0 et versions ultérieures. Cargo 1.74.0 est la première version qui prend en charge l'authentification sur un CodeArtifact référentiel.

Rubriques

- [Configurer et utiliser Cargo avec CodeArtifact](#)
- [Assistance au commandement du fret](#)

Configurer et utiliser Cargo avec CodeArtifact

Vous pouvez utiliser Cargo pour publier et télécharger des caisses depuis CodeArtifact des référentiels ou pour récupérer des caisses depuis [crates.io](#), le registre des caisses de la communauté Rust. Cette rubrique décrit comment configurer Cargo pour s'authentifier auprès d'un CodeArtifact référentiel et l'utiliser.

Configurez Cargo avec CodeArtifact

Pour utiliser Cargo pour installer et publier des caisses à partir de celles-ci AWS CodeArtifact, vous devez d'abord les configurer avec les informations de votre CodeArtifact référentiel. Suivez les étapes de l'une des procédures suivantes pour configurer Cargo avec les informations et les informations d'identification du point de terminaison de votre CodeArtifact référentiel.

Configurer Cargo à l'aide des instructions de la console

Vous pouvez utiliser les instructions de configuration de la console pour connecter Cargo à votre CodeArtifact référentiel. Les instructions de la console fournissent une configuration Cargo personnalisée pour votre CodeArtifact référentiel. Vous pouvez utiliser cette configuration personnalisée pour configurer Cargo sans avoir à rechercher et à renseigner vos CodeArtifact informations.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Repositories, puis choisissez un référentiel pour vous connecter à Cargo.
3. Choisissez Afficher les instructions de connexion.
4. Choisissez votre système d'exploitation.
5. Choisissez Cargo.
6. Suivez les instructions générées pour connecter Cargo à votre CodeArtifact dépôt.

Configurer Cargo manuellement

Si vous ne pouvez pas ou ne souhaitez pas utiliser les instructions de configuration de la console, vous pouvez utiliser les instructions suivantes pour connecter Cargo à votre CodeArtifact référentiel manuellement.

macOS and Linux

Pour configurer Cargo avec CodeArtifact, vous devez définir votre CodeArtifact référentiel en tant que registre dans la configuration de Cargo et fournir des informations d'identification.

- *my_registry*Remplacez-le par votre nom de registre.
- *my_domain*Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez **111122223333** par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure--domain-owner. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo*Remplacez-le par le nom de votre CodeArtifact dépôt.

Copiez la configuration pour publier et télécharger les packages Cargo dans votre référentiel et enregistrez-la dans le `~/.cargo/config.toml` fichier pour une configuration au niveau du système ou `.cargo/config.toml` pour une configuration au niveau du projet :

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/cargo/my_repo/"
```

```
credential-provider = "cargo:token-from-stdout aws codeartifact get-authorization-  
token --domain my_domain --domain-owner 111122223333 --region us-west-2 --query  
authorizationToken --output text"  
  
[registry]  
default = "my_registry"  
  
[source.crates-io]  
replace-with = "my_registry"
```

Windows: Download packages only

Pour configurer Cargo avec CodeArtifact, vous devez définir votre CodeArtifact référentiel en tant que registre dans la configuration de Cargo et fournir des informations d'identification.

- *my_registry* Remplacez-le par votre nom de registre.
- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

Copiez la configuration pour télécharger uniquement les packages Cargo depuis votre référentiel et enregistrez-la dans le `%USERPROFILE%\.cargo\config.toml` fichier pour une configuration au niveau du système ou `.cargo\config.toml` pour une configuration au niveau du projet :

```
[registries.my_registry]  
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/cargo/my_repo/"  
credential-provider = "cargo:token-from-stdout aws codeartifact get-authorization-  
token --domain my_domain --domain-owner 111122223333 --region us-west-2 --query  
authorizationToken --output text"  
  
[registry]  
default = "my_registry"  
  
[source.crates-io]  
replace-with = "my_registry"
```

Windows: Publish and download packages

1. Pour configurer Cargo avec CodeArtifact, vous devez définir votre CodeArtifact référentiel en tant que registre dans la configuration de Cargo et fournir des informations d'identification.
 - *my_registry* Remplacez-le par votre nom de registre.
 - *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
 - Remplacez **111122223333** par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
 - *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

Copiez la configuration pour publier et télécharger les packages Cargo dans votre référentiel et enregistrez-la dans le `%USERPROFILE%\.cargo\config.toml` fichier pour une configuration au niveau du système ou `.cargo\config.toml` pour une configuration au niveau du projet.

Il est recommandé d'utiliser le fournisseur d'informations d'identification `cargo:token`, qui utilise les informations d'identification stockées dans votre `~/.cargo/credentials.toml` fichier. Vous pouvez rencontrer une erreur lors `cargo publish` de l'utilisation, `cargo:token-from-stdout` car le client Cargo ne découpe pas correctement le jeton d'autorisation pendant cette utilisation `cargo publish`.

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/cargo/my_repo/"
credential-provider = "cargo:token"

[registry]
default = "my_registry"

[source.crates-io]
replace-with = "my_registry"
```

2. Pour publier des packages Cargo dans votre référentiel avec Windows, vous devez utiliser la `CodeArtifact get-authorization-token` commande et la `login` commande Cargo pour récupérer un jeton d'autorisation et vos informations d'identification.

- *my_registry* Remplacez-le par votre nom de registre tel que défini dans [registries.*my_registry*].
- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure --domain-owner. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

```
aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --region us-west-2 --query authorizationToken --output text | cargo login --registry my_registry
```

 Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

La [registries.*my_registry*] section de l'exemple précédent définit un registre contenant *my_registry* index et fournit des credential-provider informations.

- index spécifie l'URL de l'index de votre registre, qui est le point de terminaison du CodeArtifact référentiel qui se termine par un /. Le sparse+ préfixe est obligatoire pour les registres qui ne sont pas des référentiels Git.

 Note

Pour utiliser un point de terminaison à double pile, utilisez le codeartifact.*region*.on.aws point de terminaison.

- credential-provider spécifie le fournisseur d'informations d'identification pour le registre donné. Si credential-provider ce n'est pas le cas, les fournisseurs registry.global-credential-providers inclus seront utilisés. En le credential-provider réglant sur cargo:token-from-stdout, le client Cargo récupérera automatiquement le nouveau jeton d'autorisation lors de la publication ou du téléchargement depuis votre CodeArtifact référentiel.

Vous n'avez donc pas besoin d'actualiser manuellement le jeton d'autorisation toutes les 12 heures.

La [registry] section définit le registre par défaut utilisé.

- default indique le nom du registre défini dans [registries.*my_registry*], à utiliser par défaut lors de la publication ou du téléchargement depuis votre CodeArtifact référentiel.

La [source.crates-io] section définit le registre par défaut utilisé lorsqu'aucun registre n'est spécifié.

- replace-with = "*my_registry*" remplace le registre public crates.io par votre CodeArtifact dépôt défini dans. [registries.*my_registry*] Cette configuration est recommandée si vous devez demander des packages via une connexion externe telle que crates.io.

Pour bénéficier de tous les avantages CodeArtifact, tels que le contrôle de l'origine des packages qui empêche les attaques de confusion de dépendances, il est recommandé d'utiliser le remplacement de source. Avec le remplacement de la source, CodeArtifact transmet toutes les demandes par proxy à la connexion externe et copie le package depuis la connexion externe vers votre référentiel. Sans le remplacement de la source, le client Cargo récupérera directement le package en fonction de la configuration de votre Cargo.toml fichier dans votre projet. Si une dépendance n'est pas marquée par `unregistry=my_registry`, le client Cargo la récupérera directement depuis crates.io sans communiquer avec votre CodeArtifact dépôt.

 Note

Si vous commencez à utiliser le remplacement de source, puis que vous mettez à jour votre fichier de configuration pour ne pas utiliser le remplacement de source, vous risquez de rencontrer des erreurs. Le scénario inverse peut également entraîner des erreurs. Il est donc recommandé d'éviter de modifier la configuration de votre projet.

Installation de caisses de chargement

Utilisez les procédures suivantes pour installer les caisses Cargo depuis un CodeArtifact dépôt ou depuis [crates.io](#).

Installez les caisses Cargo à partir de CodeArtifact

Vous pouvez utiliser la CLI Cargo (cargo) pour installer rapidement une version spécifique d'une caisse Cargo à partir de votre CodeArtifact référentiel.

Pour installer des caisses Cargo à partir d'un CodeArtifact référentiel avec **cargo**

1. Si ce n'est pas le cas, suivez les étapes décrites [Configurer et utiliser Cargo avec CodeArtifact](#) pour configurer la cargo CLI afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.
2. Utilisez la commande suivante pour installer les caisses Cargo à partir de CodeArtifact :

```
cargo add my_cargo_package@1.0.0
```

Pour plus d'informations, voir [Ajouter une cargaison](#) dans The Cargo Book.

Publication de caisses Cargo sur CodeArtifact

Utilisez la procédure suivante pour publier des caisses Cargo dans un CodeArtifact référentiel à l'aide de la cargo CLI.

1. Si ce n'est pas le cas, suivez les étapes décrites [Configurer et utiliser Cargo avec CodeArtifact](#) pour configurer la cargo CLI afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.
2. Utilisez la commande suivante pour publier les caisses Cargo dans un CodeArtifact référentiel :

```
cargo publish
```

Pour plus d'informations, consultez la section [Cargo Publish](#) dans The Cargo Book.

Assistance au commandement du fret

Les sections suivantes résument les commandes Cargo prises en charge par les CodeArtifact référentiels, ainsi que les commandes spécifiques qui ne sont pas prises en charge.

Table des matières

- [Commandes prises en charge qui nécessitent l'accès au registre](#)

- [Commandes non prises en charge](#)

Commandes prises en charge qui nécessitent l'accès au registre

Cette section répertorie les commandes Cargo pour lesquelles le client Cargo doit accéder au registre avec lequel il a été configuré. Il a été vérifié que ces commandes fonctionnent correctement lorsqu'elles sont invoquées dans un CodeArtifact dépôt.

Command	Description
construire	Construit des packages locaux et leurs dépendances.
vérifier	Vérifie la présence d'erreurs dans les packages locaux et leurs dépendances.
aller chercher	Récupère les dépendances d'un package.
publier	Publie un package dans le registre.

Commandes non prises en charge

Ces commandes Cargo ne sont pas prises en charge par les CodeArtifact référentiels.

Command	Description
propriétaire	Gère les propriétaires de la caisse sur le registre.
search	Recherche des packages dans le registre.

Utilisation CodeArtifact avec Maven

Le format de dépôt Maven est utilisé par de nombreux langages, notamment Java, Kotlin, Scala et Clojure. Il est pris en charge par de nombreux outils de construction différents, notamment Maven, Gradle, Scala SBT, Apache Ivy et Leiningen.

Nous avons testé et confirmé la compatibilité avec CodeArtifact les versions suivantes :

- Dernière version de Maven : 3.6.3.
- Dernière version de Gradle : 6.4.1. 5.5.1 a également été testée.
- La dernière version de Clojure : 1.11.1 a également été testée.

Rubriques

- [Utiliser CodeArtifact avec Gradle](#)
- [Utiliser CodeArtifact avec MVN](#)
- [À utiliser CodeArtifact avec deps.edn](#)
- [Publier avec curl](#)
- [Utiliser les checksums Maven](#)
- [Utiliser les instantanés Maven](#)
- [Demande de packages Maven depuis des connexions en amont et externes](#)
- [Résolution des problèmes liés à Maven](#)

Utiliser CodeArtifact avec Gradle

Une fois que vous avez le jeton d' CodeArtifact authentification dans une variable d'environnement, comme décrit dans [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#), suivez ces instructions pour utiliser les packages Maven depuis un référentiel et y publier de nouveaux packages. CodeArtifact

Rubriques

- [Récupérer les dépendances](#)
- [Plug-ins de récupération](#)
- [Publier des artefacts](#)

- [Exécuter une compilation Gradle dans IntelliJ IDEA](#)

Récupérer les dépendances

Pour récupérer les dépendances CodeArtifact dans une version Gradle, utilisez la procédure suivante.

Pour récupérer les dépendances CodeArtifact dans une version Gradle

1. Si ce n'est pas le cas, créez et stockez un jeton d'authentification dans une variable d'environnement en suivant la procédure décrite dans [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#).
2. Ajoutez une `maven` section à la `repositories` section du `build.gradle` fichier de projet.

```
maven {  
    url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/  
    maven/my_repo/'  
    credentials {  
        username "aws"  
        password System.env.CODEARTIFACT_AUTH_TOKEN  
    }  
}
```

Dans l'exemple précédent, il s'agit du point de terminaison de votre CodeArtifact dépôt. Gradle utilise le point de terminaison pour se connecter à votre référentiel. Dans l'exemple, `my_domain` il s'agit du nom de votre domaine, `111122223333` de l'identifiant du propriétaire du domaine et `my_repo` du nom de votre référentiel. Vous pouvez récupérer le point de terminaison d'un dépôt à l'aide de la `get-repository-endpoint` AWS CLI commande.

Par exemple, avec un dépôt nommé `my_repo` dans un domaine nommé `my_domain`, la commande est la suivante :

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-  
owner 111122223333 --repository my_repo --format maven
```

La `get-repository-endpoint` commande renverra le point de terminaison du référentiel :

```
url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/  
maven/my_repo/'
```

L'credentials objet de l'exemple précédent inclut le jeton d' CodeArtifact authentification que vous avez créé à l'étape 1 et que Gradle utilise pour s'authentifier. CodeArtifact

Note

Pour utiliser un point de terminaison à double pile, utilisez le codeartifact.*region*.on.aws point de terminaison.

3. (Facultatif) - Pour utiliser le CodeArtifact référentiel comme seule source pour les dépendances de votre projet, supprimez toutes les autres sections repositories de build.gradle. Si vous avez plusieurs référentiels, Gradle recherche les dépendances dans chaque référentiel dans l'ordre dans lequel elles sont répertoriées.
4. Après avoir configuré le référentiel, vous pouvez ajouter des dépendances de projet à la dependencies section avec la syntaxe Gradle standard.

```
dependencies {  
    implementation 'com.google.guava:guava:27.1-jre'  
    implementation 'commons-cli:commons-cli:1.4'  
    testImplementation 'org.testng:testng:6.14.3'  
}
```

Plug-ins de récupération

Par défaut, Gradle résoudra les plugins à partir du portail public des [plugins Gradle](#). Pour extraire des plugins d'un CodeArtifact dépôt, procédez comme suit.

Pour extraire des plugins d'un CodeArtifact référentiel

1. Si ce n'est pas le cas, créez et stockez un jeton d' CodeArtifact authentification dans une variable d'environnement en suivant la procédure décrite dans [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#).
2. Ajoutez un pluginManagement bloc à votre settings.gradle fichier. Le pluginManagement bloc doit apparaître avant toute autre instruction dans settings.gradle, voir l'extrait suivant :

```
pluginManagement {  
    repositories {  
        maven {  
            name 'my_repo'  
            url  
            'https://my_domain-11112222333.d.codeartifact.region.amazonaws.com/  
            maven/my_repo/'  
            credentials {  
                username 'aws'  
                password System.env.CODEARTIFACT_AUTH_TOKEN  
            }  
        }  
    }  
}
```

Cela garantira que Gradle résout les plugins à partir du référentiel spécifié. Le référentiel doit avoir un référentiel en amont avec une connexion externe au portail des plugins Gradle (par exemple `gradle-plugins-store`) afin que les plugins Gradle couramment requis soient disponibles pour la compilation. Pour plus d'informations, consultez la [documentation Gradle](#).

Publier des artefacts

Cette section décrit comment publier une bibliothèque Java créée avec Gradle dans un CodeArtifact référentiel.

Ajoutez d'abord le `maven-publish` plugin dans la `plugins` section du `build.gradle` fichier du projet.

```
plugins {  
    id 'java-library'  
    id 'maven-publish'  
}
```

Ajoutez ensuite une `publishing` section au `build.gradle` fichier de projet.

```
publishing {  
    publications {  
        mavenJava(MavenPublication) {  
            groupId = 'group-id'  
        }  
    }  
}
```

```
        artifactId = 'artifact-id'  
        version = 'version'  
        from components.java  
    }  
}  
repositories {  
    maven {  
        url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/  
maven/my_repo'  
        credentials {  
            username "aws"  
            password System.env.CODEARTIFACT_AUTH_TOKEN  
        }  
    }  
}  
}
```

Le maven-publish plugin génère un fichier POM basé sur le groupIdartifactId, et version spécifié dans la publishing section.

Une fois ces modifications build.gradle terminées, exécutez la commande suivante pour créer le projet et le télécharger dans le référentiel.

```
./gradlew publish
```

list-package-versionsÀ utiliser pour vérifier que le package a bien été publié.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333  
--repository my_repo --format maven\n--namespace com.company.framework --package my-package-name
```

Exemple de sortie :

```
{  
    "format": "maven",  
    "namespace": "com.company.framework",  
    "package": "example",  
    "versions": [  
        {  
            "version": "1.0",  
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",  
            "status": "Published"  
        }  
    ]  
}
```

```
    }
]
}
```

Pour plus d'informations, consultez les rubriques suivantes sur le site Web de Gradle :

- [Création de bibliothèques Java](#)
- [Publier un projet sous forme de module](#)

Exécuter une compilation Gradle dans IntelliJ IDEA

Vous pouvez exécuter une version Gradle dans IntelliJ IDEA qui extrait les dépendances de. CodeArtifact Pour vous authentifier CodeArtifact, vous devez fournir à Gradle un jeton d' CodeArtifact autorisation. Il existe trois méthodes pour fournir un jeton d'authentification.

- Méthode 1 : Stockage du jeton d'authentification dans `gradle.properties`. Utilisez cette méthode si vous êtes en mesure de remplacer ou d'ajouter du contenu au `gradle.properties` fichier.
- Méthode 2 : Stockage du jeton d'authentification dans un fichier séparé. Utilisez cette méthode si vous ne souhaitez pas modifier votre `gradle.properties` fichier.
- Méthode 3 : Génération d'un nouveau jeton d'authentification pour chaque exécution en l'exécutant en aws tant que script en ligne dans `build.gradle`. Utilisez cette méthode si vous souhaitez que le script Gradle récupère un nouveau jeton à chaque exécution. Le jeton ne sera pas stocké dans le système de fichiers.

Token stored in `gradle.properties`

Méthode 1 : Stockage du jeton d'authentification dans **`gradle.properties`**

 Note

L'exemple montre le `gradle.properties` fichier situé dans `GRADLE_USER_HOME`.

1. Mettez à jour votre `build.gradle` fichier avec l'extrait suivant :

```
repositories {
```

```
maven {  
    url  
    'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/  
    maven/my_repo'  
    credentials {  
        username "aws"  
        password "$codeartifactToken"  
    }  
}  
}
```

2. Pour récupérer des plugins CodeArtifact, ajoutez un pluginManagement bloc à votre settings.gradle fichier. Le pluginManagement bloc doit apparaître avant toute autre instruction dans settings.gradle.

```
pluginManagement {  
    repositories {  
        maven {  
            name 'my_repo'  
            url  
            'https://my_domain-111122223333.codeartifact.region.amazonaws.com/  
            maven/my_repo'  
            credentials {  
                username 'aws'  
                password "$codeartifactToken"  
            }  
        }  
    }  
}
```

3. Récupérez un jeton d' CodeArtifact authentification :

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --  
domain my_domain --domain-owner 111122223333 --query authorizationToken --output  
text --profile profile-name`
```

4. Écrivez le jeton d'authentification dans le gradle.properties fichier :

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > ~/.gradle/gradle.properties
```

Token stored in separate file

Méthode 2 : Stockage du jeton d'authentification dans un fichier séparé

1. Mettez à jour votre `build.gradle` fichier avec l'extrait suivant :

```
def props = new Properties()
file("file").withInputStream { props.load(it) }

repositories {

    maven {
        url
        'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
        credentials {
            username "aws"
            password props.getProperty("codeartifactToken")
        }
    }
}
```

2. Pour récupérer des plugins CodeArtifact, ajoutez un `pluginManagement` bloc à votre `settings.gradle` fichier. Le `pluginManagement` bloc doit apparaître avant toute autre instruction dans `settings.gradle`.

```
pluginManagement {
    def props = new Properties()
    file("file").withInputStream { props.load(it) }
    repositories {
        maven {
            name 'my_repo'
            url
            'https://my_domain-111122223333.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password props.getProperty("codeartifactToken")
            }
        }
    }
}
```

3. Récupérez un jeton d'CodeArtifact authentification :

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text --profile profile-name`
```

4. Écrivez le jeton d'authentification dans le fichier spécifié dans votre build.gradle fichier :

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > file
```

Token generated for each run in build.gradle

Méthode 3 : Génération d'un nouveau jeton d'authentification pour chaque exécution en exécutant un script **aws** en ligne dans **build.gradle**

1. Mettez à jour votre build.gradle fichier avec l'extrait suivant :

```
def codeartifactToken = "aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text --profile profile-name".execute().text
repositories {
    maven {
        url
        'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
        maven/my_repo/'
        credentials {
            username "aws"
            password codeartifactToken
        }
    }
}
```

2. Pour récupérer des plugins CodeArtifact, ajoutez un pluginManagement bloc à votre settings.gradle fichier. Le pluginManagement bloc doit apparaître avant toute autre instruction dans settings.gradle.

```
pluginManagement {
    def codeartifactToken = "aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text --profile profile-name".execute().text
    repositories {
```

```
maven {  
    name 'my_repo'  
    url  
    'https://my_domain-111122223333.codeartifact.region.amazonaws.com/  
    maven/my_repo'  
    credentials {  
        username 'aws'  
        password codeartifactToken  
    }  
}  
}
```

Utiliser CodeArtifact avec MVN

Vous utilisez la mvn commande pour exécuter les builds de Maven. Cette section explique comment configurer l'utilisation mvn d'un CodeArtifact référentiel.

Rubriques

- [Extraire les dépendances](#)
- [Publier des artefacts](#)
- [Publier des artefacts tiers](#)
- [Restreindre les téléchargements de dépendances Maven à un référentiel CodeArtifact](#)
- [Informations sur le projet Apache Maven](#)

Extraire les dépendances

Pour configurer mvn afin d'extraire les dépendances d'un CodeArtifact référentiel, vous devez modifier le fichier de configuration Maven et, éventuellement `settings.xml`, le POM de votre projet.

1. Si ce n'est pas le cas, créez et stockez un jeton d'CodeArtifact authentification dans une variable d'environnement, comme décrit dans la section [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#) pour configurer l'authentification auprès de votre CodeArtifact référentiel.
2. Dans `settings.xml` (généralement trouvé à `~/.m2/settings.xml`), ajoutez une `<servers>` section avec une référence à la variable d'CODEARTIFACT_AUTH_TOKEN environnement afin que Maven transmette le jeton dans les requêtes HTTP.

```
<settings>
...
  <servers>
    <server>
      <id>codeartifact</id>
      <username>aws</username>
      <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
    </server>
  </servers>
...
</settings>
```

3. Ajoutez le point de terminaison URL de votre CodeArtifact référentiel dans un `<repository>` élément. Vous pouvez le faire dans le `settings.xml` fichier POM de votre projet.

Vous pouvez récupérer le point de terminaison de votre dépôt à l'aide de la `get-repository-endpoint` AWS CLI commande.

Par exemple, avec un dépôt nommé `my_repo` à l'intérieur d'un domaine nommé `my_domain`, la commande est la suivante :

```
aws codeartifact get-repository-endpoint --domain my_domain --repository my_repo --format maven
```

La `get-repository-endpoint` commande renverra le point de terminaison du référentiel :

```
url 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/'
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

Ajoutez le point de terminaison du référentiel `settings.xml` comme suit.

```
<settings>
...

```

```
<profiles>
  <profile>
    <id>default</id>
    <repositories>
      <repository>
        <id>codeartifact</id>
        <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/</url>
      </repository>
    </repositories>
  </profile>
</profiles>
<activeProfiles>
  <activeProfile>default</activeProfile>
</activeProfiles>
...
</settings>
```

Vous pouvez également ajouter la <repositories> section à un fichier POM de projet CodeArtifact pour l'utiliser uniquement pour ce projet.

```
<project>
...
  <repositories>
    <repository>
      <id>codeartifact</id>
      <name>codeartifact</name>
      <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/</url>
    </repository>
  </repositories>
...
</project>
```

Important

Vous pouvez utiliser n'importe quelle valeur dans l'<id> élément, mais elle doit être identique dans les <repository> éléments <server> et. Cela permet d'inclure les informations d'identification spécifiées dans les demandes adressées à CodeArtifact.

Après avoir apporté ces modifications de configuration, vous pouvez créer le projet.

```
mvn compile
```

Maven enregistre l'URL complète de toutes les dépendances qu'il télécharge sur la console.

```
[INFO] -----< com.example.example:myapp >-----  
[INFO] Building myapp 1.0  
[INFO] -----[ jar ]-----  
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom  
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom (11 kB at 3.9 kB/s)  
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom  
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom  
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom (68 kB at 123  
kB/s)  
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar  
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/  
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar (54 kB at 134 kB/s)
```

Publier des artefacts

Pour publier un artefact Maven dans un CodeArtifact dépôt, vous devez également modifier `~/.m2/settings.xml` le POM du projet. mvn

1. Si ce n'est pas le cas, créez et stockez un jeton d'CodeArtifact authentification dans une variable d'environnement, comme décrit dans la section [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#) pour configurer l'authentification auprès de votre CodeArtifact référentiel.
2. Ajoutez une `<servers>` section à `settings.xml` avec une référence à la variable d'CODEARTIFACT_AUTH_TOKEN environnement afin que Maven transmette le jeton dans les requêtes HTTP.

```
<settings>
```

```
...
```

```
<servers>
  <server>
    <id>codeartifact</id>
    <username>aws</username>
    <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
  </server>
</servers>
...
</settings>
```

3. Ajoutez une <distributionManagement> section à celle de votre projet pom.xml.

```
<project>
  ...
  <distributionManagement>
    <repository>
      <id>codeartifact</id>
      <name>codeartifact</name>
      <url>https://my_domain-1112223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo</url>
    </repository>
  </distributionManagement>
  ...
</project>
```

Après avoir apporté ces modifications de configuration, vous pouvez créer le projet et le publier dans le référentiel spécifié.

```
mvn deploy
```

À utiliser pour vérifier que le package a bien été publié.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 1112223333
--repository my_repo --format maven \
--namespace com.company.framework --package my-package-name
```

Exemple de sortie :

```
{
  "defaultDisplayVersion": null,
```

```
"format": "maven",
"namespace": "com.company.framework",
"package": "my-package-name",
"versions": [
  {
    "version": "1.0",
    "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
    "status": "Published"
  }
]
```

Publier des artefacts tiers

Vous pouvez publier des artefacts Maven tiers dans un CodeArtifact référentiel avec `mvn deploy:deploy-file`. Cela peut être utile aux utilisateurs qui souhaitent publier des artefacts et qui ne disposent que de fichiers JAR et qui n'ont pas accès au code source du package ou aux fichiers POM.

La `mvn deploy:deploy-file` commande générera un fichier POM basé sur les informations transmises dans la ligne de commande.

Publier des artefacts Maven tiers

1. Si ce n'est pas le cas, créez et stockez un jeton d'CodeArtifact authentification dans une variable d'environnement, comme décrit dans la section [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#) pour configurer l'authentification auprès de votre CodeArtifact référentiel.
2. Créez un `~/.m2/settings.xml` fichier avec le contenu suivant :

```
<settings>
  <servers>
    <server>
      <id>codeartifact</id>
      <username>aws</username>
      <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
    </server>
  </servers>
</settings>
```

3. Exécutez la commande mvn deploy:deploy-file :

```
mvn deploy:deploy-file -DgroupId=commons-cli           \
-DartifactId=commons-cli      \
-Dversion=1.4           \
-Dfile=./commons-cli-1.4.jar  \
-Dpackaging=jar           \
-DrepositoryId=codeartifact \
-Durl=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/ \
maven/repo-name/
```

Note

L'exemple ci-dessus publie commons-cli 1.4. Modifiez les arguments groupId, artifactID, version et fichier pour publier un JAR différent.

Ces instructions sont basées sur des exemples du [Guide de déploiement d'un tiers JARs sur un dépôt distant](#), extrait de la documentation d'Apache Maven.

Restreindre les téléchargements de dépendances Maven à un référentiel CodeArtifact

Si un package ne peut pas être extrait d'un référentiel configuré, par défaut, la mvn commande le récupère depuis Maven Central. Ajoutez l'element `mirrors` dans `settings.xml` pour mvn toujours utiliser votre CodeArtifact dépôt.

```
<settings>
  ...
  <mirrors>
    <mirror>
      <id>central-mirror</id>
      <name>CodeArtifact Maven Central mirror</name>
      <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ \
maven/my_repo</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
  ...
```

```
</settings>
```

Si vous ajoutez un `mirrors` élément, vous devez également en avoir un `pluginRepository` dans votre `settings.xml` ou `pom.xml`. L'exemple suivant extrait les dépendances des applications et les plug-ins Maven depuis un CodeArtifact référentiel.

```
<settings>
...
<profiles>
  <profile>
    <pluginRepositories>
      <pluginRepository>
        <id>codeartifact</id>
        <name>CodeArtifact Plugins</name>
        <url>https://my_domain-11112223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_repo</url>
        <releases>
          <enabled>true</enabled>
        </releases>
        <snapshots>
          <enabled>true</enabled>
        </snapshots>
      </pluginRepository>
    </pluginRepositories>
  </profile>
</profiles>
...
</settings>
```

L'exemple suivant extrait les dépendances des applications depuis un CodeArtifact référentiel et extrait les plug-ins Maven depuis Maven Central.

```
<profiles>
  <profile>
    <id>default</id>
    ...
    <pluginRepositories>
      <pluginRepository>
        <id>central-plugins</id>
        <name>Central Plugins</name>
        <url>https://repo.maven.apache.org/maven2/</url>
        <releases>
```

```
        <enabled>true</enabled>
    </releases>
    <snapshots>
        <enabled>true</enabled>
    </snapshots>
</pluginRepository>
</pluginRepositories>
.....
</profile>
</profiles>
```

Informations sur le projet Apache Maven

Pour plus d'informations sur Maven, consultez les rubriques suivantes sur le site Web du projet Apache Maven :

- [Configuration de plusieurs référentiels](#)
- [Référence des paramètres](#)
- [Gestion de la distribution](#)
- [Profilés](#)

À utiliser CodeArtifact avec deps.edn

Vous utilisez deps.edn with clj pour gérer les dépendances des projets Clojure. Cette section explique comment configurer l'utilisation deps.edn d'un CodeArtifact référentiel.

Rubriques

- [Extraire les dépendances](#)
- [Publier des artefacts](#)

Extraire les dépendances

Pour configurer Clojure afin de récupérer les dépendances d'un CodeArtifact référentiel, vous devez modifier le fichier de configuration Maven, settings.xml

1. Dans settings.xml, ajoutez une <servers> section avec une référence à la variable d'CODEARTIFACT_AUTH_TOKEN environnement afin que Clojure transmette le jeton dans les requêtes HTTP.

Note

Clojure s'attend à ce que le fichier settings.xml se trouve à l'~/ .m2/ settings.xml adresse. Si ce n'est pas le cas, créez le fichier à cet emplacement.

```
<settings>
...
  <servers>
    <server>
      <id>codeartifact</id>
      <username>aws</username>
      <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
    </server>
  </servers>
...
</settings>
```

2. Si vous n'en avez pas déjà un, générez un XML POM pour votre projet à l'aide de `clj -Spom`.
3. Dans votre fichier `deps.edn` de configuration, ajoutez un dépôt correspondant à l'identifiant du serveur de Maven `settings.xml`.

```
:mvn/repos {
  "clojars" nil
  "central" nil
  "codeartifact" {:url "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/"}
}
```

Note

- `tools.depsgarantit que les clojars référentiels central et seront d'abord vérifiés pour les bibliothèques Maven. Ensuite, les autres référentiels répertoriés dans deps.edn seront vérifiés.`
- Pour empêcher le téléchargement directement depuis Clojars et Maven Central, `central clojars` vous devez le configurer sur. `nil`

Assurez-vous d'avoir le jeton CodeArtifact Auth dans une variable d'environnement (voir [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#)). Lors de la création du package après ces modifications, les dépendances deps.edn seront CodeArtifact extraites.

 Note

Pour utiliser un point de terminaison à double pile, utilisez le codeartifact.region.on.aws point de terminaison.

Publier des artefacts

1. Mettez à jour vos paramètres Maven et deps.edn incluez-les en CodeArtifact tant que serveur reconnu par Maven (voir). [Extraire les dépendances](#) Vous pouvez utiliser un outil tel que [deps-deploy](#) pour télécharger des artefacts vers CodeArtifact
2. Dans votre build.clj, ajoutez une deploy tâche pour télécharger les artefacts requis dans le codeartifact référentiel précédemment configuré.

```
(ns build
  (:require [deps-deploydeps-deploy :as dd]))  
  
(defn deploy []
  (dd/deploy {:installer :remote
              :artifact "PATH_TO_JAR_FILE.jar"
              :pom-file "pom.xml" ;; pom containing artifact coordinates
              :repository "codeartifact"}))
```

3. Publiez l'artefact en exécutant la commande suivante : clj -T:build deploy

Pour plus d'informations sur la modification des référentiels par défaut, consultez [la section Modification des référentiels par défaut dans les documents](#) Clojure Deps et la justification de référence de la CLI.

Publier avec curl

Cette section explique comment utiliser le client HTTP curl pour publier des artefacts Maven dans un CodeArtifact référentiel. La publication d'artefacts avec curl peut être utile si vous ne possédez pas ou ne souhaitez pas installer le client Maven dans vos environnements.

Publiez un artefact Maven avec curl

1. Récupérez un jeton CodeArtifact d'autorisation en suivant les étapes indiquées [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#) et revenez à ces étapes.
2. Utilisez la curl commande suivante pour publier le fichier JAR dans un CodeArtifact référentiel :

Dans chacune des curl commandes de cette procédure, remplacez les espaces réservés suivants :

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- *111122223333* Remplacez-le par l'ID du propriétaire de votre CodeArtifact domaine.
- Remplacez *us-west-2* par la région dans laquelle se trouve votre CodeArtifact domaine.
- Remplacez *my_repo* par le nom de votre CodeArtifact dépôt.

```
curl --request PUT https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/com/mycompany/app/my-app/1.0/my-app-1.0.jar \
--user "aws:$CODEARTIFACT_AUTH_TOKEN" --header "Content-Type: application/octet-stream" \
--data-binary @my-app-1.0.jar
```

Important

Vous devez préfixer la valeur du --data-binary paramètre par un @ caractère.

Lorsque vous mettez la valeur entre guillemets, @ celle-ci doit être incluse entre guillemets.

3. Utilisez la curl commande suivante pour publier le POM dans un CodeArtifact référentiel :

```
curl --request PUT https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/maven/my_repo/com/mycompany/app/my-app/1.0/my-app-1.0.pom \
--user "aws:$CODEARTIFACT_AUTH_TOKEN" --header "Content-Type: application/octet-stream" \
```

```
--data-binary @my-app-1.0.pom
```

4. À ce stade, l'artefact Maven sera dans votre CodeArtifact dépôt avec un statut de Unfinished. Pour pouvoir consommer le package, celui-ci doit être en bon Published état. Vous pouvez déplacer le package de Unfinished vers Published en téléchargeant un maven-metadata.xml fichier dans votre package ou en appelant l'[UpdatePackageVersionsStatus API](#) pour modifier le statut.

- a. Option 1 : utilisez la curl commande suivante pour ajouter un maven-metadata.xml fichier à votre package :

```
curl --request PUT  
https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/  
maven/my_repo/com/mycompany/app/my-app/maven-metadata.xml \  
--user "aws:$CODEARTIFACT_AUTH_TOKEN" --header "Content-Type: application/  
octet-stream" \  
--data-binary @maven-metadata.xml
```

Voici un exemple du contenu d'un maven-metadata.xml fichier :

```
<metadata modelVersion="1.1.0">  
  <groupId>com.mycompany.app</groupId>  
  <artifactId>my-app</artifactId>  
  <versioning>  
    <latest>1.0</latest>  
    <release>1.0</release>  
    <versions>  
      <version>1.0</version>  
    </versions>  
    <lastUpdated>20200731090423</lastUpdated>  
  </versioning>  
</metadata>
```

- b. Option 2 : mettez à jour le statut du package Published avec l'[UpdatePackageVersionsStatusAPI](#).

```
aws codeartifact update-package-versions-status \  
  --domain my_domain \  
  --domain-owner 111122223333 \  
  --repository my_repo \  
  --format maven \  
  --status published
```

```
--namespace com.mycompany.app \
--package my-app \
--versions 1.0 \
--target-status Published
```

Si vous ne disposez que du fichier JAR d'un artefact, vous pouvez publier une version de package consommable dans un CodeArtifact référentiel à l'aide de `mvn`. Cela peut être utile si vous n'avez pas accès au code source ou au POM de l'artefact. Consultez [Publier des artefacts tiers](#) pour plus de détails.

Utiliser les checksums Maven

Lorsqu'un artefact Maven est publié dans un AWS CodeArtifact référentiel, la somme de contrôle associée à chaque ressource ou fichier du package est utilisée pour valider le téléchargement. Les fichiers jar, pom et war sont des exemples d'actifs. Pour chaque ressource, l'artefact Maven contient plusieurs fichiers de somme de contrôle qui utilisent le nom de la ressource avec une extension supplémentaire, telle que ou. md5 sha1 Par exemple, les fichiers de somme de contrôle d'un fichier nommé `my-maven-package.jar` peuvent être `my-maven-package.jar.md5` et `my-maven-package.jar.sha1`.

 Note

Maven utilise le terme `artifact`. Dans ce guide, un package Maven est identique à un artefact Maven. Pour plus d'informations, consultez le [AWS CodeArtifact package](#).

Stockage de Checksum

CodeArtifact ne stocke pas les sommes de contrôle Maven en tant qu'actifs. Cela signifie que les checksums n'apparaissent pas sous forme d'actifs individuels dans le résultat de [l'API ListPackageVersionAssets](#). Au lieu de cela, les sommes de contrôle calculées par CodeArtifact sont disponibles pour chaque actif dans tous les types de somme de contrôle pris en charge. Par exemple, une partie de la réponse à l'appel de `ListPackageVersionAssets` la version du package Maven `commons-lang:commons-lang 2.1` est la suivante :

```
{
  "name": "commons-lang-2.1.jar",
  "size": 207723,
```

```
"hashes": {
    "MD5": "51591549f1662a64543f08a1d4a0cf87",
    "SHA-1": "4763ecc9d78781c915c07eb03e90572c7ff04205",
    "SHA-256": "2ded7343dc8e57decd5e6302337139be020fdd885a2935925e8d575975e480b9",
    "SHA-512": "a312a5e33b17835f2e82e74ab52ab81f0dec01a7e72a2ba58bb76b6a197ffcd2bb410e341ef7b3720f3b595ce49fd
},
{
    "name": "commons-lang-2.1.pom",
    "size": 9928,
    "hashes": {
        "MD5": "8e41bacdd69de9373c20326d231c8a5d",
        "SHA-1": "a34d992202615804c534953aba402de55d8ee47c",
        "SHA-256": "f1a709cd489f23498a0b6b3dfbfc0d21d4f15904791446dec7f8a58a7da5bd6a",
        "SHA-512": "1631ce8fe4101b6cde857f5b1db9b29b937f98ba445a60e76cc2b8f2a732ff24d19b91821a052c1b56b73325104e9
    }
},
{
    "name": "maven-metadata.xml",
    "size": 121,
    "hashes": {
        "MD5": "11bb3d48d984f2f49cea1e150b6fa371",
        "SHA-1": "7ef872be17357751ce65cb907834b6c5769998db",
        "SHA-256": "d04d140362ea8989a824a518439246e7194e719557e8d701831b7f5a8228411c",
        "SHA-512": "001813a0333ce4b2a47cf44900470bc2265ae65123a8c6b5ac5f2859184608596baa4d8ee0696d0a497755dade0f6
    }
}
```

Même si les sommes de contrôle ne sont pas stockées en tant qu'actifs, les clients Maven peuvent toujours publier et télécharger les sommes de contrôle aux emplacements prévus. Par exemple, commons-lang:commons-lang 2.1 s'il se trouve dans un référentiel appelé maven-repo, le chemin URL de la somme de contrôle SHA-256 du fichier JAR serait :

```
/maven/maven-repo/commons-lang/commons-lang/2.1/commons-lang-2.1.jar.sha256
```

Si vous téléchargez des packages Maven existants (par exemple, des packages précédemment stockés dans Amazon S3) à CodeArtifact l'aide d'un client HTTP générique tel que curl, il n'est pas nécessaire de télécharger les checksums. CodeArtifact les générera automatiquement. Si vous souhaitez vérifier que les actifs ont été chargés correctement, vous pouvez utiliser l'opération

ListPackageVersionAssets API pour comparer les sommes de contrôle de la réponse aux valeurs de contrôle d'origine de chaque ressource.

Incompatibilité des checksum lors de la publication

Outre les actifs et les checksums, les artefacts Maven contiennent également un maven-metadata.xml fichier. La séquence de publication normale d'un package Maven consiste à télécharger d'abord tous les actifs et les sommes de contrôle, puis à. maven-metadata.xml Par exemple, la séquence de publication de la version du package Maven commons-lang 2.1 décrite précédemment, en supposant que le client ait été configuré pour publier des fichiers de somme de contrôle SHA-256, serait la suivante :

```
PUT commons-lang-2.1.jar
PUT commons-lang-2.1.jar.sha256
PUT commons-lang-2.1.pom
PUT commons-lang-2.1.pom.sha256
PUT maven-metadata.xml
PUT maven-metadata.xml.sha256
```

Lors du téléchargement du fichier de somme de contrôle pour un actif, tel qu'un fichier JAR, la demande de téléchargement de somme de contrôle échouera avec une réponse de 400 (mauvaise demande) en cas de non-correspondance entre la valeur de la somme de contrôle téléchargée et la valeur de la somme de contrôle calculée par. CodeArtifact Si l'actif correspondant n'existe pas, la demande échouera avec une réponse 404 (introuvable). Pour éviter cette erreur, vous devez d'abord télécharger la ressource, puis télécharger la somme de contrôle.

Lorsqu'il maven-metadata.xml est téléchargé, le statut de la version du package Maven passe CodeArtifact normalement de Unfinished à Published. Si une incompatibilité de somme de contrôle est détectée pour un actif, CodeArtifact renvoie un 400 (mauvaise demande) en réponse à la demande de maven-metadata.xml publication. Cette erreur peut empêcher le client de télécharger les fichiers correspondant à cette version du package. Si cela se produit et que le maven-metadata.xml fichier n'est pas chargé, aucune ressource de la version du package déjà téléchargée ne peut être téléchargée. Cela est dû au fait que le statut de la version du package n'est pas défini sur Published et reste le mêmeUnfinished.

CodeArtifact permet d'ajouter d'autres actifs à une version de package Maven même après maven-metadata.xml le téléchargement et le statut de la version du package défini sur. Published Dans ce statut, une demande de téléchargement d'un fichier de somme de contrôle non concordant échouera également avec une réponse 400 (mauvaise demande). Toutefois, comme le statut de

version du package a déjà été défini sur `Published`, vous pouvez télécharger n'importe quelle ressource du package, y compris celles pour lesquelles le téléchargement du fichier de somme de contrôle a échoué. Lors du téléchargement d'une somme de contrôle pour une ressource dont le téléchargement du fichier de somme de contrôle a échoué, la valeur de la somme de contrôle que le client reçoit est la valeur de la somme de contrôle calculée sur la CodeArtifact base des données de l'actif téléchargées.

CodeArtifact les comparaisons de checksum font la distinction majuscules/minuscules, et les checksums calculés par CodeArtifact sont formatés en minuscules. Par conséquent, si la somme de contrôle `909FA780F76DA393E992A3D2D495F468` est téléchargée, elle échouera en raison d'une incompatibilité de somme de contrôle, car elle CodeArtifact n'est pas traitée comme égale à `909fa780f76da393e992a3d2d495f468`.

Réparation après des incohérences entre les checksum

Si le téléchargement d'un checksum échoue en raison d'une incompatibilité de checksum, essayez l'une des méthodes suivantes pour le récupérer :

- Exécutez à nouveau la commande qui publie l'artefact Maven. Cela peut fonctionner si un problème réseau a endommagé le fichier checksum. Si cela résout le problème réseau, la somme de contrôle correspond et le téléchargement est réussi.
- Supprimez la version du package, puis republiez-la. Pour plus d'informations, consultez [DeletePackageVersions](#) le manuel de référence des CodeArtifact API AWS.

Utiliser les instantanés Maven

Un instantané Maven est une version spéciale d'un package Maven qui fait référence au dernier code de branche de production. Il s'agit d'une version de développement qui précède la version finale. Vous pouvez identifier une version instantanée d'un package Maven à l'aide du suffixe `SNAPSHOT` ajouté à la version du package. Par exemple, l'instantané de la version `1.1` est `1.1-SNAPSHOT`. Pour plus d'informations, voir [Qu'est-ce qu'une version SNAPSHOT ?](#) sur le site Web du projet Apache Maven.

AWS CodeArtifact prend en charge la publication et la consommation d'instantanés Maven. Les instantanés uniques qui utilisent un numéro de version basé sur le temps sont les seuls instantanés pris en charge. CodeArtifact ne prend pas en charge les instantanés non uniques générés par les clients Maven 2. Vous pouvez publier un instantané Maven compatible dans n'importe quel CodeArtifact référentiel.

Rubriques

- [Publication d'instantanés dans CodeArtifact](#)
- [Consommation de versions instantanées](#)
- [Supprimer des versions de snapshots](#)
- [Publication d'instantanés avec curl](#)
- [Instantanés et connexions externes](#)
- [Instantanés et référentiels en amont](#)

Publication d'instantanés dans CodeArtifact

AWS CodeArtifact prend en charge les modèles de demande que les clients, par exemple Maven, utilisent lors de la publication de snapshots. De ce fait, vous pouvez suivre la documentation de votre outil de génération ou de votre gestionnaire de packages sans avoir une compréhension détaillée de la manière dont les instantanés Maven sont publiés. Si vous effectuez une opération plus complexe, cette section décrit en détail comment CodeArtifact gérer les instantanés.

Lorsqu'un instantané Maven est publié dans un CodeArtifact référentiel, sa version précédente est préservée dans une nouvelle version appelée build. Chaque fois qu'un instantané Maven est publié, une nouvelle version de build est créée. Toutes les versions précédentes d'un instantané sont conservées dans ses versions de compilation. Lorsqu'un instantané Maven est publié, le statut de la version de son package est défini sur Published et le statut de la version contenant la version précédente est défini sur Unlisted. Ce comportement s'applique uniquement aux versions de package Maven dont le suffixe est -SNAPSHOT la version du package.

Par exemple, les versions instantanées d'un package Maven appelé com.mycompany.myapp:pkg-1 sont téléchargées dans un CodeArtifact référentiel appelé my-maven-repo. La version instantanée est 1.0-SNAPSHOT. Jusqu'à présent, aucune version de com.mycompany.myapp:pkg-1 n'a été publiée. Tout d'abord, les actifs de la version initiale sont publiés sur les chemins suivants :

```
PUT maven/my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/  
pkg-1-1.0-20210728.194552-1.jar  
PUT maven/my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/  
pkg-1-1.0-20210728.194552-1.pom
```

Notez que l'horodatage 20210728.194552-1 est généré par le client qui publie les versions instantanées.

Une fois les fichiers .pom et .jar chargés, la seule version com.mycompany.myapp:pkg-1 qui existe dans le référentiel est 1.0-20210728.194552-1. Cela se produit même si la version spécifiée dans le chemin précédent est 1.0-SNAPSHOT. L'état de la version du package à ce stade est Unfinished.

```
aws codeartifact list-package-versions --domain my-domain --repository \
my-maven-repo --package pkg-1 --namespace com.mycompany.myapp --format maven
{
  "versions": [
    {
      "version": "1.0-20210728.194552-1",
      "revision": "GipMW+599JmwTcTLaXo9YvDsVQ2bcrrk/02rWJhoKUU=",
      "status": "Unfinished"
    }
  ],
  "defaultDisplayVersion": null,
  "format": "maven",
  "package": "pkg-1",
  "namespace": "com.mycompany.myapp"
}
```

Ensuite, le client télécharge le maven-metadata.xml fichier correspondant à la version du package :

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

Lorsque le fichier maven-metadata.xml est correctement chargé, CodeArtifact crée la version 1.0-SNAPSHOT du package et définit la 1.0-20210728.194552-1 version sur Unlisted.

```
aws codeartifact list-package-versions --domain my-domain --repository \
my-maven-repo --package pkg-1 --namespace com.mycompany.myapp --format maven
{
  "versions": [
    {
      "version": "1.0-20210728.194552-1",
      "revision": "GipMW+599JmwTcTLaXo9YvDsVQ2bcrrk/02rWJhoKUU=",
      "status": "Unlisted"
    },
    {
      "version": "1.0-SNAPSHOT",
      "revision": "tWu8n3IX5HR82vzVZQAxlwcvvA4U/+S80edWNakil24=",
      "status": "Unlisted"
    }
  ]
}
```

```
        "status": "Published"
    }
],
"defaultDisplayVersion": "1.0-SNAPSHOT",
"format": "maven",
"package": "pkg-1",
"namespace": "com.mycompany.myapp"
}
```

À ce stade, la version instantanée 1.0-SNAPSHOT peut être utilisée dans une version. Bien qu'il existe deux versions de com.mycompany.myapp:pkg-1 dans le référentiel my-maven-repo, elles contiennent toutes les deux les mêmes actifs.

```
aws codeartifact list-package-version-assets --domain my-domain --repository \
my-maven-repo --format maven --namespace com.mycompany.myapp \
--package pkg-1 --package-version 1.0-SNAPSHOT --query 'assets[*].name'
[
    "pkg-1-1.0-20210728.194552-1.jar",
    "pkg-1-1.0-20210728.194552-1.pom"
]
```

L'exécution de la même list-package-version-assets commande que celle indiquée précédemment avec le --package-version paramètre modifié 1.0-20210728.194552-1 permet d'obtenir une sortie identique.

Au fur et à mesure que des versions supplémentaires de 1.0-SNAPSHOT sont ajoutées au référentiel, une nouvelle version de Unlisted package est créée pour chaque nouvelle version. Les actifs de la version 1.0-SNAPSHOT sont mis à jour à chaque fois afin que la version fasse toujours référence à la dernière version pour cette version. La mise à jour du 1.0-SNAPSHOT avec les dernières ressources est lancée en téléchargeant le maven-metadata.xml fichier pour la nouvelle version.

Consommation de versions instantanées

Si vous demandez un instantané, la version avec le statut Published est renvoyée. Il s'agit toujours de la version la plus récente de l'instantané Maven. Vous pouvez également demander une version particulière d'un instantané en utilisant le numéro de version du build (par exemple, 1.0-20210728.194552-1) au lieu de la version instantanée (par exemple, 1.0-SNAPSHOT) dans le chemin URL. Pour voir les versions de compilation d'un instantané Maven,

utilisez l'[ListPackageVersions](#) API du guide des CodeArtifact API et définissez le paramètre d'Unlistedétat sur.

Supprimer des versions de snapshots

Pour supprimer toutes les versions de build d'un instantané Maven, utilisez l'[DeletePackageVersions](#) API en spécifiant les versions que vous souhaitez supprimer.

Publication d'instantanés avec curl

Si vous avez des versions de snapshots stockées dans Amazon Simple Storage Service (Amazon S3) ou dans un autre produit de référentiel d'artefacts, vous souhaiterez peut-être les republier sur AWS CodeArtifact. En raison de la prise CodeArtifact en charge des instantanés Maven (voir [Publication d'instantanés dans CodeArtifact](#)), la publication d'instantanés avec un client HTTP générique tel que la publication de versions de Maven curl est plus complexe que la publication de versions de Maven, comme décrit dans [Publier avec curl](#). Notez que cette section n'est pas pertinente si vous créez et déployez des versions de snapshots avec un client Maven tel que mvn ou gradle. Vous devez suivre la documentation relative à ce client.

La publication d'une version instantanée implique la publication d'une ou de plusieurs versions d'une version instantanée. Dans CodeArtifact, s'il existe n versions d'une version instantanée, il y aura n + 1 CodeArtifact versions : n versions de version avec un statut deUnlisted, et une version instantanée (la dernière version publiée) avec un statut dePublished. La version instantanée (c'est-à-dire la version dont la chaîne de version contient « -SNAPSHOT ») contient un ensemble d'actifs identique à celui de la dernière version publiée. La méthode la plus simple pour créer cette structure curl est la suivante :

1. Publiez tous les actifs de toutes les versions à l'aide de curl.
2. Publiez le maven-metadata.xml fichier de la dernière version (c'est-à-dire la version avec l'horodatage le plus récent) avec curl. Cela créera une version avec « -SNAPSHOT » dans la chaîne de version et avec le bon ensemble d'actifs.
3. Utilisez l'[UpdatePackageVersionsStatus](#) API pour définir le statut de toutes les versions de version non récentes surUnlisted.

Utilisez les curl commandes suivantes pour publier les actifs de capture d'écran (tels que les fichiers .jar et .pom) pour la version instantanée 1.0-SNAPSHOT d'un package : com.mycompany.app:pkg-1

```
curl --user "aws:$CODEARTIFACT_AUTH_TOKEN" -H "Content-Type: application/octet-stream"
 \
 -X PUT https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_maven_repo/com/mycompany/app/pkg-1/1.0-SNAPSHOT/
pkg-1-1.0-20210729.171330-2.jar \
--data-binary @pkg-1-1.0-20210728.194552-1.jar
```

```
curl --user "aws:$CODEARTIFACT_AUTH_TOKEN" -H "Content-Type: application/octet-stream"
 \
 -X PUT https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_maven_repo/com/mycompany/app/pkg-1/1.0-SNAPSHOT/
pkg-1-1.0-20210729.171330-2.pom \
--data-binary @pkg-1-1.0-20210728.194552-1.pom
```

Lorsque vous utilisez ces exemples :

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- *111122223333* Remplacez-le par l' Compte AWS ID du propriétaire de votre CodeArtifact domaine.
- *us-west-2* Remplacez-le par celui Région AWS dans lequel se trouve votre CodeArtifact domaine.
- *my_maven_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

Important

Vous devez préfixer la valeur du --data-binary paramètre par le @ caractère. Lorsque vous mettez la valeur entre guillemets, @ celle-ci doit être incluse entre guillemets.

Vous pouvez avoir plus de deux actifs à télécharger pour chaque build. Par exemple, il peut y avoir des fichiers Javadoc et JAR source en plus du JAR principal et. pom. xml Il n'est pas nécessaire de publier des fichiers de somme de contrôle pour les actifs de la version du package, car des sommes de contrôle CodeArtifact sont automatiquement générées pour chaque ressource téléchargée. Pour vérifier que les actifs ont été chargés correctement, récupérez les sommes de contrôle générées à l'aide de la list-package-version-assets commande et comparez-les aux sommes de contrôle d'origine. Pour plus d'informations sur la façon dont CodeArtifact les sommes de contrôle Maven sont gérées, consultez. [Utiliser les checksums Maven](#)

Utilisez la commande curl suivante pour publier le `maven-metadata.xml` fichier correspondant à la dernière version de build :

```
curl --user "aws:$CODEARTIFACT_AUTH_TOKEN" -H "Content-Type: application/octet-stream"  
\\  
-X PUT https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/  
maven/my_maven_repo/com/mycompany/app/pkg-1/1.0-SNAPSHOT/maven-metadata.xml \\  
--data-binary @maven-metadata.xml
```

Le `maven-metadata.xml` fichier doit faire référence à au moins un des actifs de la dernière version de build de l'`<snapshotVersions>` élément. En outre, la `<timestamp>` valeur doit être présente et correspondre à l'horodatage figurant dans les noms des fichiers d'actifs. Par exemple, pour le 20210729.171330-2 build publié précédemment, le contenu de `maven-metadata.xml` serait :

```
<?xml version="1.0" encoding="UTF-8"?>  
<metadata>  
  <groupId>com.mycompany.app</groupId>  
  <artifactId>pkg-1</artifactId>  
  <version>1.0-SNAPSHOT</version>  
  <versioning>  
    <snapshot>  
      <timestamp>20210729.171330</timestamp>  
      <buildNumber>2</buildNumber>  
    </snapshot>  
    <lastUpdated>20210729171330</lastUpdated>  
    <snapshotVersions>  
      <snapshotVersion>  
        <extension>jar</extension>  
        <value>1.0-20210729.171330-2</value>  
        <updated>20210729171330</updated>  
      </snapshotVersion>  
      <snapshotVersion>  
        <extension>pom</extension>  
        <value>1.0-20210729.171330-2</value>  
        <updated>20210729171330</updated>  
      </snapshotVersion>  
    </snapshotVersions>  
  </versioning>  
</metadata>
```

Après `maven-metadata.xml` la publication, la dernière étape consiste à définir toutes les autres versions de build (c'est-à-dire toutes les versions de build à l'exception de la dernière version) pour

qu'elles aient le statut de version du package `deUnlisted`. Par exemple, si la `1.0-SNAPSHOT` version comporte deux versions, la première étant la version `20210728.194552-1`, la commande permettant de définir cette version `Unlisted` est la suivante :

```
aws codeartifact update-package-versions-status --domain my-domain --domain-owner
111122223333 \
--repository my-maven-repo --format maven --namespace com.mycompany.app --package
pkg-1 \
--versions 1.0-20210728.194552-1 --target-status Unlisted
```

Instantanés et connexions externes

Les instantanés Maven ne peuvent pas être récupérés depuis un dépôt public Maven via une connexion externe. AWS CodeArtifact ne prend en charge que l'importation des versions de Maven.

Instantanés et référentiels en amont

En général, les instantanés Maven fonctionnent de la même manière que les versions de Maven lorsqu'ils sont utilisés avec des référentiels en amont, mais il existe une limite si vous prévoyez de publier des instantanés de la même version de package dans deux référentiels ayant une relation en amont. Supposons, par exemple, qu'il existe deux référentiels dans un AWS CodeArtifact domaine R et U que se U trouve un dépôt en amont de R. Si vous publiez une nouvelle version dans R, lorsqu'un client Maven demande la dernière version de cette version instantanée, CodeArtifact renvoie la dernière version de U. Cela peut être inattendu puisque la dernière version est maintenant disponible R, non U. Il existe deux moyens d'éviter cela :

1. Ne publiez pas de versions d'une version instantanée telle que `1.0-SNAPSHOT` dans R, si `1.0-SNAPSHOT` elle existe dans U.
2. Utilisez les contrôles d'origine CodeArtifact du package pour désactiver les flux ascendants sur ce package dans R. Ce dernier vous permettra de publier des versions de `1.0-SNAPSHOT` in R, mais cela R empêchera également d'obtenir d'autres versions de ce package U qui ne sont pas déjà conservées.

Demande de packages Maven depuis des connexions en amont et externes

Importation de noms d'actifs standard

Lors de l'importation d'une version de package Maven depuis un référentiel public, tel que Maven Central, AWS CodeArtifact tente d'importer toutes les ressources de cette version de package.

Comme décrit dans [Demande d'une version de package avec des référentiels en amont](#), l'importation a lieu lorsque :

- Un client demande un actif Maven à partir d'un CodeArtifact référentiel.
- La version du package n'est pas déjà présente dans le référentiel ou dans ses flux ascendants.
- Il existe une connexion externe accessible à un dépôt Maven public.

Même si le client n'a demandé qu'une seule ressource, il CodeArtifact tente d'importer toutes les ressources qu'il trouve pour cette version de package. La CodeArtifact manière de découvrir quels actifs sont disponibles pour une version de package Maven dépend du référentiel public concerné. Certains référentiels Maven publics permettent de demander une liste d'actifs, mais d'autres ne le font pas. Pour les référentiels qui ne permettent pas de répertorier les actifs, CodeArtifact génère un ensemble de noms d'actifs susceptibles d'exister. Par exemple, lorsqu'un actif de la version du package Maven `junit 4.13.2` est demandé, CodeArtifact il tente d'importer les actifs suivants :

- `junit-4.13.2.pom`
- `junit-4.13.2.jar`
- `junit-4.13.2-javadoc.jar`
- `junit-4.13.2-sources.jar`

Importation de noms d'actifs non standard

Lorsqu'un client Maven demande un actif qui ne correspond pas à l'un des modèles décrits ci-dessus, CodeArtifact vérifie si cet actif est présent dans le référentiel public. Si la ressource est présente, elle sera importée et ajoutée à l'enregistrement de version du package existant, le cas échéant. Par exemple, la version du package Maven `com.android.tools.build:aapt2 7.3.1-8691043` contient les actifs suivants :

- `aapt2-7.3.1-8691043.pom`
- `aapt2-7.3.1-8691043-windows.jar`
- `aapt2-7.3.1-8691043-osx.jar`

- aapt2-7.3.1-8691043-linux.jar

Lorsqu'un client demande le fichier POM, s'il n' CodeArtifact est pas en mesure de répertorier les actifs de la version du package, le POM sera le seul actif importé. Cela est dû au fait qu'aucun des autres actifs ne correspond aux modèles de noms d'actifs standard. Toutefois, lorsque le client demande l'une des ressources JAR, cette ressource est importée et ajoutée à la version du package existante stockée dans CodeArtifact. Les versions du package du référentiel le plus en aval (le référentiel contre lequel le client a fait la demande) et du référentiel associé à la connexion externe seront mises à jour pour contenir le nouvel actif, comme décrit dans [Conservation des packages depuis les référentiels en amont](#)

Normalement, une fois qu'une version de package est conservée dans un CodeArtifact référentiel, elle n'est pas affectée par les modifications apportées aux référentiels en amont. Pour de plus amples informations, veuillez consulter [Conservation des packages depuis les référentiels en amont](#). Cependant, le comportement des actifs Maven avec des noms non standard décrit précédemment constitue une exception à cette règle. Bien que la version du package en aval ne change pas sans qu'un actif supplémentaire ne soit demandé par un client, dans ce cas, la version du package conservée est modifiée après avoir été initialement conservée et n'est donc pas immuable. Ce comportement est nécessaire car les actifs Maven portant des noms non standard ne seraient autrement pas accessibles via CodeArtifact. Ce comportement est également activé s'ils sont ajoutés à une version de package Maven sur un référentiel public une fois que la version du package a été conservée dans un CodeArtifact référentiel.

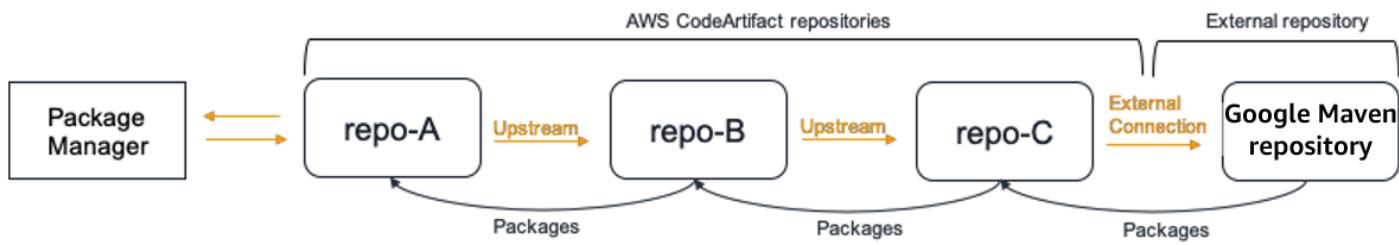
Vérifier l'origine des actifs

Lorsque vous ajoutez un nouvel actif à une version de package Maven précédemment conservée, vous CodeArtifact confirmez que l'origine de la version de package conservée est la même que celle du nouvel actif. Cela empêche de créer une version de package « mixte » dans laquelle différents actifs proviennent de différents référentiels publics. Sans cette vérification, un mélange d'actifs peut se produire si une version de package Maven est publiée dans plusieurs référentiels publics et que ces référentiels font partie du graphe en amont d'un CodeArtifact référentiel.

Importation de nouveaux actifs et de l'état de la version du package dans les référentiels en amont

L'[état des versions](#) de package des versions de package dans les référentiels en amont peut CodeArtifact empêcher de conserver ces versions dans les référentiels en aval.

Par exemple, supposons qu'un domaine possède trois référentiels : `repo-A`, `repo-B`, et `repo-C`, où `repo-B` trouve en amont `repo-A` et `repo-C` en amont de `repo-B`.



La version 7.3.1 du package Maven `com.android.tools.build:aapt2` est présente dans `repo-B` et a un statut `dePublished`. Il n'est pas présent dans `repo-A`. Si un client demande un actif de cette version de package dans `repo-A`, la réponse sera 200 (OK) et la version du package Maven 7.3.1 sera conservée dans `repo-A`. Toutefois, si le statut de la version 7.3.1 du package `repo-B` est `Archived` ou `Disposed`, la réponse sera 404 (Introuvable) car les actifs des versions du package correspondant à ces deux statuts ne sont pas téléchargeables.

Notez que le [fait de définir le contrôle d'origine du package](#) sur `upstream=BLOCK` pour `com.android.tools.build:aapt2` in `repo-A` et `repo-B`, et `repo-C` empêchera la récupération de nouveaux actifs pour toutes les versions de ce package dans `repo-A`, quel que soit le statut de la version du package.

Résolution des problèmes liés à Maven

Les informations suivantes peuvent vous aider à résoudre les problèmes courants liés à Maven et CodeArtifact

Désactivez les mises en parallèle pour corriger l'erreur 429 : Too Many Requests

À partir de la version 3.9.0, Maven télécharge les artefacts du package en parallèle (jusqu'à 5 fichiers à la fois). Cela peut entraîner une réponse occasionnelle avec un code de réponse d'erreur 429 (trop de demandes). Si vous rencontrez cette erreur, vous pouvez désactiver les mises en parallèle pour la corriger.

Pour désactiver les sorties parallèles, définissez la `aether.connector.basic.parallelPut` propriété sur `false` dans votre profil dans votre `settings.xml` fichier, comme illustré dans l'exemple suivant :

```
<settings>
```

```
<profiles>
  <profile>
    <id>default</id>
    <properties>
      <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
    </properties>
  </profile>
</profiles>
<settings>
```

Pour plus d'informations, consultez les [options de configuration d'Artifact Resolver](#) dans la documentation Maven.

Utilisation CodeArtifact avec npm

Ces rubriques décrivent comment utiliser npm, le gestionnaire de packages Node.js, avec CodeArtifact.

 Note

CodeArtifact supports node v4.9.1 et plus tard npm v5.0.0 et plus tard.

Rubriques

- [Configurer et utiliser npm avec CodeArtifact](#)
- [Configurer et utiliser Yarn avec CodeArtifact](#)
- [support des commandes npm](#)
- [gestion des balises npm](#)
- [Support pour les gestionnaires de paquets compatibles npm](#)

Configurer et utiliser npm avec CodeArtifact

Après avoir créé un référentiel dans CodeArtifact, vous pouvez utiliser le client npm pour installer et publier des packages. La méthode recommandée pour configurer npm avec le point de terminaison et le jeton d'autorisation de votre référentiel consiste à utiliser la `aws codeartifact login` commande. Vous pouvez également configurer npm manuellement.

Table des matières

- [Configuration de npm avec la commande login](#)
- [Configuration de npm sans utiliser la commande de connexion](#)
- [Exécution de commandes npm](#)
- [Vérification de l'authentification et de l'autorisation npm](#)
- [Revenir au registre npm par défaut](#)
- [Résolution des problèmes d'installation lents avec npm 8.x ou supérieur](#)

Configuration de npm avec la commande login

Utilisez la `aws codeartifact login` commande pour récupérer les informations d'identification à utiliser avec npm.

Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

Important

Si vous utilisez npm 10.x ou version ultérieure, vous devez utiliser la AWS CLI version 2.9.5 ou plus récente pour exécuter correctement la commande `aws codeartifact login`

```
aws codeartifact login --tool npm --domain my_domain --domain-owner 111122223333 --  
repository my_repo
```

Cette commande apporte les modifications suivantes à votre fichier `~/.npmrc` :

- Ajoute un jeton d'autorisation après l'avoir récupéré à l' CodeArtifact aide de vos AWS informations d'identification.
- Définit le registre npm sur le référentiel spécifié par l'`--repository` option.
- Pour npm 6 et versions antérieures : ajoute `"always-auth=true"` afin que le jeton d'autorisation soit envoyé pour chaque commande npm.

La période d'autorisation par défaut après l'appel `login` est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la login commande](#).

Configuration de npm sans utiliser la commande de connexion

Vous pouvez configurer npm avec votre CodeArtifact référentiel sans la `aws codeartifact login` commande en mettant à jour manuellement la configuration npm.

Pour configurer npm sans utiliser la commande de connexion

1. Dans une ligne de commande, récupérez un jeton CodeArtifact d'autorisation et stockez-le dans une variable d'environnement. npm utilisera ce jeton pour s'authentifier auprès de votre dépôt. CodeArtifact

Note

La commande suivante est destinée aux machines macOS ou Linux. Pour plus d'informations sur la configuration des variables d'environnement sur un ordinateur Windows, consultez [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#).

```
CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

2. Obtenez le point de terminaison de votre CodeArtifact dépôt en exécutant la commande suivante. Le point de terminaison de votre référentiel est utilisé pour pointer npm vers votre référentiel afin d'installer ou de publier des packages.

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format npm
```

L'URL suivante est un exemple de point de terminaison du référentiel.

```
https://my\_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my\_repo/
```

⚠️ Important

L'URL du registre doit se terminer par une barre oblique (/). Dans le cas contraire, vous ne pourrez pas vous connecter au référentiel.

3. Utilisez la `npm config set` commande pour définir le registre sur votre CodeArtifact référentiel. Remplacez l'URL par l'URL du point de terminaison du référentiel de l'étape précédente.

```
npm config set
  registry=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
  npm/my_repo/
```

ℹ️ Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

4. Utilisez la `npm config set` commande pour ajouter votre jeton d'autorisation à votre configuration npm.

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
  npm/my_repo/:_authToken=$CODEARTIFACT_AUTH_TOKEN
```

Pour npm 6 ou inférieur : pour que npm transmette toujours le jeton d'authentification CodeArtifact, même pour les GET demandes, définissez la variable de `always-auth` configuration avec `npm config set`

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
  npm/my_repo/:always-auth=true
```

Exemple de fichier de configuration npm () `.npmrc`

Voici un exemple de `.npmrc` fichier après avoir suivi les instructions précédentes pour définir le point de terminaison CodeArtifact du registre, ajouter un jeton d'authentification et configurer `always-auth`.

```
registry=https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my-  
cli-repo/  
//my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/  
my_repo/:_authToken=eyJ2ZX...  
//my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/:always-  
auth=true
```

Exécution de commandes npm

Après avoir configuré le client npm, vous pouvez exécuter les commandes npm. En supposant qu'un package soit présent dans votre dépôt ou dans l'un de ses référentiels en amont, vous pouvez l'installer avec `npm install`. Par exemple, utilisez ce qui suit pour installer le `lodash` package.

```
npm install lodash
```

Utilisez la commande suivante pour publier un nouveau package npm dans un CodeArtifact référentiel.

```
npm publish
```

Pour plus d'informations sur la création de packages npm, voir [Création de modules Node.js](#) sur le site Web de documentation de npm. Pour une liste des commandes npm prises en charge par CodeArtifact, consultez [npm Command Support](#).

Vérification de l'authentification et de l'autorisation npm

L'invocation de la `npm ping` commande permet de vérifier les points suivants :

- Vous avez correctement configuré vos informations d'identification afin de pouvoir vous authentifier auprès d'un CodeArtifact référentiel.
- La configuration d'autorisation vous accorde l'`ReadFromRepository` autorisation.

Le résultat d'un appel réussi de `npm ping` ressemble à ce qui suit.

```
$ npm -d ping  
npm info it worked if it ends with ok  
npm info using npm@6.4.1  
npm info using node@v9.5.0  
npm info attempt registry request try #1 at 4:30:59 PM
```

```
npm http request GET https://<domain>.d.codeartifact.us-west-2.amazonaws.com/npm/  
shared/-/ping?write=true  
npm http 200 https://npm/shared/-/ping?write=true  
Ping success: {}  
npm timing npm Completed in 716ms  
npm info ok
```

L'-option oblige npm à imprimer des informations de débogage supplémentaires, y compris l'URL du référentiel. Ces informations permettent de confirmer facilement que npm est configuré pour utiliser le référentiel que vous attendez.

Revenir au registre npm par défaut

La configuration de npm with CodeArtifact définit le registre npm sur le référentiel spécifié CodeArtifact. Vous pouvez exécuter la commande suivante pour rétablir le registre npm sur son registre par défaut lorsque vous avez terminé de vous connecter à CodeArtifact.

```
npm config set registry https://registry.npmjs.com/
```

Résolution des problèmes d'installation lents avec npm 8.x ou supérieur

Il existe un problème connu dans les versions 8.x et supérieures de npm : si une demande est envoyée à un référentiel de packages et que le référentiel redirige le client vers Amazon S3 au lieu de diffuser directement les actifs, le client npm peut se bloquer pendant plusieurs minutes par dépendance.

Les CodeArtifact référentiels étant conçus pour toujours rediriger la demande vers Amazon S3, ce problème se produit parfois, ce qui entraîne de longs délais de construction en raison des longs délais d'installation de npm. Les instances de ce comportement se présenteront sous la forme d'une barre de progression affichée pendant plusieurs minutes.

Pour éviter ce problème, utilisez les `progress=false` drapeaux `--no-progress` or avec les commandes npm cli, comme indiqué dans l'exemple suivant.

```
npm install lodash --no-progress
```

Configurer et utiliser Yarn avec CodeArtifact

Après avoir créé un dépôt, vous pouvez utiliser le client Yarn pour gérer les packages npm.

Note

Yarn 1.Xlit et utilise les informations de votre fichier de configuration npm (.npmrc), alors que ce n'est pas le cas. Yarn 2.X La configuration pour Yarn 2.X doit être définie dans le fichier .yarnrc.yml.

Table des matières

- [Configurez Yarn 1.X avec la commande aws codeartifact login](#)
- [Configurez Yarn 2.X avec la commande yarn config set](#)

Configurez Yarn 1.X avec la commande **aws codeartifact login**

CarYarn 1.X, vous pouvez configurer Yarn à CodeArtifact l'aide de la `aws codeartifact login` commande. La `login` commande configurera votre fichier `~/.npmrc` avec les informations et les informations d'identification du point de terminaison de votre CodeArtifact référentiel. AvecYarn 1.X, `yarn` les commandes utilisent les informations de configuration du fichier `~/.npmrc`.

Pour configurer à l'**Yarn 1.X** aide de la commande de connexion

1. Si vous ne l'avez pas déjà fait, configurez vos AWS informations d'identification pour les utiliser avec AWS CLI, comme décrit dans[Commencer avec CodeArtifact](#).
2. Pour exécuter correctement la `aws codeartifact login` commande, npm doit être installé. Voir [Téléchargement et installation de Node.js et de npm](#) dans la documentation de npm pour les instructions d'installation.
3. Utilisez la `aws codeartifact login` commande pour récupérer les informations d'CodeArtifact identification et configurer votre fichier `~/.npmrc`.
 - *my_domain*Remplacez-le par votre nom de CodeArtifact domaine.
 - Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure--domain-owner. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
 - *my_repo*Remplacez-le par le nom de votre CodeArtifact dépôt.

```
aws codeartifact login --tool npm --domain my_domain --domain-owner 111122223333 --repository my_repo
```

La `login` commande apporte les modifications suivantes à votre fichier `~/.npmrc` :

- Ajoute un jeton d'autorisation après l'avoir récupéré à l'aide de vos AWS informations d'identification.
- Définit le registre npm sur le référentiel spécifié par l'`--repository` option.
- Pour npm 6 et versions antérieures : ajoute le jeton d'autorisation "always-auth=true" afin que le jeton d'autorisation soit envoyé pour chaque commande npm.

La période d'autorisation par défaut après `login` un appel est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la `login` commande](#).

4. Pour npm 7.X et 8.X, vous devez ajouter `always-auth=true` à votre fichier `~/.npmrc` pour utiliser Yarn.
 - Ouvrez votre fichier `~/.npmrc` dans un éditeur de texte et ajoutez-y `always-auth=true` une nouvelle ligne.

Vous pouvez utiliser la `yarn config list` commande pour vérifier que Yarn utilise la bonne configuration. Après avoir exécuté la commande, vérifiez les valeurs indiquées dans la `info npm config` section. Le contenu doit ressembler à l'extrait suivant.

```
info npm config
{
  registry: 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/
my_repo/',
  '//my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/
my_repo/:_authToken': 'eyJ2ZXI...',
  'always-auth': true
}
```

Configurez Yarn 2.X avec la commande `yarn config set`

La procédure suivante explique comment configurer Yarn 2.X en mettant à jour votre `.yarnrc.yml` configuration à partir de la ligne de commande avec la `yarn config set` commande.

Pour mettre à jour la `yarnrc.yml` configuration depuis la ligne de commande

1. Si vous ne l'avez pas déjà fait, configurez vos AWS informations d'identification pour les utiliser avec AWS CLI, comme décrit dans [Commencer avec CodeArtifact](#).
2. Utilisez la `aws codeartifact get-repository-endpoint` commande pour obtenir le point de terminaison de votre CodeArtifact dépôt.
 - *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
 - Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
 - *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format npm
```

3. Mettez à jour la `npmRegistryServer` valeur de votre fichier `.yarnrc.yml` avec le point de terminaison de votre référentiel.

```
yarn config set npmRegistryServer  
"https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"
```

4. Récupérez un jeton CodeArtifact d'autorisation et stockez-le dans une variable d'environnement.

Note

La commande suivante est destinée aux machines macOS ou Linux. Pour plus d'informations sur la configuration des variables d'environnement sur un ordinateur Windows, consultez [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#).

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

5. Utilisez la `yarn config set` commande pour ajouter votre jeton CodeArtifact d'authentification à votre fichier `.yarnrc.yml`. Remplacez l'URL de la commande suivante par l'URL du point de terminaison de votre référentiel à l'étape 2.

```
yarn config set
  'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
  "${CODEARTIFACT_AUTH_TOKEN}"
```

6. Utilisez la `yarn config set` commande pour définir la valeur de `npmAlwaysAuth` à `true`. Remplacez l'URL de la commande suivante par l'URL du point de terminaison de votre référentiel à l'étape 2.

```
yarn config set
  'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAlwaysAuth'
  "true"
```

Après la configuration, votre fichier de configuration `.yarnrc.yml` doit avoir un contenu similaire à celui de l'extrait suivant.

```
npmRegistries:
  "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/":
  npmAlwaysAuth: true
  npmAuthToken: eyJ2ZXI...
```

```
npmRegistryServer: "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/"
```

Vous pouvez également utiliser la `yarn config` commande pour vérifier les valeurs de `npmRegistries` et `npmRegistryServer`.

support des commandes npm

Les sections suivantes résument les commandes npm prises en charge par les CodeArtifact référentiels, en plus des commandes spécifiques qui ne sont pas prises en charge.

Table des matières

- [Commandes prises en charge qui interagissent avec un référentiel](#)
- [Commandes côté client prises en charge](#)
- [Commandes non prises en charge](#)

Commandes prises en charge qui interagissent avec un référentiel

Cette section répertorie les commandes npm dans lesquelles le client npm envoie une ou plusieurs requêtes au registre avec lequel il a été configuré (par exemple, avec `npm config set registry`). Il a été vérifié que ces commandes fonctionnent correctement lorsqu'elles sont invoquées dans un CodeArtifact dépôt.

Command	Description
punaises	Essaie de deviner l'emplacement de l'URL de suivi des bogues d'un package, puis essaie de l'ouvrir.
ci	Installe un projet sur une table rase.
déprécier	Déprécie une version d'un package.
dist-tag	Modifie les balises de distribution des packages.
documents	Essaie de deviner l'emplacement de l'URL de documentation d'un package, puis essaie de

Command	Description
	l'ouvrir à l'aide du paramètre de <code>--browser</code> configuration.
<u>médecin</u>	Exécute un ensemble de vérifications pour s'assurer que votre installation npm dispose de ce dont elle a besoin pour gérer vos JavaScript packages.
<u>installer</u>	Installe un package.
<u>install-ci-test</u>	Installe un projet sur une table rase et exécute des tests. Pseudonyme : <code>npm ci</code> . Cette commande exécute un et immédiatement <code>npm ci</code> suivis d'un <code>npm test</code> .
<u>installation-test</u>	Installe le package et exécute des tests. Fonctionne et est <code>npm install</code> suivi immédiatement par <code>unnpm test</code> .
<u>désuet</u>	Vérifie le registre configuré pour voir si les packages installés sont actuellement obsolètes.
<u>ping</u>	Envoie une requête ping au registre npm configuré ou donné et vérifie l'authentification.
<u>publier</u>	Publie une version du package dans le registre.
<u>update</u>	Devine l'emplacement de l'URL du dépôt d'un package, puis essaie de l'ouvrir à l'aide du paramètre de <code>--browser</code> configuration.
<u>vue</u>	Affiche les métadonnées du package. Peut être utilisé pour imprimer les propriétés des métadonnées.

Commandes côté client prises en charge

Ces commandes ne nécessitent aucune interaction directe avec un dépôt, il n'est donc pas nécessaire de faire quoi que ce soit pour les prendre en charge.

Command	Description
<u>construire</u>	Construit un package.
<u>cache</u>	Manipule le cache des packages.
<u>achèvement</u>	Permet de compléter les onglets dans toutes les commandes npm.
<u>configuration</u>	Met à jour le contenu des <code>npmrc</code> fichiers utilisateur et globaux.
<u>dédupliquez</u>	Effectue une recherche dans l'arborescence locale des packages et tente de simplifier la structure en déplaçant les dépendances plus haut dans l'arborescence, où elles peuvent être partagées plus efficacement par plusieurs packages dépendants.
<u>modifier</u>	Modifie un package installé. Sélectionne une dépendance dans le répertoire de travail actuel et ouvre le dossier du package dans l'éditeur par défaut.
<u>explorez</u>	Parcourt un package installé. Génère un sous-shell dans le répertoire du package installé spécifié. Si une commande est spécifiée, elle est exécutée dans le sous-shell, qui s'arrête alors immédiatement.
<u>help</u>	Obtient de l'aide sur npm.
<u>aide-recherche</u>	Recherche la documentation d'aide de npm.

Command	Description
<u>initialisation</u>	Crée un package.json fichier.
<u>lien</u>	Établit des liens symboliques vers un dossier de package.
<u>ls</u>	Répertorie les packages installés.
<u>emballer</u>	Crée une archive tar à partir d'un package.
<u>prefix</u>	Affiche le préfixe. Il s'agit du répertoire parent le plus proche à contenir un package.json fichier, sauf indication contraire -g.
<u>pruneau</u>	Supprime les packages qui ne figurent pas dans la liste des dépendances du package parent.
<u>reconstruire</u>	Exécute la npm build commande sur les dossiers correspondants.
<u>redémarrer</u>	Exécute les scripts d'arrêt, de redémarrage et de démarrage d'un package ainsi que les pré-scripts et post-scripts associés.
<u>racine</u>	Imprime le node_modules dossier effectif en sortie standard.
<u>exécuter un script</u>	Exécute des scripts de package arbitraires.
<u>shrinkwrap</u>	Verrouille les versions de dépendance pour publication.
<u>désinstaller</u>	Désinstalle un package.

Commandes non prises en charge

Ces commandes npm ne sont pas prises en charge par les CodeArtifact référentiels.

Command	Description	Remarques
<u>accès</u>	Définit le niveau d'accès aux packages publiés.	CodeArtifact utilise un modèle d'autorisation différent de celui du référentiel public npmjs.
<u>ajouter un utilisateur</u>	Ajoute un compte utilisateur au registre	CodeArtifact utilise un modèle utilisateur différent du référentiel public npmjs.
<u>audit</u>	Exécute un audit de sécurité.	CodeArtifact ne vend actuellement pas de données relatives aux failles de sécurité.
<u>crochet</u>	Gère les hooks npm, y compris l'ajout, la suppression, la liste et la mise à jour.	CodeArtifact ne prend actuellement en charge aucun type de mécanisme de notification des modifications.
<u>login</u>	Authentifie un utilisateur. Ceci est un alias pour npm adduser.	CodeArtifact utilise un modèle d'authentification différent de celui du dépôt public npmjs. Pour plus d'informations, voir <u>Authentification avec npm</u> .
<u>logout</u>	Se déconnecte du registre.	CodeArtifact utilise un modèle d'authentification différent de celui du dépôt public npmjs. Il n'est pas possible de se déconnecter d'un CodeArtifact référentiel, mais les jetons d'authentification expirent après leur date d'expiration configurable. La durée du jeton par défaut est de 12 heures.

Command	Description	Remarques
<u>propriétaire</u>	Gère les propriétaires de packages.	CodeArtifact utilise un modèle d'autorisations différent de celui du référentiel public npmjs.
<u>profile</u>	Modifie les paramètres de votre profil de registre.	CodeArtifact utilise un modèle utilisateur différent du référentiel public npmjs.
<u>search</u>	Recherche dans le registre les packages correspondant aux termes de recherche.	CodeArtifact prend en charge une fonctionnalité de recherche limitée avec la <u>commande list-packages</u> .
<u>étoile</u>	Marque vos packages préférés.	CodeArtifact ne prend actuellement en charge aucun type de mécanisme de favoris.
<u>étoiles</u>	Affiche les packages marqués comme favoris.	CodeArtifact ne prend actuellement en charge aucun type de mécanisme de favoris.
<u>équipe</u>	Gère les équipes de l'organisation et les adhésions aux équipes.	CodeArtifact utilise un modèle d'adhésion d'utilisateurs et de groupes différent de celui du référentiel public npmjs. Pour plus d'informations, consultez la section <u>Identités (utilisateurs, groupes et rôles)</u> dans le guide de l'utilisateur IAM.
https://docs.npmjs.com/cli/ <u>token</u>	Gère vos jetons d'authentification.	CodeArtifact utilise un modèle différent pour obtenir des jetons d'authentification. Pour plus d'informations, voir <u>Authentification avec npm</u> .

Command	Description	Remarques
<u>dépublier</u>	Supprime un package du registre.	CodeArtifact ne prend pas en charge la suppression d'une version de package d'un référentiel à l'aide du client npm. Vous pouvez utiliser la commande <u>delete-package-version</u> .
<u>whoami</u>	Affiche le nom d'utilisateur npm.	CodeArtifact utilise un modèle utilisateur différent du référentiel public npmjs.

gestion des balises npm

Les registres npm prennent en charge les balises, qui sont des alias de chaîne pour les versions de packages. Vous pouvez utiliser des balises pour fournir un alias au lieu de numéros de version. Par exemple, vous pouvez avoir un projet comportant plusieurs flux de développement et utiliser une balise différente (par exemple `stable`, `beta`, `dev`, `canary`) pour chaque flux. Pour plus d'informations, consultez [dist-tag sur le site Web de npm](#).

Par défaut, npm utilise la `latest` balise pour identifier la version actuelle d'un package. `npm install pkg`(sans `@version` ou `@tag` spécificateur) installe la dernière balise. Généralement, les projets utilisent la dernière balise uniquement pour les versions stables. D'autres balises sont utilisées pour les versions instables ou préliminaires.

Modifier les balises avec le client npm

Les trois npm `dist-tag` commandes (`add`, `rm`, et `ls`) fonctionnent de la même manière dans CodeArtifact les référentiels que dans le registre [npm par défaut](#).

les balises npm et l'API CopyPackageVersions

Lorsque vous utilisez l'CopyPackageVersions API pour copier une version de package npm, toutes les balises aliasant cette version sont copiées dans le référentiel de destination. Lorsqu'une version copiée possède une balise également présente dans la destination, l'opération de copie

définit la valeur de la balise dans le référentiel de destination pour qu'elle corresponde à la valeur du référentiel source.

Par exemple, supposons que le référentiel S et le référentiel D contiennent une seule version du `web-helper` package avec le dernier ensemble de balises, comme indiqué dans ce tableau.

Référentiel.	Nom du package	Étiquettes du package
S	<code>web-helper</code>	dernier (alias pour la version 1.0.1)
D	<code>web-helper</code>	dernier (alias pour la version 1.0.0)

`CopyPackageVersion` est invoqué pour copier la version `web-helper` 1.0.1 de S vers D. Une fois l'opération terminée, le `latest` tag activé `web-helper` dans le dépôt D prend un alias 1.0.1, et non 1.0.0.

Si vous devez modifier les balises après la copie, utilisez la `npm dist-tag` commande pour modifier les balises directement dans le référentiel de destination. Pour plus d'informations sur l'`CopyPackageVersions` API, consultez [Copier des packages entre des référentiels](#).

balises npm et référentiels en amont

Lorsque npm demande les balises d'un package et que les versions de ce package sont également présentes dans un référentiel en amont, CodeArtifact fusionne les balises avant de les renvoyer au client. Par exemple, un référentiel nommé R possède un référentiel en amont nommé U. Le tableau suivant indique les balises d'un package nommé `web-helper` présent dans les deux référentiels.

Référentiel.	Nom du package	Étiquettes du package
R	<code>web-helper</code>	dernier (alias pour la version 1.0.0)
U	<code>web-helper</code>	alpha (alias pour la version 1.0.1)

Dans ce cas, lorsque le client npm récupère les balises du `web-helper` package dans le référentiel R, il reçoit à la fois les balises les plus récentes et les balises alpha. Les versions vers lesquelles pointent les balises ne changeront pas.

Lorsque la même balise est présente sur le même package dans le référentiel en amont et en aval, CodeArtifact utilise la balise présente dans le référentiel en amont. Supposons, par exemple, que les balises de `webhelper` aient été modifiées pour ressembler à ce qui suit.

Référentiel.	Nom du package	Étiquettes du package
R	<code>web-helper</code>	dernier (alias pour la version 1.0.0)
U	<code>web-helper</code>	dernier (alias pour la version 1.0.1)

Dans ce cas, lorsque le client npm récupère les balises du package `web-helper` à partir du référentiel R, la dernière balise alias la version 1.0.1 car c'est ce qui se trouve dans le référentiel en amont. Cela permet de consommer facilement de nouvelles versions de package dans un référentiel en amont qui ne sont pas encore présentes dans un référentiel en aval lors de l'exécution `npm update`.

L'utilisation de la balise dans le référentiel en amont peut s'avérer problématique lors de la publication de nouvelles versions d'un package dans un référentiel en aval. Par exemple, supposons que la dernière balise du package `web-helper` soit la même en R et en U.

Référentiel.	Nom du package	Étiquettes du package
R	<code>web-helper</code>	dernier (alias pour la version 1.0.1)
U	<code>web-helper</code>	dernier (alias pour la version 1.0.1)

Lorsque la version 1.0.2 est publiée sur R, npm met à jour la dernière balise vers 1.0.2.

Référentiel	Nom du package	Étiquettes du package
R	web-helper	dernier (alias pour la version 1.0.2)
U	web-helper	dernier (alias pour la version 1.0.1)

Cependant, le client npm ne voit jamais cette valeur de balise car la valeur de la dernière version de U est 1.0.1. L'exécution npm `install` sur le référentiel R immédiatement après la publication de la version 1.0.2 installe la version 1.0.1 au lieu de la version qui vient d'être publiée. Pour installer la dernière version publiée, vous devez spécifier la version exacte du package, comme suit.

```
npm install web-helper@1.0.2
```

Support pour les gestionnaires de paquets compatibles npm

Ces autres gestionnaires de packages sont compatibles CodeArtifact et fonctionnent avec le format de package npm et le protocole npm wire :

- [gestionnaire de paquets pnpm](#). La dernière version dont la compatibilité a été confirmée CodeArtifact est la 3.3.4, publiée le 18 mai 2019.
- [Gestionnaire de packages Yarn](#). La dernière version dont la compatibilité a été confirmée CodeArtifact est la 1.21.1, publiée le 11 décembre 2019.

Note

Nous vous recommandons d'utiliser Yarn 2.x avec CodeArtifact. Yarn 1.x ne propose pas de nouvelles tentatives HTTP, ce qui signifie qu'il est plus vulnérable aux pannes de service intermittentes qui entraînent des codes d'état ou des erreurs de niveau 500. Il n'y a aucun moyen de configurer une stratégie de nouvelle tentative différente pour Yarn 1.x, mais cela a été ajouté dans Yarn 2.x. Vous pouvez utiliser Yarn 1.x, mais vous devrez peut-être ajouter des tentatives de niveau supérieur dans les scripts de compilation. Par exemple, exécutez votre commande `yarn` en boucle afin qu'elle réessaie en cas d'échec du téléchargement des packages.

Utilisation CodeArtifact avec NuGet

Ces rubriques décrivent comment consommer et publier des NuGet packages à l'aide de CodeArtifact.

Note

AWS CodeArtifact ne prend en charge que les [versions NuGet .exe 4.8](#) et supérieures.

Rubriques

- [Utilisation CodeArtifact avec Visual Studio](#)
- [À utiliser CodeArtifact avec la CLI nuget ou dotnet](#)
- [NuGet normalisation du nom du package, de la version et du nom de l'actif](#)
- [NuGet compatibilité](#)

Utilisation CodeArtifact avec Visual Studio

Vous pouvez utiliser des packages CodeArtifact directement depuis Visual Studio avec le fournisseur CodeArtifact d'informations d'identification. Le fournisseur d'informations d'identification simplifie la configuration et l'authentification de vos CodeArtifact référentiels dans Visual Studio et est disponible dans le. [AWS Toolkit for Visual Studio](#)

Note

AWS Toolkit for Visual Studio Il n'est pas disponible pour Visual Studio pour Mac.

Pour configurer et utiliser NuGet avec les outils de la CLI, consultez [À utiliser CodeArtifact avec la CLI nuget ou dotnet](#).

Rubriques

- [Configurer Visual Studio avec le fournisseur d' CodeArtifact informations d'identification](#)
- [Utiliser la console Visual Studio Package Manager](#)

Configurer Visual Studio avec le fournisseur d' CodeArtifact informations d'identification

Le fournisseur CodeArtifact d'informations d'identification simplifie la configuration et l'authentification continue entre Visual Studio CodeArtifact et Visual Studio. CodeArtifact les jetons d'authentification sont valides pendant 12 heures au maximum. Pour éviter d'avoir à actualiser manuellement le jeton lorsque vous travaillez dans Visual Studio, le fournisseur d'informations d'identification récupère régulièrement un nouveau jeton avant que le jeton actuel n'expire.

Important

Pour utiliser le fournisseur d'informations d'identification, assurez-vous que toutes les AWS CodeArtifact informations d'identification existantes sont effacées de votre nuget .config fichier et qu'elles ont peut-être été ajoutées manuellement ou en exécutant `aws codeartifact login` la configuration NuGet précédente.

Utilisation CodeArtifact dans Visual Studio avec AWS Toolkit for Visual Studio

1. Installez le AWS Toolkit for Visual Studio en suivant les étapes ci-dessous. Le kit d'outils est compatible avec Visual Studio 2017 et 2019 en suivant ces étapes. AWS CodeArtifact ne prend pas en charge Visual Studio 2015 et les versions antérieures.
 1. Le Toolkit pour Visual Studio pour Visual Studio 2017 et Visual Studio 2019 est distribué sur le [Visual Studio Marketplace](#). Vous pouvez également installer et mettre à jour le kit d'outils dans Visual Studio en utilisant Outils > Extensions et mises à jour (Visual Studio 2017) ou Extensions > Gérer les extensions (Visual Studio 2019).
 2. Une fois le kit d'outils installé, ouvrez-le en choisissant AWS Explorer dans le menu Afficher.
2. Configurez le Toolkit for Visual Studio avec vos AWS informations d'identification en suivant les étapes décrites dans la section [Fournir des AWS informations d'identification](#) dans le guide de AWS Toolkit for Visual Studio l'utilisateur.
3. (Facultatif) Définissez le AWS profil que vous souhaitez utiliser CodeArtifact. S'il n'est pas défini, CodeArtifact il utilisera le profil par défaut. Pour définir le profil, accédez à Outils > Gestionnaire de NuGet packages > Sélectionner un CodeArtifact AWS profil.
4. Ajoutez votre CodeArtifact référentiel en tant que source de package dans Visual Studio.

1. Accédez à votre dépôt dans la fenêtre de l'AWS explorateur, cliquez avec le bouton droit de la souris et sélectionnez **Copy NuGet Source Endpoint**.
2. Utilisez la commande **Outils > Options** et faites défiler l'écran jusqu'au **Gestionnaire de NuGet packages**.
3. Sélectionnez le nœud **Package Sources**.
4. Sélectionnez **+**, modifiez le nom, collez le point de terminaison URL du référentiel copié à l'étape 3a dans la zone **Source**, puis sélectionnez **Mettre à jour**.
5. Cochez la case correspondant à la source du package que vous venez d'ajouter pour l'activer.

 **Note**

Nous vous recommandons d'ajouter une connexion externe à NuGet.org à votre CodeArtifact dépôt et de désactiver la source du package nuget.org dans Visual Studio. Lorsque vous utilisez une connexion externe, tous les packages extraits de NuGet.org seront stockés dans votre CodeArtifact dépôt. Si NuGet.org devient indisponible, les dépendances de votre application seront toujours disponibles pour les builds CI et le développement local. Pour plus d'informations sur les connexions externes, consultez [Connect un CodeArtifact dépôt à un dépôt public](#).

5. Redémarrez Visual Studio pour que les modifications prennent effet.

Après la configuration, Visual Studio peut consommer des packages depuis votre CodeArtifact dépôt, n'importe lequel de ses référentiels en amont ou depuis [NuGet.org](#) si vous avez ajouté une connexion externe. Pour plus d'informations sur la navigation et l'installation de NuGet packages dans Visual Studio, voir [Installer et gérer des packages dans Visual Studio à l'aide NuGet du gestionnaire](#) de packages dans la NuGet documentation.

Utiliser la console Visual Studio Package Manager

La console Visual Studio Package Manager n'utilisera pas la version Visual Studio du fournisseur d'CodeArtifact informations d'identification. Pour l'utiliser, vous devez configurer le fournisseur d'informations d'identification en ligne de commande. Pour plus d'informations, consultez [À utiliser CodeArtifact avec la CLI nuget ou dotnet](#).

À utiliser CodeArtifact avec la CLI nuget ou dotnet

Vous pouvez utiliser des outils CLI tels que nuget et dotnet pour publier et consommer des packages à partir de CodeArtifact. Ce document fournit des informations sur la configuration des outils CLI et leur utilisation pour publier ou consommer des packages.

Rubriques

- [Configuration de la CLI nuget ou dotnet](#)
- [Consommez NuGet des colis provenant de CodeArtifact](#)
- [Publiez NuGet des packages sur CodeArtifact](#)
- [CodeArtifact NuGet Référence du fournisseur d'informations d'identification](#)
- [CodeArtifact NuGet Versions du fournisseur d'informations d'identification](#)

Configuration de la CLI nuget ou dotnet

Vous pouvez configurer la CLI nuget ou dotnet avec le fournisseur CodeArtifact NuGet d'informations d'identification, avec ou manuellement AWS CLI. La configuration NuGet avec le fournisseur d'informations d'identification est vivement recommandée pour une configuration simplifiée et une authentification continue.

Méthode 1 : Configuration avec le fournisseur CodeArtifact NuGet d'informations d'identification

Le fournisseur CodeArtifact NuGet d'informations d'identification simplifie l'authentification et la configuration à l' CodeArtifact aide des outils NuGet CLI. CodeArtifact les jetons d'authentification sont valides pendant 12 heures au maximum. Pour éviter d'avoir à actualiser manuellement le jeton lors de l'utilisation de la CLI nuget ou dotnet, le fournisseur d'informations d'identification récupère régulièrement un nouveau jeton avant l'expiration du jeton actuel.

Important

Pour utiliser le fournisseur d'informations d'identification, assurez-vous que toutes les AWS CodeArtifact informations d'identification existantes sont effacées de votre nuget.config fichier et qu'elles ont peut-être été ajoutées manuellement ou en exécutant aws codeartifact login la configuration NuGet précédente.

Installation et configuration du fournisseur d' CodeArtifact NuGet informations d'identification

dotnet

1. Téléchargez la dernière version d'[AWS. CodeArtifact. NuGet. CredentialProvider outil de NuGet .org](#) avec la dotnet commande suivante.

```
dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
```

2. Utilisez la `codeartifact-creds install` commande pour copier le fournisseur d'informations d'identification dans le dossier NuGet des plugins.

```
dotnet codeartifact-creds install
```

3. (Facultatif) : Définissez le AWS profil que vous souhaitez utiliser avec le fournisseur d'informations d'identification. S'il n'est pas défini, le fournisseur d'informations d'identification utilisera le profil par défaut. Pour plus d'informations sur les AWS CLI profils, consultez la section [Profils nommés](#).

```
dotnet codeartifact-creds configure set profile profile_name
```

nuget

Procédez comme suit pour utiliser la NuGet CLI afin d'installer le fournisseur CodeArtifact NuGet d'informations d'identification à partir d'un compartiment Amazon S3 et de le configurer. Le fournisseur d'informations d'identification utilisera le AWS CLI profil par défaut. Pour plus d'informations sur les profils, voir [Profils nommés](#).

1. Téléchargez la dernière version du [fournisseur CodeArtifact NuGet d'informations d'identification \(codeartifact-nuget-credentialprovider.zip\)](#) depuis un compartiment Amazon S3.

Pour consulter et télécharger les versions antérieures, voir [CodeArtifact NuGet Versions du fournisseur d'informations d'identification](#).

2. Décompressez le fichier.
3. Copiez l'AWS. CodeArtifact. NuGetCredentialProvider dossier du dossier netfx vers `%user_profile%/.nuget/plugins/netfx/` Windows, `~/.nuget/plugins/netfx` Linux ou macOS.

4. Copiez l'AWS. CodeArtifact. NuGetCredentialProviderdossier du dossier netcore vers %user_profile%/.nuget/plugins/netcore/ Windows, ~/ .nuget/plugins/netcore Linux ou macOS.

Après avoir créé un référentiel et configuré le fournisseur d'informations d'identification, vous pouvez utiliser les outils nuget ou dotnet CLI pour installer et publier des packages. Pour plus d'informations, consultez [Consommez NuGet des colis provenant de CodeArtifact](#) et [Publiez NuGet des packages sur CodeArtifact](#).

Méthode 2 : configurer nuget ou dotnet avec la commande de connexion

La `codeartifact login` commande du AWS CLI ajoute un point de terminaison du référentiel et un jeton d'autorisation à votre fichier NuGet de configuration, permettant à Nuget ou Dotnet de se connecter à votre CodeArtifact référentiel. Cela modifiera la NuGet configuration au niveau utilisateur qui se trouve dans `%appdata%\NuGet\NuGet.Config` pour Windows `~/.config/NuGet/NuGet.Config` et/ou `~/.nuget/NuGet/NuGet.Config` pour Mac/Linux. Pour plus d'informations sur les NuGet configurations, consultez la section [NuGet Configurations courantes](#).

Configurez nuget ou dotnet avec la commande **login**

1. Configurez vos AWS informations d'identification à utiliser avec le AWS CLI, comme décrit dans [Commencer avec CodeArtifact](#).
2. Assurez-vous que l'outil NuGet CLI (nugettoudotnet) a été correctement installé et configuré. Pour obtenir des instructions, consultez la documentation de [Nuget](#) ou [Dotnet](#).
3. Utilisez la CodeArtifact `login` commande pour récupérer les informations d'identification à utiliser avec NuGet.

Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

dotnet

⚠ Important

Utilisateurs de Linux et de macOS : le chiffrement n'étant pas pris en charge sur les plateformes autres que Windows, les informations d'identification que vous avez récupérées seront stockées sous forme de texte brut dans votre fichier de configuration.

```
aws codeartifact login --tool dotnet --domain my_domain --domain-owner 111122223333 --repository my_repo
```

nuget

```
aws codeartifact login --tool nuget --domain my_domain --domain-owner 111122223333 --repository my_repo
```

La commande de connexion permettra de :

- Récupérez un jeton d'autorisation à CodeArtifact l'aide de vos AWS informations d'identification.
- Mettez à jour votre NuGet configuration au niveau utilisateur avec une nouvelle entrée pour la source de votre NuGet package. La source qui pointe vers le point de terminaison de votre CodeArtifact dépôt sera appelée *domain_name/repo_name*.

La période d'autorisation par défaut après l'appel `login` est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la `login` commande](#).

Après avoir créé un référentiel et configuré l'authentification, vous pouvez utiliser les clients `nuget` `dotnet`, ou `msbuild` CLI pour installer et publier des packages. Pour plus d'informations, consultez [Consommez NuGet des colis provenant de CodeArtifact](#) et [Publiez NuGet des packages sur CodeArtifact](#).

Méthode 3 : configurer nuget ou dotnet sans la commande de connexion

Pour la configuration manuelle, vous devez ajouter un point de terminaison de référentiel et un jeton d'autorisation à votre fichier de NuGet configuration pour permettre à nuget ou dotnet de se connecter à votre CodeArtifact référentiel.

Configurez manuellement nuget ou dotnet pour vous connecter à votre CodeArtifact dépôt.

1. Déterminez le point de terminaison de votre CodeArtifact dépôt à l'aide de la `get-repository-endpoint` AWS CLI commande.

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget
```

Exemple de sortie :

```
{  
  "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/nuget/my_repo/"  
}
```

2. Obtenez un jeton d'autorisation pour vous connecter à votre dépôt depuis votre gestionnaire de packages à l'aide de la `get-authorization-token` AWS CLI commande.

```
aws codeartifact get-authorization-token --domain my_domain
```

Exemple de sortie :

```
{  
  "authorizationToken": "eyJ2I...vi0w",  
  "expiration": 1601616533.0  
}
```

3. Créez l'URL complète du point de terminaison du référentiel en l'ajoutant `/v3/index.json` à l'URL renvoyée `get-repository-endpoint` à l'étape 3.
4. Configurez nuget ou dotnet pour utiliser le point de terminaison du référentiel de l'étape 1 et le jeton d'autorisation de l'étape 2.

Note

L'URL source doit se terminer par `/v3/index.json` pour que nuget ou dotnet se connecte correctement à un CodeArtifact dépôt.

dotnet

Utilisateurs de Linux et de macOS : le chiffrement n'étant pas pris en charge sur les plateformes autres que Windows, vous devez ajouter l'`--store-password-in-clear-text` indicateur à la commande suivante. Notez que cela enregistrera votre mot de passe sous forme de texte brut dans votre fichier de configuration.

```
dotnet nuget add source https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/nuget/my_repo/v3/index.json --name packageSourceName --password eyJ2I...vi0w --username aws
```

Note

Pour mettre à jour une source existante, utilisez la `dotnet nuget update source` commande.

nuget

```
nuget sources add -name domain_name/repo_name -Source https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/nuget/my_repo/v3/index.json -password eyJ2I...vi0w -username aws
```

Exemple de sortie :

```
Package source with Name: domain_name/repo_name added successfully.
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

Consommez NuGet des colis provenant de CodeArtifact

Une fois que vous avez [configuré NuGet avec CodeArtifact](#), vous pouvez utiliser les NuGet packages stockés dans votre CodeArtifact référentiel ou dans l'un de ses référentiels en amont.

Pour utiliser une version de package à partir d'un CodeArtifact référentiel ou de l'un de ses référentiels en amont avec nuget ou dotnet, exécutez la commande suivante en remplaçant par `packageName` le nom du package que vous souhaitez consommer et `packageSourceName` par le nom de la source de votre CodeArtifact référentiel dans votre fichier de NuGet configuration. Si vous avez utilisé la `login` commande pour configurer votre NuGet configuration, le nom de la source est `domain_name/repo_name`.

Note

Lorsqu'un package est demandé, le NuGet client met en cache les versions de ce package qui existent. En raison de ce comportement, l'installation d'un package précédemment demandé avant que la version souhaitée ne soit disponible peut échouer. Pour éviter cet échec et installer correctement un package existant, vous pouvez soit vider le NuGet cache avant une installation avec `nuget locals all --clear` ou `dotnet nuget locals all --clear`, soit éviter d'utiliser le cache pendant `restore` les commandes `install` et en fournissant l'`-NoCache` option pour `nuget` ou l'`--no-cache` option pour `dotnet`.

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

Pour installer une version spécifique d'un package

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

Consultez [Gérer les packages à l'aide de la CLI nuget.exe](#) ou [Installer et gérer les packages à l'aide de la CLI dotnet](#) dans la documentation Microsoft pour plus d'informations.

Consommez NuGet des packages provenant de NuGet.org

Vous pouvez utiliser des NuGet packages provenant de [NuGet.org](#) via un CodeArtifact référentiel en configurant le référentiel avec une connexion externe à NuGet.org. Les paquets consommés à partir de NuGet.org sont ingérés et stockés dans votre CodeArtifact dépôt. Pour plus d'informations sur l'ajout de connexions externes, consultez [Connect un CodeArtifact dépôt à un dépôt public](#).

Publiez NuGet des packages sur CodeArtifact

Une fois que vous avez [configuré NuGet avec CodeArtifact](#), vous pouvez utiliser nuget ou dotnet publier des versions de package dans CodeArtifact des référentiels.

Pour transférer une version de package vers un CodeArtifact référentiel, exécutez la commande suivante en indiquant le chemin complet de votre .nupkg fichier et le nom de la source de votre CodeArtifact référentiel dans votre fichier NuGet de configuration. Si vous avez utilisé la login commande pour configurer votre NuGet configuration, le nom de la source estdomain_name/repo_name.

Note

Vous pouvez créer un NuGet package si vous n'en avez pas à publier. Pour plus d'informations, consultez la section [Flux de travail de création de packages](#) dans la documentation Microsoft.

dotnet

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

nuget

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg -Source packageSourceName
```

CodeArtifact NuGet Référence du fournisseur d'informations d'identification

Le fournisseur CodeArtifact NuGet d'informations d'identification facilite la configuration et l'authentification NuGet auprès de vos CodeArtifact référentiels.

CodeArtifact NuGet Commandes du fournisseur d'informations d'identification

Cette section inclut la liste des commandes pour le fournisseur CodeArtifact NuGet d'informations d'identification. Ces commandes doivent être préfixées par un préfixe, `dotnet codeartifact-creds` comme dans l'exemple suivant.

```
dotnet codeartifact-creds command
```

- `configure set profile profile`: Configure le fournisseur d'informations d'identification pour qu'il utilise le profil fourni AWS .
- `configure unset profile`: Supprime le profil configuré s'il est défini.
- `install`: copie le fournisseur d'informations d'identification `plugins` dans le dossier.
- `install --profile profile`: copie le fournisseur d'informations d'identification `plugins` dans le dossier et le configure pour utiliser le profil fourni AWS .
- `uninstall`: désinstalle le fournisseur d'informations d'identification. Cela ne supprime pas les modifications apportées au fichier de configuration.
- `uninstall --delete-configuration`: désinstalle le fournisseur d'informations d'identification et supprime toutes les modifications apportées au fichier de configuration.

CodeArtifact NuGet Journaux du fournisseur d'informations d'identification

Pour activer la journalisation pour le fournisseur CodeArtifact NuGet d'informations d'identification, vous devez définir le fichier journal dans votre environnement. Les journaux du fournisseur d'informations d'identification contiennent des informations de débogage utiles, telles que :

- Le AWS profil utilisé pour établir des connexions
- Toute erreur d'authentification
- Si le point de terminaison fourni n'est pas une CodeArtifact URL

Définissez le fichier journal du fournisseur d'informations d' identification

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

Une fois le fichier journal défini, toute `codeartifact-creds` commande ajoutera sa sortie de journal au contenu de ce fichier.

CodeArtifact NuGet Versions du fournisseur d'informations d'identification

Le tableau suivant contient des informations sur l'historique des versions et des liens de téléchargement pour le fournisseur CodeArtifact NuGet d'informations d'identification.

Version	Modifications	Date de publication	Lien de téléchargement (S3)
1.0.2 (dernière version)	Dépendances améliorées	26/06/2024	Télécharger v1.0.2
1.0.1	Ajout du support pour les profils net5, net6 et SSO	05/03/2022	Télécharger v1.0.1
1.0.0	Version initiale du fournisseur CodeArtifact NuGet d'informations d'identification	20/11/2020	Télécharger v1.0.0

NuGet normalisation du nom du package, de la version et du nom de l'actif

CodeArtifact normalise les noms des packages et des actifs ainsi que les versions des packages avant de les stocker, ce qui signifie que les noms ou les versions CodeArtifact peuvent être différents de ceux fournis lors de la publication du package ou de l'actif.

Normalisation des noms de packages : CodeArtifact normalise les noms de NuGet packages en convertissant toutes les lettres en minuscules.

Normalisation des versions des packages : CodeArtifact normalise les versions des NuGet packages en utilisant le même modèle que NuGet. Les informations suivantes proviennent des [numéros de version normalisés](#) de la NuGet documentation.

- Les zéros en tête sont supprimés des numéros de version :
 - 1.00est traité comme 1.0
 - 1.01.1est traité comme 1.1.1
 - 1.00.0.1est traité comme 1.0.0.1
- Un zéro dans la quatrième partie du numéro de version sera omis :
 - 1.0.0.0est traité comme 1.0.0
 - 1.0.01.0est traité comme 1.0.1
- SemVer Les métadonnées de build 2.0.0 sont supprimées :
 - 1.0.7+r3456est traité comme 1.0.7

Normalisation du nom de l'actif du package : CodeArtifact construit le nom de l'actif NuGet du package à partir du nom du package normalisé et de la version du package.

Le nom du package et le nom de version non normalisés peuvent être utilisés avec les demandes d'API et de CLI car ils CodeArtifact normalisent le nom du package et les entrées de version pour ces demandes. Par exemple, les entrées de `--package Newtonsoft.JSON` et `--version 12.0.03.0` seraient normalisées et renverraient un package dont le nom de package `newtonsoft.json` et la version de sont normalisés. `12.0.3`

Vous devez utiliser le nom d'actif du package normalisé dans les demandes d'API et de CLI, car CodeArtifact il n'effectue pas de normalisation sur l'`--asset` entrée.

Vous devez utiliser des noms et des versions normalisés dans ARNs.

Pour trouver le nom normalisé d'un package, utilisez la `aws codeartifact list-packages` commande. Pour de plus amples informations, veuillez consulter [Lister les noms de packages](#).

Pour trouver le nom non normalisé d'un package, utilisez la `aws codeartifact describe-package-version` commande. Le nom non normalisé du package est renvoyé dans le `displayName` champ. Pour de plus amples informations, veuillez consulter [Afficher et mettre à jour les détails et les dépendances des versions du package](#).

NuGet compatibilité

Ce guide contient des informations sur CodeArtifact la compatibilité avec différents NuGet outils et versions.

Rubriques

- [NuGet Compatibilité générale](#)
- [NuGet support en ligne de commande](#)

NuGet Compatibilité générale

AWS CodeArtifact prend en charge les versions NuGet 4.8 et supérieures.

AWS CodeArtifact supporte uniquement la version 3 du protocole NuGet HTTP. Cela signifie que certaines commandes CLI qui reposent sur la version 2 du protocole ne sont pas prises en charge. Consultez la section [prise en charge des commandes nuget.exe](#) pour plus d'informations.

AWS CodeArtifact ne prend pas en charge la version PowerShellGet 2.x.

NuGet support en ligne de commande

AWS CodeArtifact prend en charge les outils CLI NuGet (nuget .exe/dotnet) et .NET Core () .

prise en charge des commandes nuget.exe

Comme elle CodeArtifact ne prend en charge que le protocole HTTP NuGet de la version 3 de, les commandes suivantes ne fonctionneront pas lorsqu'elles sont utilisées avec des CodeArtifact ressources :

- **list:** La `nuget list` commande affiche une liste de packages provenant d'une source donnée. Pour obtenir la liste des packages d'un CodeArtifact dépôt, vous pouvez utiliser la [Lister les noms de packages](#) commande de la AWS CLI.

Utilisation CodeArtifact avec Python

Ces rubriques décrivent comment utiliser `pip` le gestionnaire de packages Python et `twine` l'utilitaire de publication de packages Python avec CodeArtifact.

Rubriques

- [Configurer et utiliser pip avec CodeArtifact](#)
- [Configurer et utiliser Twine avec CodeArtifact](#)
- [Normalisation du nom des paquets Python](#)
- [Compatibilité avec Python](#)
- [Demande de packages Python depuis des connexions en amont et externes](#)

Configurer et utiliser pip avec CodeArtifact

[pip](#) est l'installateur de packages pour les packages Python. Pour utiliser pip pour installer des packages Python à partir de votre CodeArtifact dépôt, vous devez d'abord configurer le client pip avec les informations et les informations d'identification de votre CodeArtifact référentiel.

pip ne peut être utilisé que pour installer des packages Python. Pour publier des packages Python, vous pouvez utiliser [twine](#). Pour de plus amples informations, veuillez consulter [Configurer et utiliser Twine avec CodeArtifact](#).

Configurez pip avec la commande `login`

Tout d'abord, configurez vos AWS informations d'identification à utiliser avec le AWS CLI, comme décrit dans [Commencer avec CodeArtifact](#). Utilisez ensuite la CodeArtifact `login` commande pour récupérer les informations d'identification et les configurer pip avec elles.

Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

Pour configurer pip, exécutez la commande suivante.

```
aws codeartifact login --tool pip --domain my_domain --domain-owner 111122223333 --repository my_repo
```

La `login` commande récupère un jeton d'autorisation à l'CodeArtifact aide de vos AWS informations d'identification. La `login` commande sera configurée `pip` pour être utilisée CodeArtifact en éditant `~/.config/pip/pip.conf` pour définir le référentiel spécifié par l'`--repository` option.

La période d'autorisation par défaut après l'appel `login` est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la `login` commande](#).

Configurer pip sans la commande de connexion

Si vous ne pouvez pas utiliser la `login` commande pour configurer `pip`, vous pouvez utiliser `pip config`.

1. Utilisez le AWS CLI pour récupérer un nouveau jeton d'autorisation.

 Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, il n'est pas nécessaire d'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

```
CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

2. Utilisez `pip config` pour définir l'URL du CodeArtifact registre et les informations d'identification. La commande suivante ne mettra à jour que le fichier de configuration de l'environnement actuel. Pour mettre à jour le fichier de configuration à l'échelle du système, remplacez-le site par `global`.

```
pip config set site.index-url https://aws:$CODEARTIFACT_AUTH_TOKEN@my_domain-111122223333.d.codeartifact.region.amazonaws.com/pypi/my_repo/simple/
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

Important

L'URL du registre doit se terminer par une barre oblique (/). Dans le cas contraire, vous ne pourrez pas vous connecter au référentiel.

Exemple de fichier de configuration pip

Voici un exemple de `pip.conf` fichier après avoir défini l'URL du CodeArtifact registre et les informations d'identification.

```
[global]
index-url = https://aws:eyJ2ZX...@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/pypi/my_repo/simple/
```

Exécuter pip

Pour exécuter pip des commandes, vous devez configurer pip avec CodeArtifact. Pour plus d'informations, consultez la documentation suivante.

1. Suivez les étapes décrites dans la [Configuration avec AWS CodeArtifact](#) section pour configurer votre AWS compte, vos outils et vos autorisations.
2. Configurez twine en suivant les étapes décrites dans [Configurer et utiliser Twine avec CodeArtifact](#).

En supposant qu'un package soit présent dans votre dépôt ou dans l'un de ses référentiels en amont, vous pouvez l'installer avec `pip install`. Par exemple, utilisez la commande suivante pour installer le `requests` package.

```
pip install requests
```

Utilisez `-i` cette option pour revenir temporairement à l'installation de packages depuis <https://pypi.org> plutôt que depuis votre CodeArtifact dépôt.

```
pip install -i https://pypi.org/simple requests
```

Configurer et utiliser Twine avec CodeArtifact

[twine](#) est un utilitaire de publication de packages pour les packages Python. Pour utiliser twine pour publier des packages Python dans votre CodeArtifact dépôt, vous devez d'abord configurer twine avec les informations et les informations d'identification de votre CodeArtifact dépôt.

twine ne peut être utilisé que pour publier des packages Python. Pour installer des packages Python, vous pouvez utiliser [pip](#). Pour de plus amples informations, veuillez consulter [Configurer et utiliser pip avec CodeArtifact](#).

Configurer la ficelle avec la commande `login`

Tout d'abord, configurez vos AWS informations d'identification à utiliser avec le AWS CLI, comme décrit dans [Commencer avec CodeArtifact](#). Utilisez ensuite la CodeArtifact `login` commande pour récupérer les informations d'identification et configurer Twine avec elles.

Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

Pour configurer Twine, exédez la commande suivante.

```
aws codeartifact login --tool twine --domain my_domain --domain-owner 111122223333 --repository my_repo
```

`login` récupère un jeton d'autorisation à l' CodeArtifact aide de vos AWS informations d'identification. La `login` commande configure la ficelle à utiliser CodeArtifact en modifiant `~/.pypirc` pour ajouter le référentiel spécifié par l'`--repository` option avec des informations d'identification.

La période d'autorisation par défaut après l'appel `login` est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la `login` commande](#).

Configurer la ficelle sans la commande `login`

Si vous ne pouvez pas utiliser la `login` commande pour configurer Twine, vous pouvez utiliser le `~/.pypirc` fichier ou les variables d'environnement. Pour utiliser le `~/.pypirc` fichier, ajoutez-y les entrées suivantes. Le mot de passe doit être un jeton d'authentification acquis par l'`get-authorization-tokenAPI`.

```
[distutils]
index-servers =
  codeartifact
[codeartifact]
repository = https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
pypi/my_repo/
password = auth-token
username = aws
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

Pour utiliser des variables d'environnement, procédez comme suit.

Note

Si vous accédez à un référentiel dans un domaine qui vous appartient, il n'est pas nécessaire d'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

```
export TWINE_USERNAME=aws
export TWINE_PASSWORD=`aws codeartifact get-authorization-token --domain my_domain --
domain-owner 111122223333 --query authorizationToken --output text`
```

```
export TWINE_REPOSITORY_URL=`aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 1112222333 --repository my_repo --format pypi --query repositoryEndpoint --output text`
```

Run twine

Avant d'utiliser twine pour publier les actifs d'un package Python, vous devez d'abord configurer CodeArtifact les autorisations et les ressources.

1. Suivez les étapes décrites dans la [Configuration avec AWS CodeArtifact](#) section pour configurer votre AWS compte, vos outils et vos autorisations.
2. Configurez la ficelle en suivant les étapes décrites dans [Configurer la ficelle avec la commande login](#) ou [Configurer la ficelle sans la commande login](#).

Après avoir configuré Twine, vous pouvez exécuter des twine commandes. Utilisez la commande suivante pour publier les actifs du package Python.

```
twine upload --repository codeartifact mypackage-1.0.tgz
```

Pour plus d'informations sur la création et le package de votre application Python, consultez la section [Génération d'archives de distribution](#) sur le site Web de la Python Packaging Authority.

Normalisation du nom des paquets Python

CodeArtifact normalise les noms de package avant de les stocker, ce qui signifie que les noms de package CodeArtifact peuvent être différents du nom fourni lors de la publication du package.

Pour les packages Python, lors de la normalisation, le nom du package est en minuscules et toutes les instances des caractères « . - , » _ sont remplacées par un seul caractère. - Ainsi, les noms des `pigeon_cli` packages `pigeon.cli` sont normalisés et stockés sous `pigeon-cli` la forme. Le nom non normalisé peut être utilisé par pip et twine, mais le nom normalisé doit être utilisé dans les requêtes CodeArtifact CLI ou API (telles que) et dans `list-package-versions` ARNs Pour plus d'informations sur la normalisation des noms de paquets Python, consultez le [PEP 503](#) dans la documentation Python.

Compatibilité avec Python

CodeArtifact ne prend pas en charge les PyPI ou XML-RPC JSON APIs

CodeArtifact supporte PyPI, à l'exception Legacy APIs de l'simpleAPI. Bien qu' CodeArtifact il ne prenne pas en charge le point de terminaison de l'/simple/API, il prend en charge le /simple/ <project>/ point de terminaison.

Pour plus d'informations, consultez ce qui suit sur le GitHub référentiel de la Python Packaging Authority.

- [API XML-RPC](#)
- [API JSON](#)
- [API héritée](#)

support de la commande pip

Les sections suivantes résument les commandes pip prises en charge par les CodeArtifact référentiels, en plus des commandes spécifiques non prises en charge.

Rubriques

- [Commandes prises en charge qui interagissent avec un référentiel](#)
- [Commandes côté client prises en charge](#)

Commandes prises en charge qui interagissent avec un référentiel

Cette section répertorie pip les commandes dans lesquelles le pip client envoie une ou plusieurs requêtes au registre avec lequel il a été configuré. Il a été vérifié que ces commandes fonctionnent correctement lorsqu'elles sont invoquées dans un CodeArtifact dépôt.

Command	Description
installer	Installez les packages.
télécharger	Téléchargez les packages.

CodeArtifact ne met pas en œuvre `pip search`. Si vous avez configuré pip avec un CodeArtifact référentiel, l'exécution `pip search` recherchera et affichera les packages de [PyPI](#).

Commandes côté client prises en charge

Ces commandes ne nécessitent aucune interaction directe avec un dépôt, il n'est donc CodeArtifact pas nécessaire de faire quoi que ce soit pour les prendre en charge.

Command	Description
<u>désinstaller</u>	Désinstallez les packages.
<u>geler</u>	Afficher les packages installés au format des exigences.
<u>liste</u>	Répertoriez les packages installés.
<u>show</u>	Afficher les informations sur les packages installés.
<u>vérifier</u>	Vérifiez que les packages installés ont des dépendances compatibles.
<u>configuration</u>	Gérez la configuration locale et globale.
<u>roue</u>	Construisez des roues selon vos besoins.
<u>hachage</u>	Calculez les hachages des archives de packages.
<u>achèvement</u>	Aide à terminer les commandes.
<u>debug</u>	Afficher les informations utiles pour le débogage.
<u>aide</u>	Afficher l'aide pour les commandes.

Demande de packages Python depuis des connexions en amont et externes

Lorsque vous importez une version de package Python depuis pypi.org, tous les actifs de cette version de package CodeArtifact seront importés. Alors que la plupart des packages Python

contiennent un petit nombre d'actifs, certains en contiennent plus de 100, généralement pour prendre en charge plusieurs architectures matérielles et interpréteurs Python.

Il est courant que de nouveaux actifs soient publiés sur pypi.org pour une version de package existante. Par exemple, certains projets publient de nouvelles ressources lorsque de nouvelles versions de Python sont publiées. Lorsqu'un package Python est installé à partir de CodeArtifact with `pip install`, les versions du package conservées dans le CodeArtifact référentiel sont mises à jour pour refléter le dernier ensemble d'actifs de pypi.org.

De même, si de nouveaux actifs sont disponibles pour une version de package dans un CodeArtifact référentiel en amont qui ne sont pas présents dans le CodeArtifact référentiel actuel, ils seront conservés dans le référentiel actuel lors `pip install` de son exécution.

Versions du package supprimées

Certaines versions de package sur pypi.org sont marquées comme supprimées, ce qui indique à l'installateur du package (comme pip) que la version ne doit pas être installée sauf si elle est la seule à correspondre à un spécificateur de version (en utilisant l'un ou l'autre). == == Voir [PEP_592](#) pour plus d'informations.

Si une version de package CodeArtifact a été initialement récupérée à partir d'une connexion externe à [pypi.org](#), lorsque vous installez la version du package depuis un CodeArtifact dépôt, CodeArtifact assurez-vous que les métadonnées supprimées mises à jour de la version du package sont extraites de pypi.org.

Comment savoir si une version de package est supprimée

Pour vérifier si une version du package est intégrée CodeArtifact, vous pouvez essayer de l'installer avec `pip install packageName==packageVersion`. Si la version du package est supprimée, vous recevez un message d'avertissement semblable au suivant :

WARNING: The candidate selected for download or install is a yanked version

Pour vérifier si une version du package est supprimée dans [pypi.org](#), vous pouvez consulter [la liste pypi.org](#) de la version du package à l'adresse suivante. [https://pypi.org/project/*packageName*/*packageVersion*/](https://pypi.org/project/<i>packageName</i>/<i>packageVersion</i>/)

Configuration du statut de retrait sur les packages privés

CodeArtifact ne prend pas en charge la définition de métadonnées supprimées pour les packages publiés directement dans les CodeArtifact référentiels.

Pourquoi ne CodeArtifact pas récupérer les dernières métadonnées ou ressources supprimées pour une version de package ?

Normalement, CodeArtifact garantit que lorsqu'une version de package Python est extraite d'un CodeArtifact dépôt, les métadonnées extraites ont la dernière valeur sur pypi.org. up-to-date En outre, la liste des actifs de la version du package est également mise à jour avec la dernière version sur pypi.org et sur tous les référentiels en amont CodeArtifact . Cela est vrai si vous installez la version du package pour la première fois et que vous l' CodeArtifact importez depuis pypi.org dans votre CodeArtifact dépôt, ou si vous avez déjà installé le package. Cependant, dans certains cas, le client du gestionnaire de packages, tel que pip, n'extrait pas les dernières métadonnées extraites de pypi.org ou des référentiels en amont. Au lieu de cela, il CodeArtifact renverra les données déjà stockées dans votre référentiel. Cette section décrit les trois manières dont cela peut se produire :

Configuration en amont : si la connexion externe à pypi.org est supprimée du référentiel ou de ses flux en amont à l'aide de pypi.org [disassociate-external-connection](#), les métadonnées supprimées ne seront plus actualisées depuis pypi.org. De même, si vous supprimez un référentiel en amont, les ressources du référentiel supprimé et des ressources en amont du référentiel supprimé ne seront plus accessibles au référentiel actuel. Il en va de même si vous utilisez les [contrôles d'origine des CodeArtifact packages](#) pour empêcher l'extraction de nouvelles versions d'un package spécifique. Ce paramètre `upstream=BLOCK` empêchera l'actualisation des métadonnées extraites.

État de la version du package : si vous définissez le statut d'une version de package sur autre chose que « Published ou »Unlisted, les métadonnées et les actifs retirés de la version du package ne seront pas actualisés. De même, si vous récupérez une version de package spécifique (par exemple `torch 2.0.1`) et que la même version de package est présente dans un référentiel en amont avec un statut qui n'est pas Published ou Unlisted, cela bloquera également les métadonnées extraites et la propagation des actifs du référentiel en amont vers le référentiel actuel. Cela est dû au fait que les autres statuts de version de package indiquent que les versions ne sont plus destinées à être consommées dans aucun référentiel.

Publication directe : si vous publiez une version de package spécifique directement dans un CodeArtifact référentiel, cela empêchera le retrait des métadonnées et l'actualisation des ressources pour la version du package depuis ses référentiels en amont et depuis pypi.org. Supposons, par

exemple, que vous téléchargez une ressource à partir de la version du package `torch 2.0.1`, par exemple `torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`, à l'aide d'un navigateur Web, puis que vous la publiez dans votre CodeArtifact référentiel à l'aide de `twine astorch 2.0.1`. CodeArtifact vérifie que la version du package est entrée dans le domaine en la publiant directement dans votre dépôt, et non à partir d'une connexion externe à `pypi.org` ou d'un dépôt en amont. Dans ce cas, les métadonnées CodeArtifact extraites ne sont pas synchronisées avec les référentiels en amont ou avec `pypi.org`. Il en va de même si vous publiez `torch 2.0.1` dans un référentiel en amont : la présence de la version du package bloquera la propagation des métadonnées et des actifs extraits vers les référentiels situés plus bas dans le graphique en amont.

Utilisation CodeArtifact avec Ruby

Ces rubriques décrivent comment utiliser les outils RubyGems et Bundler CodeArtifact pour installer et publier des gemmes Ruby.

Note

CodeArtifact recommande Ruby 3.3 ou version ultérieure et ne fonctionne pas avec Ruby 2.6 ou version antérieure.

Rubriques

- [Configurer RubyGems et utiliser un Bundler avec CodeArtifact](#)
- [RubyGems support aux commandes](#)
- [Compatibilité avec le bundler](#)

Configurer RubyGems et utiliser un Bundler avec CodeArtifact

Après avoir créé un dépôt dans CodeArtifact, vous pouvez utiliser RubyGems (gem) et Bundler (bundle) pour installer et publier des gemmes. Cette rubrique décrit comment configurer les gestionnaires de packages pour s'authentifier auprès d'un CodeArtifact référentiel et l'utiliser.

Configure RubyGems (**gem**) et Bundler (**bundle**) avec CodeArtifact

Pour utiliser RubyGems (gem) ou Bundler (bundle) pour publier des gemmes ou en consommer AWS CodeArtifact, vous devez d'abord les configurer avec les informations de votre CodeArtifact référentiel, y compris les informations d'identification pour y accéder. Suivez les étapes de l'une des procédures suivantes pour configurer les outils gem et bundle CLI avec les informations et les informations d'identification du point de terminaison de votre CodeArtifact référentiel.

Configuration RubyGems et regroupement à l'aide des instructions de la console

Vous pouvez utiliser les instructions de configuration de la console pour connecter vos gestionnaires de packages Ruby à votre CodeArtifact référentiel. Les instructions de la console fournissent des commandes personnalisées que vous pouvez exécuter pour configurer vos gestionnaires de packages sans avoir à rechercher et à saisir vos CodeArtifact informations.

1. Ouvrez la AWS CodeArtifact console sur <https://console.aws.amazon.com/codesuite/codeartifact/home>.
2. Dans le volet de navigation, choisissez Repositories, puis choisissez le référentiel que vous souhaitez utiliser pour installer ou diffuser des gemmes Ruby.
3. Choisissez Afficher les instructions de connexion.
4. Choisissez votre système d'exploitation.
5. Choisissez le client du gestionnaire de packages Ruby que vous souhaitez configurer avec votre CodeArtifact dépôt.
6. Suivez les instructions générées pour configurer le client du gestionnaire de packages afin d'installer des gemmes Ruby ou de publier des gemmes Ruby dans le référentiel.

Configuration RubyGems et regroupement manuels

Si vous ne pouvez pas ou ne souhaitez pas utiliser les instructions de configuration de la console, vous pouvez utiliser les instructions suivantes pour vous connecter manuellement à vos gestionnaires de packages Ruby à votre CodeArtifact référentiel.

1. Dans une ligne de commande, utilisez la commande suivante pour récupérer un jeton CodeArtifact d'autorisation et le stocker dans une variable d'environnement.
 - *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
 - Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

macOS and Linux

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
for /f %i in ('aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text') do set CODEARTIFACT_AUTH_TOKEN=%i
```

- Fenêtres PowerShell :

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text
```

2. Pour publier des gemmes Ruby dans votre dépôt, utilisez la commande suivante pour récupérer le point de terminaison de votre CodeArtifact dépôt et le stocker dans la variable d'RUBYGEMS_HOSTenvironnement. La gem CLI utilise cette variable d'environnement pour déterminer où les gemmes sont publiées.

 Note

Au lieu d'utiliser la variable d'RUBYGEMS_HOSTenvironnement, vous pouvez également fournir --host cette option au point de terminaison du référentiel lorsque vous utilisez la `gem push` commande.

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

macOS and Linux

```
export RUBYGEMS_HOST=`aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint --output text | sed 's://*$::'`
```

Windows

Les commandes suivantes permettent de récupérer le point de terminaison du référentiel, de découper le point final/, puis de le stocker dans une variable d'environnement.

- Windows (en utilisant l'interface de commande par défaut) :

```
for /f %i in ('aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint --output text') do set RUBYGEMS_HOST=%i  
  
set RUBYGEMS_HOST=%RUBYGEMS_HOST:~0,-1%
```

- Fenêtres PowerShell :

```
$env:RUBYGEMS_HOST = (aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint --output text).TrimEnd("/")
```

L'URL suivante est un exemple de point de terminaison du référentiel :

```
https://my\_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my\_repo/
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

3. Pour publier des gemmes Ruby dans votre dépôt, vous devez vous authentifier auprès de CodeArtifact with RubyGems en éditant votre `~/.gem/credentials` fichier pour inclure votre jeton d'authentification. Créez un `~/.gem/` répertoire et un `~/.gem/credentials` fichier si le répertoire ou le fichier n'existe pas.

macOS and Linux

```
echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/credentials
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
echo :codeartifact_api_key: Bearer %CODEARTIFACT_AUTH_TOKEN% >> %USERPROFILE%/.gem/credentials
```

- Fenêtres PowerShell :

```
echo ":codeartifact_api_key: Bearer $env:CODEARTIFACT_AUTH_TOKEN" | Add-Content ~/.gem/credentials
```

4. Pour installer des gemmes Ruby depuis votre dépôt, vous devez ajouter les informations de point de terminaison du référentiel et le jeton d'authentification à votre `.gemrc` fichier. Vous pouvez l'ajouter au fichier global (`~/.gemrc`) ou à votre `.gemrc` fichier de projet. Les CodeArtifact informations que vous devez y ajouter `.gemrc` sont une combinaison du point de terminaison du référentiel et du jeton d'authentification. Il est formaté comme suit :

```
https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/
```

- Pour le jeton d'authentification, vous pouvez utiliser la variable `CODEARTIFACT_AUTH_TOKEN` environnement définie lors d'une étape précédente.
- Pour récupérer le point de terminaison du référentiel, vous pouvez lire la valeur de la variable `RUBYGEMS_HOST` environnement définie précédemment, ou vous pouvez utiliser la `get-repository-endpoint` commande suivante, en remplaçant les valeurs si nécessaire :

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint --output text
```

Une fois que vous avez le point final, utilisez un éditeur de texte pour l'ajouter `aws :${CODEARTIFACT_AUTH_TOKEN}@` à la position appropriée. Une fois que vous avez créé le point de terminaison du référentiel et la chaîne du jeton d'authentification, ajoutez-les à la `:sources:` section de votre `.gemrc` fichier à l'aide de la `echo` commande suivante :

⚠ Warning

CodeArtifact ne prend pas en charge l'ajout de référentiels en tant que sources à l'aide de la `gem sources -add` commande. Vous devez ajouter la source directement dans le fichier.

macOS and Linux

```
echo ":sources:
- https://aws:
${CODEARTIFACT_AUTH_TOKEN}@${my_domain-111122223333}.d.codeartifact.us-
west-2.amazonaws.com/ruby/${my_repo}/" > ~/.gemrc
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
echo ":sources:
- https://aws:${CODEARTIFACT_AUTH_TOKEN}
%@${my_domain-111122223333}.d.codeartifact.us-west-2.amazonaws.com/ruby/${my_repo}/"
> "%USERPROFILE%\.gemrc"
```

- Fenêtres PowerShell :

```
echo ":sources:
- https://aws:
$env:CODEARTIFACT_AUTH_TOKEN@${my_domain-111122223333}.d.codeartifact.us-
west-2.amazonaws.com/ruby/${my_repo}/" | Add-Content ~/.gemrc
```

- Pour utiliser Bundler, vous devez configurer Bundler avec l'URL du point de terminaison et le jeton d'authentification de votre référentiel en exécutant la commande suivante : `bundle config`

macOS and Linux

```
bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
```

- Fenêtres PowerShell :

```
bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
```

Maintenant que vous avez configuré RubyGems (gem) et Bundler (bundle) avec votre CodeArtifact dépôt, vous pouvez les utiliser pour publier et consommer des gemmes Ruby depuis et vers celui-ci.

Installation de Ruby Gems depuis CodeArtifact

Utilisez les procédures suivantes pour installer les gemmes Ruby à partir d'un CodeArtifact référentiel avec les outils gem ou bundle CLI.

Installez Ruby Gems avec **gem**

Vous pouvez utiliser la CLI RubyGems (gem) pour installer rapidement une version spécifique d'une gemme Ruby depuis votre CodeArtifact dépôt.

Pour installer des gemmes Ruby à partir d'un CodeArtifact dépôt avec **gem**

1. Si ce n'est pas le cas, suivez les étapes décrites [Configure RubyGems \(gem\) et Bundler \(bundle\) avec CodeArtifact](#) pour configurer la gem CLI afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

2. Utilisez la commande suivante pour installer les gemmes Ruby depuis CodeArtifact :

```
gem install my_ruby_gem --version 1.0.0
```

Installez Ruby Gems avec **bundle**

Vous pouvez utiliser la CLI Bundler (**bundle**) pour installer les gemmes Ruby configurées dans votre `Gemfile`.

Pour installer des gemmes Ruby à partir d'un CodeArtifact dépôt avec **bundle**

1. Si ce n'est pas le cas, suivez les étapes décrites [Configure RubyGems \(gem\) et Bundler \(bundle\) avec CodeArtifact](#) pour configurer la bundle CLI afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

 Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

2. Ajoutez l'URL du point de terminaison de votre CodeArtifact dépôt à votre `Gemfile` adresse `source` pour installer des gemmes Ruby configurées à partir de votre CodeArtifact référentiel et de ses flux en amont.

```
source "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/  
ruby/my_repo/"  
  
gem 'my_ruby_gem'
```

3. Utilisez la commande suivante pour installer les gemmes Ruby comme indiqué dans votre `Gemfile` :

```
bundle install
```

Publier des gemmes Ruby sur CodeArtifact

Utilisez la procédure suivante pour publier des gemmes Ruby dans un CodeArtifact référentiel à l'aide de la gem CLI.

1. Si ce n'est pas le cas, suivez les étapes décrites [Configure RubyGems \(gem\) et Bundler \(bundle\) avec CodeArtifact](#) pour configurer la gem CLI afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

 Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

2. Utilisez la commande suivante pour publier des gemmes Ruby dans un CodeArtifact dépôt. Notez que si vous n'avez pas défini la variable d'RUBYGEMS_HOSTenvironnement, vous devez indiquer le point de terminaison de votre CodeArtifact référentiel dans l'--hostoption.

```
gem push --key codeartifact_api_key my_ruby_gem-0.0.1.gem
```

RubyGems support aux commandes

CodeArtifact prend en charge les `gem push` commandes `gem install` et. CodeArtifact ne prend pas en charge les `gem` commandes suivantes :

- `gem fetch`
- `gem info --remote`
- `gem list --remote`
- `gem mirror`
- `gem outdated`
- `gem owner`
- `gem query`
- `gem search`
- `gem signin`
- `gem signout`
- `gem sources --add`
- `gem sources --update`
- `gem specification --remote`
- `gem update`
- `gem yank`

Compatibilité avec le bundler

Ce guide contient des informations sur CodeArtifact la compatibilité avec Bundler.

Compatibilité avec le bundler

AWS CodeArtifact recommande Bundler 2.4.11 ou supérieur. Si vous rencontrez des problèmes lors de l'installation, mettez à jour la CLI Bundler vers la dernière version.

Support des versions du bundler

Dans les versions de Bundler inférieures à 2.4.11, il existe une limite de 500 dépendances qui peuvent être définies dans le Gemfile avant que Bundler ne décide d'interroger l'index complet., specs .4 .8 .gz Comme CodeArtifact il ne prend pas en charge l'index complet, la spécification de plus de 500 dépendances ne fonctionnera pas CodeArtifact lors de l'utilisation de versions de Bundler inférieures à 2.4.11.

Pour définir plus de 500 dépendances dans votre Gemfile avec CodeArtifact, mettez à jour Bundler vers la version 2.4.11 ou supérieure.

Assistance aux opérations du bundler

CodeArtifactn'inclut RubyGems pas le Bundler Compact Index APIs (l'/_versionsAPI n'est pas prise en charge). CodeArtifact ne prend en charge que l'API Dependencies.

De plus, CodeArtifact ne prend pas en charge les différentes spécifications APIs, telles que specs .4 .8 .gz.

Utilisation CodeArtifact avec Swift

Ces rubriques décrivent comment utiliser le gestionnaire de packages Swift CodeArtifact pour installer et publier des packages Swift.

Note

CodeArtifact supporte Swift 5.8 et versions ultérieures et Xcode 14.3 et versions ultérieures. CodeArtifact recommande Swift 5.9 et versions ultérieures et Xcode 15 et versions ultérieures.

Rubriques

- [Configurez le gestionnaire de packages Swift avec CodeArtifact](#)
- [Consommation et publication de packages Swift](#)
- [Normalisation rapide du nom du package et de l'espace de noms](#)
- [Résolution rapide des problèmes](#)

Configurez le gestionnaire de packages Swift avec CodeArtifact

Pour utiliser le Swift Package Manager afin de publier des packages vers ou de consommer des packages à partir de ceux-ci AWS CodeArtifact, vous devez d'abord configurer des informations d'identification pour accéder à votre CodeArtifact référentiel. La méthode recommandée pour configurer la CLI de Swift Package Manager avec vos CodeArtifact informations d'identification et le point de terminaison du référentiel consiste à utiliser la `aws codeartifact login` commande. Vous pouvez également configurer le Swift Package Manager manuellement.

Configurer Swift avec la commande de connexion

Utilisez la `aws codeartifact login` commande pour configurer le Swift Package Manager avec CodeArtifact.

Note

Pour utiliser la commande de connexion, Swift 5.8 ou version ultérieure est requis et Swift 5.9 ou version ultérieure est recommandé.

La `aws codeartifact login` commande effectuera les opérations suivantes :

1. Récupérez un jeton d'authentification CodeArtifact et stockez-le dans votre environnement. La manière dont les informations d'identification sont stockées dépend du système d'exploitation de l'environnement :
 - a. macOS : une entrée est créée dans l'application macOS Keychain.
 - b. Linux et Windows : une entrée est créée dans le `~/.netrc` fichier.

Dans tous les systèmes d'exploitation, s'il existe une entrée d'informations d'identification, cette commande remplace cette entrée par un nouveau jeton.

2. Récupérez l'URL du point de terminaison de votre CodeArtifact dépôt et ajoutez-la à votre fichier de configuration Swift. La commande ajoute l'URL du point de terminaison du référentiel au fichier de configuration au niveau du projet situé à `l'adresse/path/to/project/.swiftpm/configuration/registries.json`.

 Note

La `aws codeartifact login` commande appelle `swift package-registry` des commandes qui doivent être exécutées à partir du répertoire contenant le `Package.swift` fichier. Pour cette raison, `aws codeartifact login` la commande doit être exécutée depuis le projet Swift.

Pour configurer Swift à l'aide de la commande de connexion

1. Accédez au répertoire du projet Swift qui contient le `Package.swift` fichier de votre projet.
2. Exécutez la commande suivante `aws codeartifact login`.

Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

```
aws codeartifact login --tool swift --domain my_domain \
--domain-owner 111122223333 --repository my_repo \
[--namespace my_namespace]
```

L'--namespace option configure l'application pour qu'elle ne consomme les packages de votre CodeArtifact dépôt que s'ils se trouvent dans l'espace de noms désigné. CodeArtifact les [espaces de noms](#) sont synonymes de portées et sont utilisés pour organiser le code en groupes logiques et pour éviter les collisions de noms qui peuvent se produire lorsque votre base de code inclut plusieurs bibliothèques.

La période d'autorisation par défaut après l'appel `login` est de 12 heures et `login` doit être appelée pour actualiser régulièrement le jeton. Pour plus d'informations sur le jeton d'autorisation créé avec la `login` commande, consultez [Jetons créés avec la login commande](#).

Configurer Swift sans la commande de connexion

Bien qu'il soit recommandé de [configurer Swift avec la aws codeartifact login commande](#), vous pouvez également configurer le Swift Package Manager sans la commande de connexion en mettant à jour manuellement la configuration du Swift Package Manager.

Dans la procédure suivante, vous allez utiliser le AWS CLI pour effectuer les opérations suivantes :

1. Récupérez un jeton d'authentification CodeArtifact et stockez-le dans votre environnement. La manière dont les informations d'identification sont stockées dépend du système d'exploitation de l'environnement :
 - a. macOS : une entrée est créée dans l'application macOS Keychain.
 - b. Linux et Windows : une entrée est créée dans le `~/.netrc` fichier.
2. Récupérez l'URL du point de terminaison de votre CodeArtifact dépôt.
3. Dans le fichier `~/.swiftpm/configuration/registries.json` de configuration, ajoutez une entrée avec l'URL du point de terminaison de votre référentiel et le type d'authentification.

Pour configurer le Swift sans la commande de connexion

1. Dans une ligne de commande, utilisez la commande suivante pour récupérer un jeton CodeArtifact d'autorisation et le stocker dans une variable d'environnement.
 - `my_domain` Remplacez-le par votre nom de CodeArtifact domaine.
 - Remplacez `111122223333` par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

macOS and Linux

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
for /f %i in ('aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text') do set CODEARTIFACT_AUTH_TOKEN=%i
```

- Fenêtres PowerShell :

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text
```

2. Obtenez le point de terminaison de votre CodeArtifact dépôt en exécutant la commande suivante. Le point de terminaison de votre dépôt est utilisé pour faire pointer le Swift Package Manager vers votre dépôt afin de consommer ou de publier des packages.

- *my_domain* Remplacez-le par votre nom de CodeArtifact domaine.
- Remplacez *111122223333* par le numéro de AWS compte du propriétaire du domaine. Si vous accédez à un référentiel dans un domaine qui vous appartient, vous n'avez pas besoin de l'inclure `--domain-owner`. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).
- *my_repo* Remplacez-le par le nom de votre CodeArtifact dépôt.

macOS and Linux

```
export CODEARTIFACT_REPO=`aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format swift --query repositoryEndpoint --output text`
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
for /f %i in ('aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format swift --query repositoryEndpoint --output text') do set CODEARTIFACT_REPO=%i
```

- Fenêtres PowerShell :

```
$env:CODEARTIFACT_REPO = aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format swift --query repositoryEndpoint --output text
```

L'URL suivante est un exemple de point de terminaison du référentiel.

```
https://my\_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/swift/my\_repo/
```

Note

Pour utiliser un point de terminaison à double pile, utilisez le `codeartifact.region.on.aws` point de terminaison.

Important

Vous devez l'ajouter à la fin du point de terminaison de l'URL du référentiel lorsque vous l'utilisez pour configurer le Swift Package Manager. Cela est fait pour vous dans les commandes de cette procédure.

3. Ces deux valeurs étant stockées dans des variables d'environnement, transmettez-les à Swift à l'aide de la `swift package-registry login` commande suivante :

macOS and Linux

```
swift package-registry login ${CODEARTIFACT_REPO}login --token  
${CODEARTIFACT_AUTH_TOKEN}
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
swift package-registry login %CODEARTIFACT_REPO%login --token  
%CODEARTIFACT_AUTH_TOKEN%
```

- Fenêtres PowerShell :

```
swift package-registry login $Env:CODEARTIFACT_REPO+"login" --token  
$Env:CODEARTIFACT_AUTH_TOKEN
```

4. Ensuite, mettez à jour le registre des packages utilisé par votre application afin que toute dépendance soit extraite de votre CodeArtifact référentiel. Cette commande doit être exécutée dans le répertoire du projet où vous essayez de résoudre la dépendance du package :

macOS and Linux

```
$ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
```

Windows

- Windows (en utilisant l'interface de commande par défaut) :

```
$ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
```

- Fenêtres PowerShell :

```
$ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
```

L'--scope option configure l'application pour qu'elle ne consomme les packages de votre CodeArtifact référentiel que s'ils se situent dans la zone d'application désignée. Les étendues sont synonymes d'[CodeArtifact espaces de noms](#) et sont utilisées pour organiser le code en

groupes logiques et pour éviter les collisions de noms qui peuvent se produire lorsque votre base de code inclut plusieurs bibliothèques.

5. Vous pouvez vérifier que la configuration a été correctement configurée en consultant le contenu du `.swiftpm/configuration/registries.json` fichier au niveau du projet en exécutant la commande suivante dans le répertoire de votre projet :

```
$ cat .swiftpm/configuration/registries.json
{
  "authentication" : {

  },
  "registries" : {
    "[default]" : {
      "url" : "https://my-domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/swift/my-repo/"
    }
  },
  "version" : 1
}
```

Maintenant que vous avez configuré le gestionnaire de packages Swift avec votre CodeArtifact dépôt, vous pouvez l'utiliser pour publier et utiliser des packages Swift depuis et vers celui-ci. Pour de plus amples informations, veuillez consulter [Consommation et publication de packages Swift](#).

Consommation et publication de packages Swift

Consommer des packages Swift à partir de CodeArtifact

Utilisez la procédure suivante pour utiliser des packages Swift à partir d'un AWS CodeArtifact dépôt.

Pour utiliser des packages Swift à partir d'un CodeArtifact dépôt

1. Si ce n'est pas le cas, suivez les étapes ci-dessous [Configurez le gestionnaire de packages Swift avec CodeArtifact](#) pour configurer le Swift Package Manager afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

2. Modifiez le `Package.swift` fichier dans le dossier du projet de votre application pour mettre à jour les dépendances des packages à utiliser par votre projet.
 - a. Si le `Package.swift` fichier ne contient pas de `dependencies` section, ajoutez-en une.
 - b. Dans la `dependencies` section du `Package.swift` fichier, ajoutez le package que vous souhaitez utiliser en ajoutant son identifiant de package. L'identifiant du package se compose de la portée et du nom du package séparés par un point. Consultez l'extrait de code suivant une étape ultérieure pour un exemple.

Tip

Pour trouver l'identifiant du package, vous pouvez utiliser la CodeArtifact console. Recherchez la version du package que vous souhaitez utiliser et consultez les instructions du raccourci d'installation sur la page de version du package.

- c. Si le `Package.swift` fichier ne contient pas de `targets` section, ajoutez-en une.
- d. Dans la `targets` section, ajoutez les cibles qui devront utiliser la dépendance.

L'extrait suivant est un exemple d'extrait illustrant les sections configurées `dependencies` et les `targets` sections d'un fichier : `Package.swift`

```
...  
],  
dependencies: [  
    .package(id: "my_scope.package_name", from: "1.0.0")  
],  
targets: [  
    .target(  
        name: "MyApp",  
        dependencies: ["package_name"]  
    ),...  
],  
...
```

3. Maintenant que tout est configuré, utilisez la commande suivante pour télécharger les dépendances du package depuis CodeArtifact.

```
swift package resolve
```

Consommation de packages Swift depuis CodeArtifact Xcode

Utilisez la procédure suivante pour utiliser des packages Swift à partir d'un CodeArtifact dépôt dans Xcode.

Pour consommer des packages Swift à partir d'un CodeArtifact dépôt dans Xcode

1. Si ce n'est pas le cas, suivez les étapes ci-dessous [Configurez le gestionnaire de packages Swift avec CodeArtifact](#) pour configurer le Swift Package Manager afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

 Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

2. Ajoutez les packages en tant que dépendance dans votre projet dans Xcode.
 - a. Choisissez Fichier > Ajouter des packages.
 - b. Recherchez votre package à l'aide de la barre de recherche. Votre recherche doit figurer dans le formulaire `package_scope.package_name`.
 - c. Une fois trouvé, choisissez le package et choisissez Ajouter un package.
 - d. Une fois le package vérifié, choisissez les produits du package que vous souhaitez ajouter en tant que dépendance, puis choisissez Ajouter un package.

Si vous rencontrez des problèmes lors de l'utilisation de votre CodeArtifact dépôt avec Xcode, consultez [Résolution rapide des problèmes](#) les problèmes courants et les solutions possibles.

Publication de packages Swift sur CodeArtifact

CodeArtifact recommande Swift 5.9 ou version ultérieure et utilise la `swift package-registry publish` commande pour publier des packages Swift. Si vous utilisez une version antérieure, vous devez utiliser une commande Curl pour publier les packages Swift dans. CodeArtifact

Publier CodeArtifact des packages à l'aide de la `swift package-registry publish` commande

Utilisez la procédure suivante avec Swift 5.9 ou version ultérieure pour publier des packages Swift dans un CodeArtifact référentiel à l'aide du Swift Package Manager.

1. Si ce n'est pas le cas, suivez les étapes ci-dessous [Configurez le gestionnaire de packages Swift avec CodeArtifact](#) pour configurer le Swift Package Manager afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

 Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis sa création.

2. Accédez au répertoire du projet Swift qui contient le `Package.swift` fichier de votre package.
3. Exécutez la `swift package-registry publish` commande suivante pour publier le package. La commande crée une archive source de package et la publie dans votre CodeArtifact dépôt.

```
swift package-registry publish packageScope.packageName packageVersion
```

Par exemple :

```
swift package-registry publish myScope.myPackage 1.0.0
```

4. Vous pouvez vérifier que le package a été publié et qu'il existe dans le référentiel en vérifiant dans la console ou en utilisant la `aws codeartifact list-packages` commande suivante :

```
aws codeartifact list-packages --domain my_domain --repository my_repo
```

Vous pouvez répertorier la version unique du package à l'aide de la aws codeartifact list-package-versions commande suivante :

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \  
--format swift --namespace my_scope --package package_name
```

Publication de CodeArtifact packages avec Curl

Bien qu'il soit recommandé d'utiliser la swift package-registry publish commande fournie avec Swift 5.9 ou version ultérieure, vous pouvez également utiliser Curl pour publier des packages Swift dans CodeArtifact

Utilisez la procédure suivante pour publier des packages Swift dans un AWS CodeArtifact dépôt avec Curl.

1. Si ce n'est pas le cas, créez et mettez à jour les variables d'CODEARTIFACT_REPOenvironnement CODEARTIFACT_AUTH_TOKEN et en suivant les étapes décrites dans [Configurez le gestionnaire de packages Swift avec CodeArtifact](#).

Note

Le jeton d'autorisation est valide pendant 12 heures. Vous devrez actualiser votre variable d'CODEARTIFACT_AUTH_TOKENenvironnement avec de nouvelles informations d'identification si 12 heures se sont écoulées depuis sa création.

2. Tout d'abord, si aucun package Swift n'a été créé, vous pouvez le faire en exécutant les commandes suivantes :

```
mkdir testDir && cd testDir  
swift package init  
git init .  
swift package archive-source
```

3. Utilisez la commande Curl suivante pour publier votre package Swift sur : CodeArtifact
macOS and Linux

```
curl -X PUT --user "aws:$CODEARTIFACT_AUTH_TOKEN" \  
-H "Accept: application/vnd.swift.registry.v1+json" \  
-F "archive=@archivePath"
```

```
-F source-archive="@test_dir_package_name.zip" \
"${CODEARTIFACT_REPO}my_scope/package_name/packageVersion"
```

Windows

```
curl -X PUT --user "aws:%CODEARTIFACT_AUTH_TOKEN%" \
-H "Accept: application/vnd.swift.registry.v1+json" \
-F source-archive="@test_dir_package_name.zip" \
"%CODEARTIFACT_REPO%my_scope/package_name/packageVersion"
```

4. Vous pouvez vérifier que le package a été publié et qu'il existe dans le référentiel en vérifiant dans la console ou en utilisant la `aws codeartifact list-packages` commande suivante :

```
aws codeartifact list-packages --domain my_domain --repository my_repo
```

Vous pouvez répertorier la version unique du package à l'aide de la `aws codeartifact list-package-versions` commande suivante :

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Récupération de packages Swift depuis GitHub et republication vers CodeArtifact

Utilisez la procédure suivante pour récupérer un package Swift GitHub et le republier dans un CodeArtifact dépôt.

Pour récupérer un package Swift GitHub et le republier dans CodeArtifact

1. Si ce n'est pas le cas, suivez les étapes ci-dessous [Configurez le gestionnaire de packages Swift avec CodeArtifact](#) pour configurer le Swift Package Manager afin d'utiliser votre CodeArtifact référentiel avec les informations d'identification appropriées.

Note

Le jeton d'autorisation généré est valide pendant 12 heures. Vous devrez en créer un nouveau si 12 heures se sont écoulées depuis la création du jeton.

- Clonez le dépôt git du package Swift que vous souhaitez récupérer et republier à l'aide de la commande suivante `git clone`. Pour plus d'informations sur le clonage de GitHub référentiels, consultez la section [Clonage d'un référentiel dans la documentation](#). GitHub

```
git clone repoURL
```

- Accédez au référentiel que vous venez de cloner :

```
cd repoName
```

- Créez le package et publiez-le sur CodeArtifact.

- Recommandé : Si vous utilisez Swift 5.9 ou une version ultérieure, vous pouvez utiliser la `swift package-registry publish` commande suivante pour créer le package et le publier dans votre CodeArtifact référentiel configuré.

```
swift package-registry publish packageScope.packageName versionNumber
```

Par exemple :

```
swift package-registry publish myScope.myPackage 1.0.0
```

- Si vous utilisez une version de Swift antérieure à la version 5.9, vous devez utiliser la `swift archive-source` commande pour créer le package, puis utiliser une commande Curl pour le publier.

- Si vous n'avez pas configuré les variables d'CODEARTIFACT_REPOenvironnement CODEARTIFACT_AUTH_TOKEN et, ou si cela fait plus de 12 heures que vous ne l'avez pas fait, suivez les étapes décrites dans [Configurer Swift sans la commande de connexion](#).
- Créez le package Swift à l'aide de la `swift package archive-source` commande :

```
swift package archive-source
```

En cas de succès, vous le verrez Created *package_name.zip* dans la ligne de commande.

- Utilisez la commande Curl suivante pour publier le package Swift sur : CodeArtifact

macOS and Linux

```
curl -X PUT --user "aws:$CODEARTIFACT_AUTH_TOKEN" \
-H "Accept: application/vnd.swift.registry.v1+json" \
-F source-archive="@package_name.zip" \
"${CODEARTIFACT_REPO}my_scope/package_name/packageVersion"
```

Windows

```
curl -X PUT --user "aws:%CODEARTIFACT_AUTH_TOKEN%" \
-H "Accept: application/vnd.swift.registry.v1+json" \
-F source-archive="@package_name.zip" \
"%CODEARTIFACT_REPO%my_scope/package_name/packageVersion"
```

5. Vous pouvez vérifier que le package a été publié et qu'il existe dans le référentiel en vérifiant dans la console ou en utilisant la `aws codeartifact list-packages` commande suivante :

```
aws codeartifact list-packages --domain my_domain --repository my_repo
```

Vous pouvez répertorier la version unique du package à l'aide de la `aws codeartifact list-package-versions` commande suivante :

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Normalisation rapide du nom du package et de l'espace de noms

CodeArtifact normalise les noms de packages et les espaces de noms avant de les stocker, ce qui signifie que les noms CodeArtifact peuvent être différents de ceux fournis lors de la publication du package.

Normalisation du nom du package et de l'espace de noms : CodeArtifact normalise les noms de packages et les espaces de noms Swift en convertissant toutes les lettres en minuscules.

Normalisation des versions des packages : CodeArtifact ne normalise pas les versions des packages Swift. [Notez que CodeArtifact seuls les modèles de version 2.0 sont pris en charge. Pour plus d'informations sur le versionnement sémantique, consultez Semantic Versioning 2.0.0.](#)

Le nom du package et l'espace de noms non normalisés peuvent être utilisés avec les demandes d'API et de CLI car ils CodeArtifact normalisent les entrées de ces demandes. Par exemple, les entrées de `--package myPackage` et `--namespace myScope` seraient normalisées et renverraient un package dont le nom de package `mypackage` et l'espace de noms normalisés sont de `myscope`.

Vous devez utiliser des noms normalisés dans les ARNs politiques IAM, par exemple.

Pour trouver le nom normalisé d'un package, utilisez la `aws codeartifact list-packages` commande. Pour de plus amples informations, veuillez consulter [Lister les noms de packages](#).

Résolution rapide des problèmes

Les informations suivantes peuvent vous aider à résoudre les problèmes courants liés à Swift et CodeArtifact.

Je reçois une erreur 401 dans Xcode même après avoir configuré le Swift Package Manager

Problème : Lorsque vous essayez d'ajouter un package depuis votre CodeArtifact dépôt en tant que dépendance à votre projet Swift dans Xcode, vous obtenez une erreur 401 non autorisée même après avoir suivi les instructions de [connexion à CodeArtifact Swift](#).

Correctifs possibles : Cela peut être dû à un problème lié à l'application macOS Keychain, dans laquelle vos CodeArtifact informations d'identification sont stockées. Pour résoudre ce problème, nous vous recommandons d'ouvrir l'application Keychain, de supprimer toutes les CodeArtifact entrées et de configurer à nouveau le Swift Package Manager avec votre CodeArtifact référentiel en suivant les instructions fournies dans [Configurez le gestionnaire de packages Swift avec CodeArtifact](#).

Xcode se bloque sur la machine CI en raison de l'invite de saisie du mot de passe par le trousseau

Problème : Lorsque vous essayez d'extraire des packages Swift dans le CodeArtifact cadre d'une version Xcode sur un serveur d'intégration continue (CI), par exemple avec GitHub Actions, l'authentification CodeArtifact peut se bloquer et éventuellement échouer avec un message d'erreur similaire au suivant :

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\' to keychain: status -60008
```

Correctifs possibles : Cela est dû au fait que les informations d'identification ne sont pas enregistrées dans le trousseau sur les machines CI et que Xcode ne prend en charge que les informations d'identification enregistrées dans le trousseau. Pour résoudre ce problème, nous vous recommandons de créer l'entrée du trousseau manuellement en suivant les étapes suivantes :

1. Préparez le trousseau.

```
KEYCHAIN_PASSWORD=$(openssl rand -base64 20)
KEYCHAIN_NAME=login.keychain
SYSTEM_KEYCHAIN=/Library/Keychains/System.keychain

if [ -f $HOME/Library/Keychains/"${KEYCHAIN_NAME}"-db ]; then
    echo "Deleting old ${KEYCHAIN_NAME} keychain"
    security delete-keychain "${KEYCHAIN_NAME}"
fi
echo "Create Keychain"
security create-keychain -p "${KEYCHAIN_PASSWORD}" "${KEYCHAIN_NAME}"

EXISTING_KEYCHAINS=( $( security list-keychains | sed -e 's/ *// ' | tr '\n' ' ' | tr -d '"' ) )
sudo security list-keychains -s "${KEYCHAIN_NAME}" "${EXISTING_KEYCHAINS[@]}"

echo "New keychain search list :"
security list-keychain

echo "Configure keychain : remove lock timeout"
security unlock-keychain -p "${KEYCHAIN_PASSWORD}" "${KEYCHAIN_NAME}"
security set-keychain-settings "${KEYCHAIN_NAME}"
```

2. Obtenez un jeton CodeArtifact d'authentification et le point de terminaison de votre référentiel.

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token \
    --region us-west-2 \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text`
```

```
export CODEARTIFACT_REPO=`aws codeartifact get-repository-endpoint \
    --region us-west-2 \
    --domain my_domain \
    --domain-owner 111122223333 \
    --format swift \
    --repository my_repo \
    --query repositoryEndpoint \
    --output text`
```

3. Créez manuellement l'entrée Keychain.

```
SERVER=$(echo $CODEARTIFACT_REPO | sed 's/https:\//g' | sed 's/.com.*$/.com/g')
AUTHORIZATION=(-T /usr/bin/security -T /usr/bin/codesign -T /usr/bin/xcodebuild -
T /usr/bin/swift \
    -T /Applications/Xcode-15.2.app/Contents/Developer/usr/bin/
xcodebuild)

security delete-internet-password -a token -s $SERVER -r https "${KEYCHAIN_NAME}"

security add-internet-password -a token \
    -s $SERVER \
    -w $CODEARTIFACT_AUTH_TOKEN \
    -r https \
    -U \
    "${AUTHORIZATION[@]}" \
    "${KEYCHAIN_NAME}"

security set-internet-password-partition-list \
    -a token \
    -s $SERVER \
    -S "com.apple.swift"
package,com.apple.security,com.apple.dt.Xcode,apple-tool:,apple:,codesign" \
    -k "${KEYCHAIN_PASSWORD}" "${KEYCHAIN_NAME}"

security find-internet-password "${KEYCHAIN_NAME}"
```

Pour plus d'informations sur cette erreur et la solution, consultez <https://github.com/apple/swift-package-manager/issues/7236>.

Utilisation CodeArtifact avec des packages génériques

Ces rubriques vous montrent comment utiliser et publier des packages génériques à l'aide de AWS CodeArtifact.

Rubriques

- [Vue d'ensemble des packages génériques](#)
- [Commandes prises en charge pour les packages génériques](#)
- [Publication et consommation de packages génériques](#)

Vue d'ensemble des packages génériques

Grâce au format de generic package, vous pouvez télécharger n'importe quel type de fichier pour créer un package dans un CodeArtifact référentiel. Les packages génériques ne sont associés à aucun langage de programmation, type de fichier ou écosystème de gestion de packages spécifique. Cela peut être utile pour stocker et versionner des artefacts de build arbitraires, tels que des installateurs d'applications, des modèles d'apprentissage automatique, des fichiers de configuration, etc.

Un package générique comprend un nom de package, un espace de noms, une version et un ou plusieurs actifs (ou fichiers). Les packages génériques peuvent coexister avec des packages d'autres formats dans un CodeArtifact référentiel unique.

Vous pouvez utiliser le SDK AWS CLI ou le SDK pour travailler avec des packages génériques. Pour obtenir la liste complète des AWS CLI commandes qui fonctionnent avec les packages génériques, consultez[Commandes prises en charge pour les packages génériques](#).

Contraintes de package génériques

- Ils ne sont jamais extraits des référentiels en amont. Ils ne peuvent être obtenus qu'à partir du référentiel dans lequel ils ont été publiés.
- Ils ne peuvent pas déclarer les dépendances à renvoyer [ListPackageVersionDependencies](#) ou à afficher dans le AWS Management Console .
- Ils peuvent stocker des fichiers README et LICENSE, mais ils ne sont pas interprétés par CodeArtifact. Les informations contenues dans ces fichiers ne sont pas renvoyées par

[GetPackageVersionReadme](#) ou [DescribePackageVersion](#), et n'apparaissent pas dans le AWS Management Console.

- Comme pour tous les packages CodeArtifact inclus, la taille des actifs et le nombre d'actifs par package sont limités. Pour plus d'informations sur les limites et les quotas dans CodeArtifact, voir [Quotas dans AWS CodeArtifact](#).
- Les noms des actifs qu'ils contiennent doivent respecter les règles suivantes :
 - Les noms d'actifs peuvent utiliser des lettres et des chiffres Unicode. Plus précisément, les catégories de caractères Unicode suivantes sont autorisées : lettre minuscule (Ll), lettre modificatrice (Lm), autre lettre (Lo), lettre majuscule (), lettre majuscule (Lt), numéro de lettre (Lu) et nombre Nl décimal (). Nd
 - Les caractères spéciaux suivants sont acceptés : ~!@^&(`)-_+[{}];,.
 - Les actifs ne peuvent pas être nommés . ou ..
 - Les espaces sont les seuls espaces autorisés. Les noms de ressources ne peuvent pas commencer ou se terminer par un espace, ni inclure des espaces consécutifs.

Commandes prises en charge pour les packages génériques

Vous pouvez utiliser le SDK AWS CLI ou le SDK pour travailler avec des packages génériques. Les CodeArtifact commandes suivantes fonctionnent avec les packages génériques :

- [copy-package-versions](#)(voir [Copier des packages entre des référentiels](#))
- [delete-package](#) (voir [Suppression d'un package \(AWS CLI\)](#))
- [delete-package-versions](#)(voir [Supprimer une version de package \(AWS CLI\)](#))
- [décrivez le package](#)
- [describe-package-version](#)(voir [Afficher et mettre à jour les détails et les dépendances des versions du package](#))
- [dispose-package-versions](#)(voir [Élimination des versions du package](#))
- [get-package-version-asset](#)(voir [Télécharger les ressources de la version du package](#))
- [list-package-version-assets](#)(voir [Répertorier les actifs de la version](#))
- [list-package-versions](#)(voir [Lister les versions des packages](#))
- [list-packages](#) (voir [Lister les noms de packages](#))
- [publish-package-version](#)(voir [Publication d'un package générique](#))
- [put-package-origin-configuration](#)(voir [Modification des contrôles d'origine des packages](#))

Note

Vous pouvez utiliser le paramètre de contrôle publish d'origine pour autoriser ou bloquer la publication d'un nom de package générique dans un référentiel. Toutefois, le upstream paramètre ne s'applique pas aux packages génériques car ils ne peuvent pas être extraits d'un référentiel en amont.

- [update-package-versions-status](#)(voir[Mettre à jour le statut de version du package](#))

Publication et consommation de packages génériques

Pour publier une version de package générique et ses ressources associées, utilisez la publish-package-version commande. Vous pouvez répertorier les actifs d'un package générique à l'aide de la list-package-version-asset commande et les télécharger à l'aide de get-package-version-asset. La rubrique suivante contient des step-by-step instructions pour publier des packages génériques ou télécharger des actifs de packages génériques à l'aide de ces commandes.

Publication d'un package générique

Un package générique comprend un nom de package, un espace de noms, une version et un ou plusieurs actifs (ou fichiers). Cette rubrique explique comment publier un package nommé `my-package`, avec l'espace de noms `my-ns`, la version `1.0.0` et contenant une ressource nommée `asset.tar.gz`.

Prérequis :

- Configurez et configurez le AWS Command Line Interface avec CodeArtifact (voir[Configuration avec AWS CodeArtifact](#))
- Disposer d'un CodeArtifact domaine et d'un référentiel (voir[Commencer à utiliser le AWS CLI](#))

Pour publier un package générique

1. Utilisez la commande suivante pour générer le SHA256 hachage de chaque fichier que vous souhaitez télécharger dans une version de package et placez la valeur dans une variable d'environnement. Cette valeur est utilisée comme contrôle d'intégrité pour vérifier que le contenu du fichier n'a pas changé après son envoi initial.

Linux

```
export ASSET_SHA256=$(sha256sum asset.tar.gz | awk '{print $1;}')
```

macOS

```
export ASSET_SHA256=$(shasum -a 256 asset.tar.gz | awk '{print $1;}')
```

Windows

```
for /f "tokens=*" %G IN ('certUtil -hashfile asset.tar.gz SHA256 ^| findstr /v "hash"') DO SET "ASSET_SHA256=%G"
```

2. Appelez `publish-package-version` pour télécharger la ressource et créer une nouvelle version du package.

Note

Si votre package contient plusieurs ressources, vous pouvez appeler `publish-package-version` une fois pour chaque ressource à télécharger. Incluez l'--unfinished argument pour chaque appel à `publish-package-version`, sauf lors du téléchargement de la ressource finale. L'omission --unfinished définira le statut de la version du package sur Published et empêchera le téléchargement de ressources supplémentaires vers celui-ci.

Vous pouvez également inclure --unfinished pour chaque appel à `publish-package-version`, puis définir le statut de la version du package à Published l'aide de la `update-package-versions-status` commande.

Linux/macOS

```
aws codeartifact publish-package-version --domain my_domain --repository my_repo \
  --format generic --namespace my-ns --package my-package --package-version 1.0.0 \
  --asset-content asset.tar.gz --asset-name asset.tar.gz \
  --asset-sha256 $ASSET_SHA256
```

Windows

```
aws codeartifact publish-package-version --domain my_domain --repository my_repo
^
  --format generic --namespace my-ns --package my-package --package-
version 1.0.0 ^
  --asset-content asset.tar.gz --asset-name asset.tar.gz ^
  --asset-sha256 %ASSET_SHA256%
```

Le résultat est présenté ci-dessous :

{

```
  "format": "generic",
  "namespace": "my-ns",
  "package": "my-package",
  "version": "1.0.0",
  "versionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
  "status": "Published",
  "asset": {
    "name": "asset.tar.gz",
    "size": 11,
    "hashes": {
      "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",
      "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",
      "SHA-256": "43f24850b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
      "SHA-512":
        "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fc
        ed7bd1e80a0dca9ce320d95
      SHA-512"
    }
  }
}
```

Répertorier les actifs des packages génériques

Pour répertorier les actifs contenus dans un package générique, utilisez la `list-package-version-assets` commande. Pour de plus amples informations, veuillez consulter [Répertorier les actifs de la version](#).

L'exemple suivant répertorie les actifs de la version 1.0.0 du packagemy-package.

Pour répertorier les actifs des versions du package

- Appelez `list-package-version-assets` pour répertorier les actifs contenus dans un package générique.

Linux/macOS

```
aws codeartifact list-package-version-assets --domain my_domain \  
  --repository my_repo --format generic --namespace my-ns \  
  --package my-package --package-version 1.0.0
```

Windows

```
aws codeartifact list-package-version-assets --domain my_domain ^  
  --repository my_repo --format generic --namespace my-ns ^  
  --package my-package --package-version 1.0.0
```

Le résultat est présenté ci-dessous :

```
{  
  "assets": [  
    {  
      "name": "asset.tar.gz",  
      "size": 11,  
      "hashes": {  
        "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",  
        "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",  
        "SHA-256":  
          "43f24850b7b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-SHA-256",  
        "SHA-512":  
          "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fc  
          ed7bd1e80a0dca9ce320d95  
        "SHA-512"  
      }  
    }  
  ],  
  "package": "my-package",  
  "format": "generic",  
  "namespace": "my-ns",  
  "version": "1.0.0",
```

```
        "versionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"  
    }
```

Téléchargement des actifs de packages génériques

Pour télécharger les ressources à partir d'un package générique, utilisez la `get-package-version-asset` commande. Pour de plus amples informations, veuillez consulter [Télécharger les ressources de la version du package](#).

L'exemple suivant télécharge la ressource `asset.tar.gz` depuis la version `1.0.0` du package `my-package` vers le répertoire de travail actuel dans un fichier également nommé `asset.tar.gz`.

Pour télécharger les ressources relatives à la version du package

- Appelez `get-package-version-asset` pour télécharger des ressources à partir d'un package générique.

Linux/macOS

```
aws codeartifact get-package-version-asset --domain my_domain \  
  --repository my_repo --format generic --namespace my-ns --package my-package \  
  --package-version 1.0.0 --asset asset.tar.gz \  
  asset.tar.gz
```

Windows

```
aws codeartifact get-package-version-asset --domain my_domain ^  
  --repository my_repo --format generic --namespace my-ns --package my-package ^  
  --package-version 1.0.0 --asset asset.tar.gz ^  
  asset.tar.gz
```

Le résultat est présenté ci-dessous :

```
{  
    "assetName": "asset.tar.gz",  
    "packageVersion": "1.0.0",  
    "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"  
}
```

Utilisation CodeArtifact avec CodeBuild

Ces rubriques décrivent comment utiliser les packages d'un CodeArtifact référentiel dans le cadre d'un projet de AWS CodeBuild construction.

Rubriques

- [Utilisation des packages npm dans CodeBuild](#)
- [Utilisation de packages Python dans CodeBuild](#)
- [Utilisation des packages Maven dans CodeBuild](#)
- [Utilisation de NuGet packages dans CodeBuild](#)
- [Mise en cache des dépendances](#)

Utilisation des packages npm dans CodeBuild

Les étapes suivantes ont été testées avec les systèmes d'exploitation répertoriés dans les [images Docker fournies par CodeBuild](#).

Configurer des autorisations avec des rôles IAM

Ces étapes sont requises lors de l'utilisation de packages npm CodeArtifact à CodeBuild partir de.

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, choisissez Roles (Rôles). Sur la page Rôles, modifiez le rôle utilisé par votre projet de CodeBuild construction. Ce rôle doit disposer des autorisations suivantes.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [ "codeartifact:GetAuthorizationToken",  
                 "codeartifact:ListAvailableArtifacts" ]  
    }  
  ]  
}
```

```
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "sts:GetServiceBearerToken",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "sts:AWSServiceName": "codeartifact.amazonaws.com"
        }
    }
}
]
```

 **Important**

Si vous souhaitez également utiliser pour CodeBuild publier des packages, ajoutez l'**codeartifact:PublishPackageVersion** autorisation.

Pour plus d'informations, consultez [la section Modification d'un rôle](#) dans le guide de l'utilisateur IAM.

Connectez-vous et utilisez npm

Pour utiliser les packages npm à partir de CodeBuild, exécutez la `login` commande depuis la `pre-build` section de votre projet `buildspec.yaml` à configurer npm pour récupérer les packages. CodeArtifact Pour plus d'informations, consultez [Authentification avec npm](#).

Une `login` fois l'exécution réussie, vous pouvez exécuter npm les commandes de la `build` section pour installer les packages npm.

Linux

Note

Il n'est nécessaire de mettre à jour le AWS CLI with que `pip3 install awscli --upgrade --user` si vous utilisez une CodeBuild image plus ancienne. Si vous utilisez les dernières versions d'image, vous pouvez supprimer cette ligne.

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - aws codeartifact login --tool npm --domain my_domain --domain-owner 111122223333
      --repository my_repo
build:
  commands:
    - npm install
```

Windows

```
version: 0.2
phases:
  install:
    commands:
      - '[Net.ServicePointManager]::SecurityProtocol = "Tls12"; Invoke-WebRequest
https://awscli.amazonaws.com/AWSCLIV2.msi -OutFile $env:TEMP/AWSCLIV2.msi'
      - Start-Process -Wait msieexec "/i $env:TEMP\AWSCLIV2.msi /quiet /norestart"
  pre_build:
    commands:
      - '&"C:\Program Files\Amazon\AWSCLIV2\aws" codeartifact login --tool npm --
domain my_domain --domain-owner 111122223333 --repository my_repo'
  build:
    commands:
      - npm install
```

Utilisation de packages Python dans CodeBuild

Les étapes suivantes ont été testées avec les systèmes d'exploitation répertoriés dans les [images Docker fournies par CodeBuild](#).

Configurer des autorisations avec des rôles IAM

Ces étapes sont obligatoires lors de l'utilisation de packages Python depuis CodeArtifact l'entrée CodeBuild.

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, choisissez Roles (Rôles). Sur la page Rôles, modifiez le rôle utilisé par votre projet de CodeBuild construction. Ce rôle doit disposer des autorisations suivantes.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [ "codeartifact:GetAuthorizationToken",  
                 "codeartifact:GetRepositoryEndpoint",  
                 "codeartifact:ReadFromRepository"  
               ],  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

⚠️ Important

Si vous souhaitez également utiliser pour CodeBuild publier des packages, ajoutez l'**codeartifact:PublishPackageVersion** autorisation.

Pour plus d'informations, consultez [la section Modification d'un rôle](#) dans le guide de l'utilisateur IAM.

Connectez-vous et utilisez un pip ou une ficelle

Pour utiliser les packages Python depuis CodeBuild, exécutez la `login` commande depuis la `pre-build` section du `buildspec.yaml` fichier de votre projet à configurer pip pour récupérer les packages. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec Python](#).

Une fois l'exécution réussie, vous pouvez exécuter pip les commandes de la `build` section pour installer ou publier des packages Python.

Linux

ⓘ Note

Il n'est nécessaire de mettre à jour le AWS CLI avec `pip3 install awscli --upgrade --user` si vous utilisez une CodeBuild image plus ancienne. Si vous utilisez les dernières versions d'image, vous pouvez supprimer cette ligne.

Pour installer des packages Python à l'aide de pip :

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - aws codeartifact login --tool pip --domain my_domain --domain-owner 111122223333
      --repository my_repo
build:
  commands:
```

```
- pip install requests
```

Pour publier des packages Python à l'aide de twine :

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - aws codeartifact login --tool twine --domain my_domain --domain-owner 111122223333 --repository my_repo
build:
  commands:
    - twine upload --repository codeartifact mypackage
```

Windows

Pour installer des packages Python à l'aide de pip :

```
version: 0.2
phases:
  install:
    commands:
      - '[Net.ServicePointManager]::SecurityProtocol = "Tls12"; Invoke-WebRequest https://awscli.amazonaws.com/AWSCLIV2.msi -OutFile $env:TEMP/AWSCLIV2.msi'
      - Start-Process -Wait msieexec "/i $env:TEMP\AWSCLIV2.msi /quiet /norestart"
  pre_build:
    commands:
      - '&"C:\Program Files\Amazon\AWSCLIV2\aws" codeartifact login --tool pip --domain my_domain --domain-owner 111122223333 --repository my_repo'
  build:
    commands:
      - pip install requests
```

Pour publier des packages Python à l'aide de twine :

```
version: 0.2
phases:
  install:
    commands:
      - '[Net.ServicePointManager]::SecurityProtocol = "Tls12"; Invoke-WebRequest https://awscli.amazonaws.com/AWSCLIV2.msi -OutFile $env:TEMP/AWSCLIV2.msi'
      - Start-Process -Wait msieexec "/i $env:TEMP\AWSCLIV2.msi /quiet /norestart"
```

```
pre_build:
  commands:
    - '&"C:\Program Files\Amazon\AWSCLIV2\aws" codeartifact login --tool twine --
domain my_domain --domain-owner 111122223333 --repository my_repo'
build:
  commands:
    - twine upload --repository codeartifact mypackage
```

Utilisation des packages Maven dans CodeBuild

Configurer des autorisations avec des rôles IAM

Ces étapes sont obligatoires lorsque vous utilisez des packages Maven depuis CodeArtifact et vers CodeBuild

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, choisissez Roles (Rôles). Sur la page Rôles, modifiez le rôle utilisé par votre projet de CodeBuild construction. Ce rôle doit disposer des autorisations suivantes.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sts:GetServiceBearerToken",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:codeartifact:<region>::repository/<repository>"
        }
      }
    }
  ]
}
```

```
        "sts:AWSServiceName": "codeartifact.amazonaws.com"
    }
}
]
}
```

Important

Si vous souhaitez également utiliser CodeBuild pour publier des packages, ajoutez les **codeartifact:PutPackageMetadata** autorisations **codeartifact:PublishPackageVersion** et.

Pour plus d'informations, consultez [la section Modification d'un rôle](#) dans le guide de l'utilisateur IAM.

Utilisez Gradle ou MVN

Pour utiliser des packages Maven avec gradle ou mvn, stockez le jeton d' CodeArtifact authentification dans une variable d'environnement, comme décrit dans [Transmettre un jeton d'authentification dans une variable d'environnement](#). Voici un exemple.

Note

Il n'est nécessaire de mettre à jour le AWS CLI with que `pip3 install awscli --upgrade --user` si vous utilisez une CodeBuild image plus ancienne. Si vous utilisez les dernières versions d'image, vous pouvez supprimer cette ligne.

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
      domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

Pour utiliser Gradle :

Si vous avez référencé la CODEARTIFACT_AUTH_TOKEN variable dans votre build.gradle fichier Gradle comme décrit dans [Utilisation CodeArtifact avec Gradle](#), vous pouvez invoquer votre build Gradle depuis la section buildspec.yaml build

```
build:  
  commands:  
    - gradle build
```

Pour utiliser MVN :

Vous devez configurer vos fichiers de configuration Maven (settings.xml et pom.xml) suivre les instructions de la section [Utilisation CodeArtifact avec mvn](#).

Utilisation de NuGet packages dans CodeBuild

Les étapes suivantes ont été testées avec les systèmes d'exploitation répertoriés dans les [images Docker fournies par CodeBuild](#).

Rubriques

- [Configurer des autorisations avec des rôles IAM](#)
- [Consommez NuGet des colis](#)
- [Construire avec des NuGet packages](#)
- [Publier des NuGet packages](#)

Configurer des autorisations avec des rôles IAM

Ces étapes sont obligatoires lors de l'utilisation de NuGet packages depuis et CodeArtifact vers CodeBuild.

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, choisissez Roles (Rôles). Sur la page Rôles, modifiez le rôle utilisé par votre projet de CodeBuild construction. Ce rôle doit disposer des autorisations suivantes.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [ "codeartifact:GetAuthorizationToken",  
                 "codeartifact:GetRepositoryEndpoint",  
                 "codeartifact:ReadFromRepository"  
               ],  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

 Important

Si vous souhaitez également utiliser pour CodeBuild publier des packages, ajoutez l'**codeartifact:PublishPackageVersion** autorisation.

Pour plus d'informations, consultez [la section Modification d'un rôle](#) dans le guide de l'utilisateur IAM.

Consommez NuGet des colis

Pour utiliser des NuGet packages à partir de CodeBuild, incluez les éléments suivants dans le `buildspec.yaml` fichier de votre projet.

1. Dans `install` cette section, installez le fournisseur CodeArtifact d'informations d'identification pour configurer des outils de ligne de commande tels que, ainsi que `msbuild` `dotnet` pour créer et publier des packages. CodeArtifact
2. Dans la `pre-build` section, ajoutez votre CodeArtifact référentiel à votre NuGet configuration.

Consultez les `buildspec.yaml` exemples suivants. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec NuGet](#).

Une fois le fournisseur d'informations d'identification installé et la source de votre référentiel ajoutée, vous pouvez exécuter les commandes de l'outil NuGet CLI depuis la `build` section pour utiliser les NuGet packages.

Linux

Pour consommer des NuGet packages à l'aide de `dotnet` :

```
version: 0.2

phases:
  install:
    runtime-versions:
      dotnet: latest
    commands:
      - export PATH="$PATH:/root/.dotnet/tools"
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact $(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)"v3/index.json"
  build:
    commands:
      - dotnet add package <packageName> --source codeartifact
```

Windows

Pour consommer des NuGet packages à l'aide de dotnet :

```
version: 0.2

phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - dotnet add package <packageName> --source codeartifact
```

Construire avec des NuGet packages

Pour créer avec des NuGet packages provenant de CodeBuild, incluez les éléments suivants dans le `buildspec.yaml` fichier de votre projet.

1. Dans `install` cette section, installez le fournisseur CodeArtifact d'informations d'identification pour configurer des outils de ligne de commande tels que, ainsi que msbuild dotnet pour créer et publier des packages. CodeArtifact
2. Dans la `pre-build` section, ajoutez votre CodeArtifact référentiel à votre NuGet configuration.

Consultez les `buildspec.yaml` exemples suivants. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec NuGet](#).

Une fois le fournisseur d'informations d'identification installé et la source de votre référentiel ajoutée, vous pouvez exécuter les commandes de l'outil NuGet CLI comme indiqué `dotnet build` dans la `build` section.

Linux

Pour créer des NuGet packages à l'aide de dotnet :

```
version: 0.2

phases:
  install:
    runtime-versions:
      dotnet: latest
    commands:
      - export PATH="$PATH:/root/.dotnet/tools"
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact $(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)"v3/index.json"
  build:
    commands:
      - dotnet build
```

Pour créer des NuGet packages à l'aide de msbuild :

```
version: 0.2

phases:
  install:
    runtime-versions:
      dotnet: latest
    commands:
      - export PATH="$PATH:/root/.dotnet/tools"
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact $(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)"v3/index.json"
  build:
    commands:
      - msbuild -t:Rebuild -p:Configuration=Release
```

Windows

Pour créer des NuGet packages à l'aide de dotnet :

```
version: 0.2

phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - dotnet build
```

Pour créer des NuGet packages à l'aide de msbuild :

```
version: 0.2

phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - msbuild -t:Rebuild -p:Configuration=Release
```

Publier des NuGet packages

Pour publier des NuGet packages CodeBuild, incluez les éléments suivants dans le `buildspec.yaml` fichier de votre projet.

1. Dans `install` cette section, installez le fournisseur CodeArtifact d'informations d'identification pour configurer des outils de ligne de commande tels que, ainsi que `msbuild` `dotnet` pour créer et publier des packages. CodeArtifact
2. Dans la `pre-build` section, ajoutez votre CodeArtifact référentiel à votre NuGet configuration.

Consultez les `buildspec.yaml` exemples suivants. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec NuGet](#).

Une fois le fournisseur d'informations d'identification installé et la source de votre référentiel ajoutée, vous pouvez exécuter les commandes de l'outil NuGet CLI depuis la `build` section et publier vos NuGet packages.

Linux

Pour publier des NuGet packages à l'aide de `dotnet` :

```
version: 0.2

phases:
  install:
    runtime-versions:
      dotnet: latest
    commands:
      - export PATH="$PATH:/root/.dotnet/tools"
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact $(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)"v3/index.json"
  build:
    commands:
      - dotnet pack -o .
      - dotnet nuget push *.nupkg -s codeartifact
```

Windows

Pour publier des NuGet packages à l'aide de `dotnet` :

```
version: 0.2
```

```
phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - dotnet pack -o .
      - dotnet nuget push *.nupkg -s codeartifact
```

Mise en cache des dépendances

Vous pouvez activer la mise en cache locale CodeBuild afin de réduire le nombre de dépendances à extraire CodeArtifact pour chaque build. Pour plus d'informations, voir [Build Caching in AWS CodeBuild dans](#) le guide de l'AWS CodeBuild utilisateur. Après avoir activé un cache local personnalisé, ajoutez le répertoire de cache au `buildspec.yaml` fichier de votre projet.

Par exemple, si vous utilisez mvn, utilisez ce qui suit.

```
cache:
  paths:
    - '/root/.m2/**/*'
```

Pour les autres outils, utilisez les dossiers de cache présentés dans ce tableau.

Outil	Répertoire du cache
mvn	<code>/root/.m2/**/*</code>
gradle	<code>/root/.gradle/caches/**/*</code>
pip	<code>/root/.cache/pip/**/*</code>
npm	<code>/root/.npm/**/*</code>

Outil	Répertoire du cache
nuget	<code>/root/.nuget/**/*</code>
yarn (classic)	<code>/root/.cache/yarn/**/*</code>

Surveillance CodeArtifact

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité CodeArtifact et des performances de vos autres AWS solutions. AWS fournit les outils de surveillance suivants pour surveiller CodeArtifact, signaler tout problème et prendre des mesures automatiques le cas échéant :

- Vous pouvez utiliser Amazon EventBridge pour automatiser vos AWS services et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements AWS liés aux services sont diffusés EventBridge en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Pour plus d'informations, consultez le [guide de EventBridge l'utilisateur Amazon](#) et [CodeArtifact format d'événement et exemple](#).
- Vous pouvez utiliser CloudWatch les métriques Amazon pour afficher CodeArtifact l'utilisation par opération. CloudWatch les statistiques incluent toutes les demandes adressées à CodeArtifact, et les demandes sont affichées par compte. Vous pouvez consulter ces métriques sous forme de CloudWatch métriques en accédant à l'espace de noms Usage/By AWS AWS Resource. Pour plus d'informations, consultez la section [Utiliser CloudWatch les métriques Amazon](#) dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- [Surveillance des CodeArtifact événements](#)
- [Utiliser un événement pour démarrer une CodePipeline exécution](#)
- [Utiliser un événement pour exécuter une fonction Lambda](#)

Surveillance des CodeArtifact événements

CodeArtifact est intégré à Amazon EventBridge, un service qui automatise les événements et y répond, notamment les modifications apportées à un CodeArtifact référentiel. Vous pouvez créer des règles pour les événements et configurer ce qui se passe lorsqu'un événement correspond à une règle. EventBridge s'appelait auparavant CloudWatch Events.

Les actions suivantes peuvent être déclenchées par un événement :

- Invocation d'une AWS Lambda fonction.
- Activation d'une machine à AWS Step Functions états.
- Notification d'une rubrique Amazon SNS ou d'une file d'attente Amazon SQS.
- Démarrage d'un pipeline dans AWS CodePipeline.

CodeArtifact crée un événement lors de la création, de la modification ou de la suppression d'une version de package. Voici des exemples d' CodeArtifact événements :

- Publier une nouvelle version de package (par exemple, en exécutant `npm publish`).
- Ajouter un nouvel actif à une version de package existante (par exemple, en transférant un nouveau fichier JAR vers un package Maven existant).
- Copier une version de package d'un référentiel à un autre en utilisant `copy-package-versions`. Pour de plus amples informations, veuillez consulter [Copier des packages entre des référentiels](#).
- Suppression de versions de package à l'aide de `delete-package-versions`. Pour de plus amples informations, veuillez consulter [Supprimer un package ou une version de package](#).
- Supprimer les versions d'un package à l'aide de `delete-package`. Un événement sera publié pour chaque version du package supprimé. Pour de plus amples informations, veuillez consulter [Supprimer un package ou une version de package](#).
- Conserver la version d'un package dans un référentiel en aval lorsqu'elle a été extraite d'un référentiel en amont. Pour de plus amples informations, veuillez consulter [Utilisation de référentiels en amont dans CodeArtifact](#).
- Ingestion d'une version de package depuis un référentiel externe dans un CodeArtifact référentiel. Pour de plus amples informations, veuillez consulter [Connect un CodeArtifact dépôt à un dépôt public](#).

Les événements sont transmis à la fois au compte propriétaire du domaine et au compte qui administre le référentiel. Supposons, par exemple, que le compte 111111111111 possède le domaine `my_domain`. Le compte crée un référentiel dans `my_domain` called `repo2`. Lorsqu'une nouvelle version du package est publiée sur `repo2`, les deux comptes reçoivent les EventBridge événements. Le compte propriétaire du domaine (111111111111) reçoit des événements pour tous les référentiels du domaine. Si un seul compte possède à la fois le domaine et le référentiel qu'il contient, un seul événement est diffusé.

Les rubriques suivantes décrivent le format de l' CodeArtifact événement. Ils vous montrent comment configurer les CodeArtifact événements et comment les utiliser avec d'autres AWS services.

Pour plus d'informations, consultez [Getting Started with Amazon EventBridge](#) dans le guide de EventBridge l'utilisateur Amazon.

CodeArtifact format d'événement et exemple

Vous trouverez ci-dessous des champs et des descriptions d'événements, ainsi qu'un exemple d'CodeArtifact événement.

CodeArtifact format de l'événement

Tous les CodeArtifact événements incluent les champs suivants.

Champ d'événement	Description
version ;	Version du format de l'événement . Il n'existe actuellement qu'une seule version,0.
id	Identifiant unique de l'événement.
detail-type	Type d'événement. Cela détermine les champs de l'detailobjet. Celui detail-type actuellement pris en charge estCodeArtifact Package Version State Change .
source	La source de l'événement. Car CodeArtifact ce sera le casaws . codeartifact .
compte	L' AWS identifiant du compte qui reçoit l'événement.
time	Heure exacte à laquelle l'événement a été déclenché.
region	Région dans laquelle l'événement a été déclenché.
resources	Une liste qui contient l'ARN du package modifié. La liste contient une entrée. Pour plus d'informations sur le format ARN du package,

Champ d'événement	Description
	consultez Accorder l'accès en écriture aux packages .
domainName	Le domaine qui contient le référentiel contenant le package.
Propriétaire du domaine	L'ID de AWS compte du propriétaire du domaine.
Nom du référentiel	Le référentiel qui contient le package.
Administrateur du référentiel	ID de AWS compte de l'administrateur du référentiel.
Format du colis	Format du package qui a déclenché l'événement.
Espace de noms de paquets	L'espace de noms du package qui a déclenché l'événement.
Nom du package	Nom du package qui a déclenché l'événement.
Version du package	Version du package qui a déclenché l'événement.
packageVersionState	État de la version du package lorsque l'événement a été déclenché. Les valeurs admises sont <code>Unfinished</code> , <code>Published</code> , <code>Unlisted</code> , <code>Archived</code> et <code>Disposed</code> .
packageVersionRevision	Valeur qui identifie de manière unique l'état des actifs et des métadonnées de la version du package lorsque l'événement a été déclenché . Si la version du package est modifiée (par exemple, en ajoutant un autre fichier JAR à un package Maven), les <code>packageVersionRevision</code> modifications sont modifiées.

Champ d'événement	Description
Changements. Actifs ajoutés	Nombre de ressources ajoutées à un package qui ont déclenché un événement. Un fichier Maven JAR ou une roue Python sont des exemples d'actifs.
Changements. Actifs supprimés	Le nombre de ressources supprimées d'un package qui a déclenché un événement.
Changements. Actifs mis à jour	Le nombre de ressources modifiées dans le package qui a déclenché l'événement.
Changes.MetadataMis à jour	Valeur booléenne définie sur <code>true</code> si l'événement inclut des métadonnées modifiées au niveau du package. Par exemple, un événement peut modifier un <code>pom.xml</code> fichier Maven.
Changements. État modifié	Valeur booléenne définie sur <code>true</code> si l'événement <code>packageVersionStatus</code> est modifié (par exemple, s'il <code>packageVersionStatus</code> passe de <code>Unfinished</code> à <code>Published</code>).
Type d'opération	Décrit le type de haut niveau du changement de version du package. Les valeurs possibles sont <code>Created</code> , <code>Updated</code> et <code>Deleted</code> .

Champ d'événement	Description
sequenceNumber	Nombre entier qui indique le numéro d'événement d'un package. Chaque événement d'un package l'incrémente <code>sequenceNumber</code> afin que les événements puissent être organisés de manière séquentielle. Un événement peut l'incrémenter de <code>sequenceNumber</code> n'importe quel nombre entier.
eventDuplicationId	<p> Note</p> <p>EventBridge les événements peuvent être reçus dans le désordre. <code>sequenceNumber</code> peuvent être utilisés pour déterminer leur commande réelle.</p> <p>ID utilisé pour différencier les EventBridge événements dupliqués. Dans de rares cas, la même règle EventBridge peut être déclenchée plusieurs fois pour un seul événement ou à une heure planifiée. Il peut également invoquer la même cible plusieurs fois pour une règle déclenchée donnée.</p>

CodeArtifact exemple d'événement

Voici un exemple d'CodeArtifact événement susceptible d'être déclenché lors de la publication d'un package npm.

```
"time":"2019-11-21T23:19:54Z",
"region":"us-west-2",
"resources":["arn:aws:codeartifact:us-west-2:111122223333:package/my_domain/
myrepo/npm//mypackage"],
"detail": {
  "domainName": "my_domain",
  "domainOwner": "111122223333",
  "repositoryName": "myrepo",
  "repositoryAdministrator": "123456789012",
  "packageFormat": "npm",
  "packageNamespace": null,
  "packageName": "mypackage",
  "packageVersion": "1.0.0",
  "packageVersionState": "Published",
  "packageVersionRevision": "0E5DE26A4CD79FDF3EBC4924FFFFFF",
  "changes": {
    "assetsAdded": 1,
    "assetsRemoved": 0,
    "metadataUpdated": true,
    "assetsUpdated": 0,
    "statusChanged": true
  },
  "operationType": "Created",
  "sequenceNumber": 1,
  "eventDeduplicationId": "2mE00A2Ke07rWUTBxk3CAiQhdTXF4N94LNaT/fffff="
}
}
```

Utiliser un événement pour démarrer une CodePipeline exécution

Cet exemple montre comment configurer une EventBridge règle Amazon afin qu'une AWS CodePipeline exécution démarre lorsqu'une version de package est publiée, modifiée ou supprimée dans un CodeArtifact référentiel.

Rubriques

- [Configuration EventBridge des autorisations](#)
- [Création de la EventBridge règle](#)
- [Création de la cible de la EventBridge règle](#)

Configuration EventBridge des autorisations

Vous devez ajouter des autorisations EventBridge à utiliser CodePipeline pour invoquer la règle que vous créez. Pour ajouter ces autorisations à l'aide de AWS Command Line Interface (AWS CLI), suivez l'étape 1 de la [section Créez une règle d' CloudWatch événements pour une CodeCommit source \(CLI\)](#) dans le guide de AWS CodePipeline l'utilisateur.

Création de la EventBridge règle

Pour créer la règle, utilisez la `put-rule` commande avec les `--event-pattern` paramètres `--name` et. Le modèle d'événement spécifie des valeurs qui sont comparées au contenu de chaque événement. La cible est déclenchée si le schéma correspond à l'événement. Par exemple, le modèle suivant correspond CodeArtifact aux événements du `myrepo` référentiel du `my_domain` domaine.

```
aws events put-rule --name MyCodeArtifactRepoRule --event-pattern \  
'{"source":["aws.codeartifact"],"detail-type":["CodeArtifact Package Version State  
Change"],  
"detail":{"domainName":["my_domain"],"domainOwner":  
["111122223333"],"repositoryName":["myrepo"]}}'
```

Création de la cible de la EventBridge règle

La commande suivante ajoute une cible à la règle afin que lorsqu'un événement correspond à la règle, une CodePipeline exécution soit déclenchée. Pour le `RoleArn` paramètre, spécifiez l'Amazon Resource Name (ARN) du rôle créé précédemment dans cette rubrique.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \  
'Id=1,Arn=arn:aws:codepipeline:us-west-2:111122223333:pipeline-name,  
RoleArn=arn:aws:iam::123456789012:role/MyRole'
```

Utiliser un événement pour exécuter une fonction Lambda

Cet exemple montre comment configurer une EventBridge règle qui lance une AWS Lambda fonction lorsqu'une version de package d'un CodeArtifact référentiel est publiée, modifiée ou supprimée.

Pour plus d'informations, consultez [Tutorial : Utilisation AWS Lambda des fonctions de planification EventBridge](#) dans le guide de EventBridge l'utilisateur Amazon.

Rubriques

- [Création de la EventBridge règle](#)
- [Création de la cible de la EventBridge règle](#)
- [Configuration EventBridge des autorisations](#)

Création de la EventBridge règle

Pour créer une règle qui lance une fonction Lambda, utilisez la `put-rule` commande avec les options `--name` et `--event-pattern`. Le modèle suivant spécifie les packages npm dans le `@types` champ d'application de n'importe quel référentiel du `my_domain` domaine.

```
aws events put-rule --name "MyCodeArtifactRepoRule" --event-pattern \  
  '{"source": ["aws.codeartifact"], "detail-type": ["CodeArtifact Package Version State  
  Change"],  
  "detail": {"domainName": ["my_domain"], "domainOwner":  
  ["111122223333"], "packageNamespace": ["types"], "packageFormat": ["npm"]}}'
```

Création de la cible de la EventBridge règle

La commande suivante ajoute une cible à la règle qui exécute la fonction Lambda lorsqu'un événement correspond à la règle. Pour le `arn` paramètre, spécifiez l'Amazon Resource Name (ARN) de la fonction Lambda.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \  
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

Configuration EventBridge des autorisations

Utilisez la `add-permission` commande pour autoriser la règle à invoquer une fonction Lambda. Pour le `--source-arn` paramètre, spécifiez l'ARN de la règle que vous avez créée précédemment dans cet exemple.

```
aws lambda add-permission --function-name MyLambdaFunction \  
  --statement-id my-statement-id --action 'lambda:InvokeFunction' \  
  --principal events.amazonaws.com \  
  --source-arn arn:aws:events:us-west-2:111122223333:rule/MyCodeArtifactRepoRule
```

Sécurité dans CodeArtifact

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures 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 [modèle de responsabilité partagée](#) décrit la sécurité du cloud et la sécurité dans le cloud comme suit :

- 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 applicables à CodeArtifact, consultez la section [Services AWS concernés par programme de conformité](#) .
- 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 son utilisation CodeArtifact. Les rubriques suivantes expliquent comment procéder à la configuration CodeArtifact pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS qui vous aident à surveiller et à sécuriser vos CodeArtifact ressources.

Rubriques

- [Protection des données dans AWS CodeArtifact](#)
- [Surveillance CodeArtifact](#)
- [Validation de conformité pour AWS CodeArtifact](#)
- [AWS CodeArtifact authentification et jetons](#)
- [Résilience dans AWS CodeArtifact](#)
- [Sécurité de l'infrastructure dans AWS CodeArtifact](#)
- [Attaques de substitution de la dépendance](#)
- [Identity and Access Management pour AWS CodeArtifact](#)

Protection des données dans AWS CodeArtifact

Le [modèle de responsabilité AWS partagée](#) de s'applique à la protection des données dans AWS CodeArtifact. 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. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations 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 .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou 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 exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section [Utilisation des CloudTrail sentiers](#) dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez [Norme FIPS \(Federal Information Processing Standard\) 140-3](#).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec CodeArtifact ou d'autres Services

AWS utilisateurs de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte 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 des données

Le chiffrement est un élément important de la CodeArtifact sécurité. Certains chiffrements, par exemple pour les données en transit, sont fournis par défaut et ne nécessitent aucune intervention de votre part. Vous pouvez configurer d'autres types de chiffrement, par exemple pour les données au repos, lorsque vous créez votre projet ou votre build.

- Chiffrement des données au repos : tous les actifs stockés CodeArtifact sont chiffrés à l'aide de AWS KMS keys (clés KMS). Cela inclut tous les actifs de tous les packages de tous les référentiels. Une clé KMS est utilisée pour chaque domaine afin de chiffrer tous ses actifs. Par défaut, une clé KMS AWS gérée est utilisée, il n'est donc pas nécessaire de créer une clé KMS. Si vous le souhaitez, vous pouvez utiliser une clé KMS gérée par le client que vous créez et configurez. Pour plus d'informations, consultez les sections [Création de AWS clés et concepts du service de gestion des clés](#) dans le guide de AWS Key Management Service l'utilisateur. Vous pouvez spécifier une clé KMS gérée par le client lorsque vous créez un domaine. Pour de plus amples informations, veuillez consulter [Travailler avec des domaines dans CodeArtifact](#).
- Chiffrement des données en transit : toutes les communications entre les clients CodeArtifact et entre les clients CodeArtifact et leurs dépendances en aval sont protégées par le chiffrement TLS.

Confidentialité du trafic

Vous pouvez améliorer la sécurité de vos CodeArtifact domaines et des actifs qu'ils contiennent en les configurant de manière CodeArtifact à utiliser une interface de point de terminaison de cloud privé virtuel (VPC). Pour ce faire, vous n'avez pas besoin de passerelle Internet, de périphérique NAT ou de passerelle privée virtuelle. Pour de plus amples informations, veuillez consulter [Utilisation des points de terminaison Amazon VPC](#). Pour plus d'informations sur AWS PrivateLink les points de terminaison VPC, consultez et [AWS PrivateLink](#) Accès aux services [AWS via](#) PrivateLink

Surveillance CodeArtifact

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS CodeArtifact et des performances de vos AWS solutions. Vous devez 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. AWS fournit les éléments suivants pour surveiller vos CodeArtifact ressources et répondre aux incidents potentiels :

Rubriques

- [Journalisation des appels d' CodeArtifact API avec AWS CloudTrail](#)

Journalisation des appels d' CodeArtifact API avec AWS CloudTrail

CodeArtifact est intégré à [AWS CloudTrail](#) un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans CodeArtifact. CloudTrail capture tous les appels d'API CodeArtifact sous forme d'événements, y compris les appels des clients du gestionnaire de packages.

Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un bucket Amazon Simple Storage Service (Amazon S3), y compris les événements pour CodeArtifact. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite CodeArtifact, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le [guide de AWS CloudTrail l'utilisateur](#).

CodeArtifact informations dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans CodeArtifact, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section [Affichage des événements avec l'historique des CloudTrail événements](#).

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements pour CodeArtifact, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à

un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. Vous pouvez également configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les rubriques suivantes :

- [Création d'un parcours pour votre compte AWS](#)
- [CloudTrail Services et intégrations pris en charge](#)
- [Configuration des notifications Amazon SNS pour CloudTrail](#)

Lorsque la CloudTrail journalisation est activée dans votre AWS compte, les appels d'API effectués aux CodeArtifact actions sont suivis dans des fichiers CloudTrail journaux, où ils sont écrits avec d'autres enregistrements de AWS service. CloudTrail détermine à quel moment créer et écrire dans un nouveau fichier en fonction d'une période et de la taille du fichier.

Toutes les CodeArtifact actions sont enregistrées par CloudTrail. Par exemple, les appels aux actions `ListRepositories` (in the AWS CLI, `aws codeartifact list-repositories`), `CreateRepository` (`aws codeartifact create-repository`) et `ListPackages` (`aws codeartifact list-packages`) génèrent des entrées dans les fichiers CloudTrail journaux, en plus des commandes client du gestionnaire de packages. Les commandes client du gestionnaire de packages envoient généralement plusieurs requêtes HTTP au serveur. Chaque demande génère un événement de CloudTrail journal distinct.

Livraison de journaux entre comptes CloudTrail

Jusqu'à trois comptes distincts reçoivent des CloudTrail journaux pour un seul appel d'API :

- Le compte qui a fait la demande, par exemple, le compte qui a appelé `GetAuthorizationToken`
- Le compte d'administrateur du référentiel, par exemple, le compte qui administre le référentiel appelé `ListPackages`.
- Le compte du propriétaire du domaine, par exemple, le compte propriétaire du domaine contenant le référentiel sur lequel une API a été appelée.

APIs Comme `ListRepositoriesInDomain` il s'agit d'actions contre un domaine et non contre un dépôt spécifique, seuls le compte appelant et le compte du propriétaire du domaine reçoivent

le CloudTrail journal. Car APIs `ListRepositories` les appels ne sont autorisés contre aucune ressource, seul le compte de l'appelant reçoit le CloudTrail journal.

Comprendre les entrées du fichier CodeArtifact journal

CloudTrail les fichiers journaux peuvent contenir une ou plusieurs entrées de journal. Chaque entrée répertorie plusieurs événements au format JSON. Un événement de journal représente une demande individuelle émise à partir d'une source quelconque et comprend des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. Les entrées de journal ne sont pas des arborescences des appels de procédure des appels d'API publique. Elles ne s'affichent donc dans aucun ordre précis.

Rubriques

- [Exemple : entrée de journal pour appeler l' GetAuthorizationToken API](#)
- [Exemple : une entrée de journal pour récupérer la version d'un package npm](#)

Exemple : entrée de journal pour appeler l' `GetAuthorizationToken` API

Une entrée de journal créée par [GetAuthorizationToken](#) inclut le nom de domaine dans le `requestParameters` champ.

```
{  
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AssumedRole",  
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",  
    "arn": "arn:aws:sts::123456789012:assumed-role/Console/example",  
    "accountId": "123456789012",  
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
    "sessionContext": {  
      "attributes": {  
        "mfaAuthenticated": "false",  
        "creationDate": "2018-12-11T13:31:37Z"  
      },  
      "sessionIssuer": {  
        "type": "Role",  
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",  
        "arn": "arn:aws:iam::123456789012:role/Console",  
        "accountId": "123456789012",  
        "userName": "Console"  
      }  
    }  
  }  
}
```

```
        }
    },
    "eventTime": "2018-12-11T13:31:37Z",
    "eventSource": "codeartifact.amazonaws.com",
    "eventName": "GetAuthorizationToken",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.50",
    "userAgent": "aws-cli/1.16.37 Python/2.7.10 Darwin/16.7.0 botocore/1.12.27",
    "requestParameters": {
        "domainName": "example-domain"
        "domainOwner": "123456789012"
    },
    "responseElements": {
        "sessionToken": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "requestID": "6b342fc0-5bc8-402b-a7f1-ffffffffffff",
    "eventID": "100fde01-32b8-4c2b-8379-ffffffffffff",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Exemple : une entrée de journal pour récupérer la version d'un package npm

Les demandes effectuées par tous les clients du gestionnaire de packages, y compris le **npmclient**, ont des données supplémentaires enregistrées, notamment le nom de domaine, le nom du référentiel et le nom du package dans le `requestParameters` champ. Le chemin de l'URL et la méthode HTTP sont enregistrés dans le `additionalEventData` champ.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/Console/example",
        "accountId": "123456789012",
        "accessKeyId": "ASIAIJI0BJIBSREXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-12-17T02:05:16Z"
            },
        }
    },
}
```

```
        "sessionIssuer": {
            "type": "Role",
            "principalId": "AIDACKCEVSQ6C2EXAMPLE",
            "arn": "arn:aws:iam::123456789012:role/Console",
            "accountId": "123456789012",
            "userName": "Console"
        }
    }
},
"eventTime": "2018-12-17T02:05:46Z",
"eventSource": "codeartifact.amazonaws.com",
"eventName": "ReadFromRepository",
"awsRegion": "us-west-2",
"sourceIPAddress": "205.251.233.50",
"userAgent": "npm/6.14.15 node/v12.22.9 linux x64 ci/custom",
"requestParameters": {
    "domainName": "example-domain",
    "domainOwner": "123456789012",
    "repositoryName": "example-repo",
    "packageName": "lodash",
    "packageFormat": "npm",
    "packageVersion": "4.17.20"
},
"responseElements": null,
"additionalEventData": {
    "httpMethod": "GET",
    "requestUri": "/npm/lodash/-/lodash-4.17.20.tgz"
},
"requestID": "9f74b4f5-3607-4bb4-9229-ffffffffffff",
"eventID": "c74e40dd-8847-4058-a14d-ffffffffffff",
"readOnly": true,
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

Validation de conformité pour AWS CodeArtifact

Pour savoir si un [programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir [Programmes de AWS conformité Programmes AWS](#) de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#).

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez [AWS la documentation de sécurité](#).

AWS CodeArtifact authentification et jetons

CodeArtifact oblige les utilisateurs à s'authentifier auprès du service afin de publier ou de consommer des versions de package. Vous devez vous authentifier auprès du CodeArtifact service en créant un jeton d'autorisation à l'aide de vos AWS informations d'identification. Pour créer un jeton d'autorisation, vous devez disposer des autorisations appropriées. Pour connaître les autorisations nécessaires à la création d'un jeton d'autorisation, consultez l'GetAuthorizationTokenentrée dans le [AWS CodeArtifact référence aux autorisations](#). Pour des informations plus générales sur CodeArtifact les autorisations, voir [Comment AWS CodeArtifact fonctionne avec IAM](#).

Pour récupérer un jeton d'autorisation CodeArtifact, vous devez appeler l'[GetAuthorizationToken API](#). À l'aide de AWS CLI, vous pouvez appeler GetAuthorizationToken avec la get-authorization-token commande login ou.

Note

Les utilisateurs root ne peuvent pas appelerGetAuthorizationToken.

- `aws codeartifact login`: Cette commande facilite la configuration des gestionnaires de packages courants à utiliser CodeArtifact en une seule étape. L'appel permet de login récupérer un jeton GetAuthorizationToken et de configurer votre gestionnaire de packages avec le jeton et le point de terminaison de CodeArtifact dépôt correct. Les gestionnaires de packages de support sont les suivants :
 - dotnet
 - npm
 - pépite
 - pip
 - rapide

- ficelle
- aws codeartifact get-authorization-token: Pour les gestionnaires de packages non pris en charge par login, vous pouvez appeler get-authorization-token directement votre gestionnaire de packages puis le configurer avec le jeton selon vos besoins, par exemple en l'ajoutant à un fichier de configuration ou en le stockant dans une variable d'environnement.

CodeArtifact les jetons d'autorisation sont valides pendant une période par défaut de 12 heures. Les jetons peuvent être configurés avec une durée de vie comprise entre 15 minutes et 12 heures. Lorsque la durée de vie expire, vous devez récupérer un autre jeton. La durée de vie du jeton commence login ou get-authorization-token est appelée.

Si login ou get-authorization-token est appelé alors que vous assumez un rôle, vous pouvez configurer la durée de vie du jeton pour qu'elle soit égale au temps restant de la durée de session du rôle en définissant la valeur de --duration-seconds to0. Sinon, la durée de vie du jeton est indépendante de la durée de session maximale du rôle. Supposons, par exemple, que vous appeliez sts assume-role et spécifiez une durée de session de 15 minutes, puis que vous appelez login pour récupérer un jeton CodeArtifact d'autorisation. Dans ce cas, le jeton est valide pendant toute la période de 12 heures, même si celle-ci est supérieure à la durée de session de 15 minutes. Pour plus d'informations sur le contrôle de la durée des sessions, consultez la section [Utilisation des rôles IAM](#) dans le guide de l'utilisateur IAM.

Jetons créés avec la **login** commande

La aws codeartifact login commande récupérera un jeton GetAuthorizationToken et configurera votre gestionnaire de packages avec le jeton et le point de terminaison du CodeArtifact référentiel correct.

Le tableau suivant décrit les paramètres de la login commande.

Paramètre	Obligatoire	Description
--tool	Oui	Le gestionnaire de packages auprès duquel s'authentifier. Les valeurs possibles sont dotnet, npm, nuget, pip, swift et twine.

Paramètre	Obligatoire	Description
--domain	Oui	Le nom de domaine auquel appartient le référentiel.
--domain-owner	Non	L'ID du propriétaire du domaine. Ce paramètre est obligatoire si vous accédez à un domaine appartenant à un AWS compte auquel vous n'êtes pas authentifié. Pour de plus amples informations, veuillez consulter Domaines multi-comptes .
--repository	Oui	Nom du référentiel auprès duquel s'authentifier.
--duration-seconds	Non	Durée, en secondes, pendant laquelle les informations de connexion sont valides. La valeur minimale est 900* et la valeur maximale est 43200.
--namespace	Non	Associe un espace de noms à votre outil de référentiel.
--dry-run	Non	Imprimez uniquement les commandes qui seront exécutées pour connecter votre outil à votre référentiel sans apporter de modifications à votre configuration.

*Une valeur de 0 est également valide lors d'un appel `login` en assumant un rôle. L'appel `login` avec `--duration-seconds 0` crée un jeton dont la durée de vie est égale au temps restant dans la durée de session d'un rôle assumé.

L'exemple suivant montre comment récupérer un jeton d'autorisation à l'aide de la `login` commande.

```
aws codeartifact login \
```

```
--tool dotnet | npm | nuget | pip | swift | twine \
--domain my_domain \
--domain-owner 111122223333 \
--repository my_repo
```

Pour obtenir des instructions spécifiques sur l'utilisation de la `login` commande avec `npm`, consultez [Configurer et utiliser npm avec CodeArtifact](#). Pour Python, voir [Utilisation CodeArtifact avec Python](#).

Autorisations requises pour appeler l'**GetAuthorizationTokenAPI**

Les autorisations `sts:GetServiceBearerToken` et les `codeartifact:GetAuthorizationToken` autorisations sont requises pour appeler l' `CodeArtifact GetAuthorizationTokenAPI`.

Pour utiliser un gestionnaire de packages avec un CodeArtifact référentiel, votre utilisateur ou rôle IAM doit l'autoriser `sts:GetServiceBearerToken`. Bien qu'`sts:GetServiceBearerToken` ne puisse être ajoutée à une politique de ressources de CodeArtifact domaine, l'autorisation n'aura aucun effet sur cette politique.

Tokens créés avec l'**GetAuthorizationTokenAPI**

Vous pouvez appeler `get-authorization-token` pour récupérer un jeton d' `CodeArtifact` autorisation.

```
aws codeartifact get-authorization-token \
--domain my_domain \
--domain-owner 111122223333 \
--query authorizationToken \
--output text
```

Vous pouvez modifier la durée de validité d'un jeton à l'aide de l'`--duration-seconds` argument. La valeur minimale est 900 et la valeur maximale est 43200. L'exemple suivant crée un jeton qui durera 1 heure (3 600 secondes).

```
aws codeartifact get-authorization-token \
--domain my_domain \
--domain-owner 111122223333 \
--query authorizationToken \
```

```
--output text \
--duration-seconds 3600
```

Si vous appelez `get-authorization-token` en assumant un rôle, la durée de vie du jeton est indépendante de la durée de session maximale du rôle. Vous pouvez configurer le jeton pour qu'il expire lorsque la durée de session du rôle supposé expire en le définissant `--duration-seconds` sur 0.

```
aws codeartifact get-authorization-token \
--domain my_domain \
--domain-owner 111122223333 \
--query authorizationToken \
--output text \
--duration-seconds 0
```

Consultez la documentation suivante pour plus d'informations :

- Pour obtenir des conseils sur les jetons et les variables d'environnement, consultez [Transmettre un jeton d'authentification à l'aide d'une variable d'environnement](#).
- Pour les utilisateurs de Python, voir [Configurer pip sans la commande de connexion](#) ou [Configurer et utiliser Twine avec CodeArtifact](#).
- Pour les utilisateurs de Maven, voir [Utiliser CodeArtifact avec Gradle](#) ou [Utiliser CodeArtifact avec MVN](#).
- Pour les utilisateurs de npm, voir [Configuration de npm sans utiliser la commande de connexion](#).

Transmettre un jeton d'authentification à l'aide d'une variable d'environnement

AWS CodeArtifact utilise des jetons d'autorisation fournis par l'GetAuthorizationTokenAPI pour authentifier et autoriser les demandes provenant d'outils de compilation tels que Maven et Gradle. Pour plus d'informations sur ces jetons d'authentification, consultez [Tokens créés avec l'GetAuthorizationTokenAPI](#).

Vous pouvez stocker ces jetons d'authentification dans une variable d'environnement qui peut être lue par un outil de génération pour obtenir le jeton dont il a besoin pour récupérer des packages depuis un CodeArtifact référentiel ou y publier des packages.

Pour des raisons de sécurité, cette approche est préférable au stockage du jeton dans un fichier où il pourrait être lu par d'autres utilisateurs ou processus, ou enregistré accidentellement dans le contrôle de source.

1. Configurez vos AWS informations d'identification comme décrit dans [Installez ou mettez à niveau, puis configurez le AWS CLI](#).
2. Définissez la variable d'environnement :CODEARTIFACT_AUTH_TOKEN

 Note

Dans certains scénarios, il n'est pas nécessaire d'inclure l'--domain-owner argument. Pour de plus amples informations, veuillez consulter [Domaines multi-comptes](#).

- macOS ou Linux :

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text`
```

- Windows (en utilisant l'interface de commande par défaut) :

```
for /f %i in ('aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text') do set CODEARTIFACT_AUTH_TOKEN=%i
```

- Fenêtres PowerShell :

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --domain my_domain --domain-owner 111122223333 --query authorizationToken --output text
```

Révocation des jetons CodeArtifact d'autorisation

Lorsqu'un utilisateur authentifié crée un jeton pour accéder aux CodeArtifact ressources, ce jeton dure jusqu'à la fin de sa période d'accès personnalisable. La période d'accès par défaut est de 12 heures. Dans certains cas, vous souhaiterez peut-être révoquer l'accès à un jeton avant l'expiration

de la période d'accès. Vous pouvez révoquer l'accès aux CodeArtifact ressources en suivant ces instructions.

Si vous avez créé le jeton d'accès à l'aide d'informations d'identification de sécurité temporaires, telles que des rôles assumés ou un accès utilisateur fédéré, vous pouvez révoquer l'accès en mettant à jour une politique IAM afin de refuser l'accès. Pour plus d'informations, consultez la section [Désactivation des autorisations pour les informations d'identification de sécurité temporaires](#) dans le guide de l'utilisateur IAM.

Si vous avez utilisé des informations d'identification utilisateur IAM à long terme pour créer le jeton d'accès, vous devez modifier la politique de l'utilisateur pour refuser l'accès ou supprimer l'utilisateur IAM. Pour plus d'informations, voir [Modification des autorisations d'un utilisateur IAM ou Suppression d'un utilisateur IAM](#).

Résilience dans AWS CodeArtifact

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. AWS CodeArtifact fonctionne dans plusieurs zones de disponibilité et stocke les données et métadonnées des artefacts dans Amazon S3 et Amazon DynamoDB. Vos données cryptées sont stockées de manière redondante sur plusieurs installations et sur plusieurs appareils dans chaque installation, ce qui les rend hautement disponibles et très durables.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section [Infrastructure AWS mondiale](#).

Sécurité de l'infrastructure dans AWS CodeArtifact

En tant que service géré, AWS CodeArtifact il 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 CodeArtifact 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.

Attaques de substitution de la dépendance

Les gestionnaires de packages simplifient le processus d'empaquetage et de partage de code réutilisable. Ces packages peuvent être des packages privés développés par une organisation pour être utilisés dans ses applications, ou il peut s'agir de packages publics, généralement open source, développés en dehors d'une organisation et distribués par des référentiels de packages publics. Lorsqu'ils demandent des packages, les développeurs s'appuient sur leur gestionnaire de packages pour récupérer les nouvelles versions de leurs dépendances. Les attaques de substitution de dépendances, également appelées attaques de confusion des dépendances, exploitent le fait qu'un gestionnaire de packages n'a généralement aucun moyen de distinguer les versions légitimes d'un package des versions malveillantes.

Les attaques par substitution de dépendance appartiennent à un sous-ensemble de piratages connus sous le nom d'attaques de la chaîne logistique logicielle. Une attaque de chaîne d'approvisionnement logiciel est une attaque qui tire parti de vulnérabilités situées à n'importe quel niveau de la chaîne d'approvisionnement logiciel.

Une attaque de substitution de dépendance peut cibler toute personne utilisant à la fois des packages développés en interne et des packages extraits de référentiels publics. Les attaquants identifient les noms de packages internes, puis placent stratégiquement le code malveillant portant le même nom dans des référentiels de packages publics. Généralement, le code malveillant est publié dans un package dont le numéro de version est élevé. Les gestionnaires de packages récupèrent le code malveillant à partir de ces flux publics car ils pensent que les packages malveillants sont les dernières versions du package. Cela provoque une « confusion » ou une « substitution » entre le package souhaité et le package malveillant, entraînant la compromission du code.

Pour empêcher les attaques de substitution de dépendance, AWS CodeArtifact fournit des contrôles d'origine des packages. Les contrôles d'origine des packages sont des paramètres qui contrôlent la manière dont les packages peuvent être ajoutés à vos référentiels. Les contrôles peuvent être utilisés pour garantir que les versions des packages ne peuvent pas être à la fois publiées directement dans votre référentiel et ingérées à partir de sources publiques, vous protégeant ainsi des attaques de substitution de dépendances. Les contrôles d'origine peuvent être définis sur des emballages

individuels et sur plusieurs emballages en définissant des contrôles d'origine sur des groupes de colis. Pour plus d'informations sur les contrôles d'origine des packages et sur la manière de les modifier, consultez [Modification des contrôles d'origine des packages](#) et [Contrôles d'origine des groupes de packages](#).

Identity and Access Management pour AWS CodeArtifact

AWS Identity and Access Management (IAM) est un outil Service AWS 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 CodeArtifact les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion de l'accès à l'aide de politiques](#)
- [Comment AWS CodeArtifact fonctionne avec IAM](#)
- [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#)
- [Utilisation de balises pour contrôler l'accès aux ressources CodeArtifact](#)
- [AWS CodeArtifact référence aux autorisations](#)
- [Résolution des problèmes AWS CodeArtifact d'identité et d'accès](#)

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir [Résolution des problèmes AWS CodeArtifact d'identité et d'accès](#))
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir [Comment AWS CodeArtifact fonctionne avec IAM](#))
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#))

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification Google/Facebook. Pour plus d'informations sur la connexion, consultez [Connexion à votre Compte AWS](#) dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez [Signature AWS Version 4 pour les demandes d'API](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez [Tâches qui requièrent les informations d'identification de l'utilisateur racine](#) dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez [Qu'est-ce que IAM Identity Center ?](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir [Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires](#) dans le guide de l'utilisateur IAM.

[Les groupes IAM](#) spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez [Cas d'utilisation pour les utilisateurs IAM](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en [passant d'un rôle utilisateur à un rôle IAM \(console\)](#) ou en appelant une opération AWS CLI ou AWS API. Pour plus d'informations, consultez [Méthodes pour endosser un rôle](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès entre comptes, les accès entre services et pour les applications exécutées sur Amazon. EC2 Pour plus d'informations, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Politiques basées sur l'identité

Les stratégies basées sur l'identité sont des documents de stratégie d'autorisations JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) — Spécifiez les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez [Politiques de contrôle de service](#) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir [Politiques de contrôle des ressources \(RCPs\)](#) dans le guide de AWS Organizations l'utilisateur.

- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez [Politiques de session](#) dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

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.

Comment AWS CodeArtifact fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à CodeArtifact, découvrez les fonctionnalités IAM disponibles. CodeArtifact

Fonctionnalités IAM que vous pouvez utiliser avec AWS CodeArtifact

Fonctionnalité IAM	CodeArtifact soutien
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Oui
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Non
ACLs	Non
ABAC (identifications dans les politiques)	Partielle
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Rôles du service	Non

Fonctionnalité IAM	CodeArtifact soutien
<u>Rôles liés à un service</u>	Non

Pour obtenir une vue d'ensemble de la façon dont CodeArtifact les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez la section [AWS Services compatibles avec IAM](#) dans le Guide de l'utilisateur IAM.

Politiques basées sur l'identité pour CodeArtifact

Prend en charge les politiques basées sur l'identité : oui

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 [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

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. Pour découvrir 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.

Exemples de politiques basées sur l'identité pour CodeArtifact

Pour consulter des exemples de politiques CodeArtifact basées sur l'identité, consultez. [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#)

Politiques basées sur les ressources au sein de CodeArtifact

Prend en charge les politiques basées sur les ressources : oui

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. 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 Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d'informations, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Actions politiques pour CodeArtifact

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des CodeArtifact actions, reportez-vous à la section [Actions définies par AWS CodeArtifact](#) dans la référence d'autorisation de service.

Les actions de politique en CodeArtifact cours utilisent le préfixe suivant avant l'action :

```
codeartifact
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
    "codeartifact:action1",
    "codeartifact:action2"
]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "codeartifact:Describe*"
```

Pour consulter des exemples de politiques CodeArtifact basées sur l'identité, consultez. [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#)

Ressources politiques pour CodeArtifact

Prend en charge les ressources de politique : oui

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 de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son [Amazon Resource Name \(ARN\)](#). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de CodeArtifact ressources et leurs caractéristiques ARNs, voir [Ressources définies par AWS CodeArtifact](#) dans la référence d'autorisation de service. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez [Actions définies par AWS CodeArtifact](#). Pour voir des exemples de spécification de CodeArtifact ressources ARNs dans les politiques, consultez [AWS CodeArtifact ressources et opérations](#).

Clés de conditions de politique pour CodeArtifact

Prend en charge les clés de condition de politique spécifiques au service : non

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 `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des [opérateurs de condition](#), tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de [contexte de condition AWS globales](#) dans le guide de l'utilisateur IAM.

Note

AWS CodeArtifact ne prend pas en charge les clés contextuelles de condition AWS globale suivantes :

- [Référent](#)
- [UserAgent](#)

Pour consulter la liste des clés de CodeArtifact condition, voir [Clés de condition pour AWS CodeArtifact](#) la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la section [Actions définies par AWS CodeArtifact](#).

Pour consulter des exemples de politiques CodeArtifact basées sur l'identité, consultez. [Exemples de politiques basées sur l'identité pour AWS CodeArtifact](#)

ACLs in CodeArtifact

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec CodeArtifact

Prend en charge ABAC (identifications dans les politiques) : partiellement

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des attributs nommés balise. Vous pouvez associer des balises aux entités et aux AWS ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans [l'élément de condition](#) d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez [Définition d'autorisations avec l'autorisation ABAC](#) dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez [Utilisation du contrôle d'accès par attributs \(ABAC\)](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur le balisage CodeArtifact des ressources, notamment des exemples de politiques basées sur l'identité permettant de limiter l'accès à une ressource en fonction des balises associées à cette ressource, consultez [Utilisation de balises pour contrôler l'accès aux ressources CodeArtifact](#)

Utilisation d'informations d'identification temporaires avec CodeArtifact

Prend en charge les informations d'identification temporaires : oui

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez [Informations d'identification de sécurité temporaires dans IAM](#) et [Services AWS compatibles avec IAM](#) dans le Guide de l'utilisateur IAM.

Autorisations principales interservices pour CodeArtifact

Prend en charge les sessions d'accès direct (FAS) : oui

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section [Sessions de transmission d'accès](#).

Deux actions d' CodeArtifact API nécessitent que le principal appelant dispose d'autorisations dans d'autres services :

1. `GetAuthorizationToken` nécessite `sts:GetServiceBearerToken` avec `codeartifact:GetAuthorizationToken`.
2. `CreateDomain`, lorsque vous fournissez une clé de chiffrement autre que celle par défaut, nécessite les deux `kms:DescribeKey` et `kms>CreateGrant` sur la clé KMS en même temps. `codeartifact>CreateDomain`

Pour plus d'informations sur les autorisations et les ressources requises pour les actions dans CodeArtifact, consultez [AWS CodeArtifact référence aux autorisations](#).

Rôles de service pour CodeArtifact

Prend en charge les rôles de service : Non

Un rôle de service est un [rôle IAM](#) qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.

Warning

La modification des autorisations associées à un rôle de service peut perturber CodeArtifact les fonctionnalités. Modifiez les rôles de service uniquement lorsque CodeArtifact vous recevez des instructions à cet effet.

Rôles liés à un service pour CodeArtifact

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un Service AWS. 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.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez [Services AWS qui fonctionnent avec IAM](#). Recherchez un service dans le tableau qui inclut un Yes dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité pour AWS CodeArtifact

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources CodeArtifact. 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.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez [Création de politiques IAM \(console\)](#) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par CodeArtifact, y compris le format de ARNs pour chacun des types de ressources, voir [Actions, ressources et clés de condition AWS CodeArtifact](#) dans la référence d'autorisation de service.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console CodeArtifact](#)
- [Stratégies gérées par AWS \(prédéfinies\) pour AWS CodeArtifact](#)
- [Autoriser un utilisateur à consulter ses propres autorisations](#)
- [Autoriser un utilisateur à obtenir des informations sur les référentiels et les domaines](#)
- [Autoriser un utilisateur à obtenir des informations sur des domaines spécifiques](#)
- [Autoriser un utilisateur à obtenir des informations sur des référentiels spécifiques](#)
- [Limiter la durée du jeton d'autorisation](#)

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer CodeArtifact des ressources dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [politiques gérées par AWS](#) ou [politiques gérées par AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre

privilege. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.

- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politiques avec IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Sécurisation de l'accès aux API avec MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

Utilisation de la console CodeArtifact

Pour accéder à la AWS CodeArtifact console, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails CodeArtifact des ressources de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la CodeArtifact console, associez également la politique `AWSCodeArtifactAdminAccess` ou la politique `AWSCodeArtifactReadOnlyAccess` AWS gérée aux entités. Pour plus d'informations, consultez [Ajout d'autorisations à un utilisateur](#) dans le Guide de l'utilisateur IAM.

Stratégies gérées par AWS (prédéfinies) pour AWS CodeArtifact

AWS répond à de nombreux cas d'utilisation courants en fournissant des politiques IAM autonomes créées et administrées par AWS. Ces politiques AWS gérées accordent les autorisations nécessaires pour les cas d'utilisation courants afin que vous n'ayez pas à rechercher les autorisations nécessaires. Pour de plus amples informations, veuillez consulter [Stratégies gérées par AWS](#) dans le Guide de l'utilisateur IAM.

Les politiques AWS gérées suivantes, que vous pouvez associer aux utilisateurs de votre compte, sont spécifiques à AWS CodeArtifact.

- `AWSCodeArtifactAdminAccess`— Fournit un accès complet CodeArtifact, y compris les autorisations d'administration des CodeArtifact domaines.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:*"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]}
```

```
}
```

- **AWSCodeArtifactReadOnlyAccess**— Fournit un accès en lecture seule à CodeArtifact JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "codeartifact:Describe*",  
                "codeartifact:Get*",  
                "codeartifact>List*",  
                "codeartifact:ReadFromRepository"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "sts:GetServiceBearerToken",  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"  
                }  
            }  
        }  
    ]  
}
```

Pour créer et gérer des rôles de CodeArtifact service, vous devez également associer la politique AWS gérée nommée **IAMFullAccess**.

Vous pouvez également créer vos propres politiques IAM personnalisées afin d'accorder des autorisations pour les actions et les ressources CodeArtifact. Vous pouvez attacher ces politiques personnalisées aux utilisateurs ou groupes IAM qui nécessitent ces autorisations.

Autoriser un utilisateur à consulter ses 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": "*"  
    }  
  ]  
}
```

Autoriser un utilisateur à obtenir des informations sur les référentiels et les domaines

La politique suivante permet à un utilisateur ou à un rôle IAM de répertorier et de décrire tout type de CodeArtifact ressource, y compris les domaines, les référentiels, les packages et les actifs. La politique inclut également l'`codeArtifact:ReadFromRepository` autorisation, qui permet au principal de récupérer des packages depuis un CodeArtifact référentiel. Il n'autorise pas la création de nouveaux domaines ou référentiels et n'autorise pas la publication de nouveaux packages.

Les `sts:GetServiceBearerToken` autorisations `codeartifact:GetAuthorizationToken` et sont requises pour appeler l'`GetAuthorizationTokenAPI`.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "codeartifact>List*",  
        "codeartifactDescribe*",  
        "codeartifactGet*",  
        "codeartifactRead*"  
      ],  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

Autoriser un utilisateur à obtenir des informations sur des domaines spécifiques

Voici un exemple de politique d'autorisation qui permet à un utilisateur de répertorier des domaines uniquement dans la us-east-2 région pour créer un compte 123456789012 pour tout domaine commençant par le nommy.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "codeartifact>ListDomains",  
      "Resource": "arn:aws:codeartifact:us-east-2:111122223333:domain/my*"  
    }  
  ]  
}
```

Autoriser un utilisateur à obtenir des informations sur des référentiels spécifiques

Voici un exemple de politique d'autorisation qui permet à un utilisateur d'obtenir des informations sur les référentiels se terminant par test, notamment des informations sur les packages qu'ils contiennent. L'utilisateur ne sera pas en mesure de publier, de créer ou de supprimer des ressources.

Les sts:GetServiceBearerToken autorisations codeartifact:GetAuthorizationToken et sont requises pour appeler l'GetAuthorizationTokenAPI.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "codeartifact>List*",  
        "codeartifact>Describe*",  
        "codeartifact>Get*"  
      ]  
    }  
  ]  
}
```

```
        "codeartifact:Read*"
    ],
    "Resource": "arn:aws:codeartifact:*:*:repository/*/*test"
},
{
    "Effect": "Allow",
    "Action": [
        "codeartifact>List*",
        "codeartifactDescribe*"
    ],
    "Resource": "arn:aws:codeartifact:*:*:package/*/*test/*/*/*"
},
{
    "Effect": "Allow",
    "Action": "sts:GetServiceBearerToken",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "sts:AWSServiceName": "codeartifact.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": "codeartifact:GetAuthorizationToken",
    "Resource": "*"
}
]
}
```

Limiter la durée du jeton d'autorisation

Les utilisateurs doivent s'authentifier à l' CodeArtifact aide de jetons d'autorisation pour publier ou consommer des versions de packages. Les jetons d'autorisation ne sont valides que pendant leur durée de vie configurée. Les jetons ont une durée de vie par défaut de 12 heures. Pour plus d'informations sur les jetons d'autorisation, consultez[AWS CodeArtifact authentification et jetons](#).

Lors de la récupération d'un jeton, les utilisateurs peuvent configurer la durée de vie du jeton. Les valeurs valides pour la durée de vie d'un jeton d'autorisation sont 0, et tout nombre compris entre 900 (15 minutes) et 43200 (12 heures). La valeur de 0 créera un jeton d'une durée égale aux informations d'identification temporaires du rôle de l'utilisateur.

Les administrateurs peuvent limiter les valeurs valides pendant la durée de vie d'un jeton d'autorisation en utilisant la clé de `sts:DurationSeconds` condition figurant dans la politique d'autorisation attachée à l'utilisateur ou au groupe. Si l'utilisateur tente de créer un jeton d'autorisation dont la durée de vie est supérieure aux valeurs valides, la création du jeton échouera.

Les exemples de politiques suivants limitent les durées possibles d'un jeton d'autorisation créé par CodeArtifact les utilisateurs.

Exemple de politique : limiter la durée de vie du jeton à exactement 12 heures (43 200 secondes)

Avec cette politique, les utilisateurs ne pourront créer que des jetons d'autorisation d'une durée de vie de 12 heures.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "codeartifact:*",  
      "Resource": "*"  
    },  
    {  
      "Sid": "sts",  
      "Effect": "Allow",  
      "Action": "sts:GetServiceBearerToken",  
      "Resource": "*",  
      "Condition": {  
        "NumericEquals": {  
          "sts:DurationSeconds": 43200  
        },  
        "StringEquals": {  
          "sts:AWSServiceName": "codeartifact.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

Exemple de politique : Limiter la durée de vie du jeton entre 15 minutes et 1 heure, ou égale à la période d'identification temporaire de l'utilisateur

Grâce à cette politique, les utilisateurs pourront créer des jetons dont la durée de validité est comprise entre 15 minutes et 1 heure. Les utilisateurs pourront également créer un jeton qui durera la durée des informations d'identification temporaires de leur rôle en spécifiant `0` pour `--durationSeconds`.

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codeartifact:*",  
            "Resource": "*"  
        },  
        {  
            "Sid": "sts",  
            "Effect": "Allow",  
            "Action": "sts:GetServiceBearerToken",  
            "Resource": "*",  
            "Condition": {  
                "NumericLessThanEquals": {  
                    "sts:DurationSeconds": 3600  
                },  
                "StringEquals": {  
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"  
                }  
            }  
        }  
    ]  
}
```

Utilisation de balises pour contrôler l'accès aux ressources CodeArtifact

Les conditions figurant dans les déclarations de politique utilisateur IAM font partie de la syntaxe que vous utilisez pour spécifier les autorisations relatives aux ressources requises par CodeArtifact les actions. L'utilisation des balises dans les conditions est un moyen de contrôler l'accès aux ressources et demandes. Pour plus d'informations sur le balisage CodeArtifact des ressources, consultez [Balisage des ressources](#). Cette rubrique explique le contrôle d'accès basé sur les balises.

Lorsque vous concevez des stratégies IAM, vous pouvez définir des autorisations granulaires en accordant l'accès à des ressources spécifiques. Au fur et à mesure que le nombre de ressources que vous gérez s'accroît, cette tâche devient plus difficile. Le balisage des ressources et l'utilisation de balises dans les déclarations de politique peuvent rendre cette tâche plus facile. Vous accordez l'accès en bloc à toute ressource avec une balise spécifique. Puis, vous appliquez cette balise à plusieurs reprises aux ressources correspondantes, lors de la création ou ultérieurement.

Les balises peuvent être attachées à la ressource ou transmises dans la demande aux services qui prennent en charge le balisage. Dans CodeArtifact, les ressources peuvent avoir des balises, et certaines actions peuvent inclure des balises. Lorsque vous créez une politique IAM, vous pouvez utiliser des clés de condition de balise pour contrôler :

- Quels utilisateurs peuvent effectuer des actions sur un domaine ou une ressource de référentiel, en fonction des balises qu'il possède déjà.
- quelles balises peuvent être transmises dans une demande d'action ;
- si des clés de balise spécifiques peuvent être utilisées dans une demande.

Pour connaître la syntaxe complète et la sémantique des clés de condition de balise, consultez [Contrôle de l'accès à l'aide de balises](#) dans le Guide de l'utilisateur IAM.

Important

Lorsque vous utilisez des balises sur des ressources pour limiter les actions, les balises doivent se trouver sur la ressource sur laquelle l'action agit. Par exemple, pour refuser `DescribeRepository` des autorisations avec des balises, celles-ci doivent se trouver sur chaque référentiel et non sur le domaine. Consultez [AWS CodeArtifact référence aux autorisations](#) la liste des actions CodeArtifact et des ressources sur lesquelles elles opèrent.

Exemples de contrôle d'accès basé sur des balises

Les exemples suivants montrent comment spécifier des conditions de balises dans les stratégies pour les utilisateurs CodeArtifact .

Example 1 : Limiter les actions en fonction des balises dans la demande

La politique des utilisateurs AWSCodeArtifactAdminAccess gérés donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodeArtifact action sur n'importe quelle ressource.

La politique suivante limite ce pouvoir et refuse aux utilisateurs non autorisés l'autorisation de créer des référentiels à moins que la demande ne contienne certaines balises. Pour ce faire, il refuse l'CreateRepositoryaction si la demande ne spécifie pas de balise nommée costcenter avec l'une des valeurs 1 ou2. L'administrateur d'un client peut attacher cette stratégie IAM aux utilisateurs IAM non autorisés et à la stratégie d'utilisateur gérée.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": "codeartifact>CreateRepository",  
      "Resource": "*",  
      "Condition": {  
        "Null": {  
          "aws:RequestTag/costcenter        }  
      }  
    },  
    {  
      "Effect": "Deny",  
      "Action": "codeartifact>CreateRepository",  
      "Resource": "*",  
      "Condition": {  
        "ForAnyValue:StringNotEquals": {  
          "aws:RequestTag/costcenter            "1",  
            "2"  
          ]  
        }  
      }  
    }  
  ]  
}
```

```
        ]
    }
}
]
}
```

Example 2 : Limiter les actions en fonction des balises de ressource

La politique des utilisateurs `AWSCodeArtifactAdminAccess` gérés donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodeArtifact action sur n'importe quelle ressource.

La politique suivante limite ce pouvoir et refuse aux utilisateurs non autorisés l'autorisation d'effectuer des actions sur des référentiels dans des domaines spécifiques. Pour ce faire, elle refuse certaines actions si la ressource possède une balise nommée `Key1` avec une valeur `Value1` ou `Value2`. (La clé de condition `aws:ResourceTag` est utilisée pour contrôler l'accès aux ressources en fonction des balises sur ces ressources.) L'administrateur d'un client peut attacher cette stratégie IAM aux utilisateurs IAM non autorisés et à la stratégie d'utilisateur gérée.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codeartifact:TagResource",
        "codeartifact:UntagResource",
        "codeartifact:DescribeDomain",
        "codeartifact:DescribeRepository",
        "codeartifact:PutDomainPermissionsPolicy",
        "codeartifact:PutRepositoryPermissionsPolicy",
        "codeartifact>ListRepositoriesInDomain",
        "codeartifact:UpdateRepository",
        "codeartifact:ReadFromRepository",
        "codeartifact>ListPackages",
        "codeartifact>ListTagsForResource"
      ],
      "Resource": "*",
    }
  ]
}
```

```
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/Key1": ["Value1", "Value2"]
    }
  }
}
```

Example 3 : Autoriser les actions en fonction des balises de ressources

La politique suivante autorise les utilisateurs à effectuer des actions et à obtenir des informations sur les référentiels et les packages qu'ils contiennent. CodeArtifact

Pour ce faire, il autorise des actions spécifiques si le référentiel possède une balise nommée Key1 avec la valeurValue1. (La clé de condition aws :RequestTag est utilisée pour contrôler les balises qui peuvent être transmises dans une demande IAM.) La condition aws :TagKeys garantit que la clé de balise est sensible à la casse. Cette stratégie est utile pour les utilisateurs IAM auxquels la stratégie utilisateur gérée AWSCodeArtifactAdminAccess n'est pas attachée. La politique gérée donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodeArtifact action sur n'importe quelle ressource.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact>ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

```
 ]  
 }
```

Example 4 : Autoriser les actions en fonction des balises de la demande

La politique suivante autorise les utilisateurs à créer des référentiels dans des domaines spécifiques dans CodeArtifact.

Pour ce faire, elle autorise les TagResource actions `CreateRepository` et si l'API de création de ressources dans la demande spécifie une balise nommée `Key1` avec la valeur `Value1`. (La clé de condition `aws:RequestTag` est utilisée pour contrôler les balises qui peuvent être transmises dans une demande IAM.) La condition `aws:TagKeys` garantit que la clé de balise est sensible à la casse. Cette stratégie est utile pour les utilisateurs IAM auxquels la stratégie utilisateur gérée `AWSCodeArtifactAdminAccess` n'est pas attachée. La politique gérée donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodeArtifact action sur n'importe quelle ressource.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "codeartifact>CreateRepository",  
        "codeartifact:TagResource"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "aws:RequestTag/Key1": "Value1"  
        }  
      }  
    }  
  ]  
}
```

AWS CodeArtifact référence aux autorisations

AWS CodeArtifact ressources et opérations

Dans AWS CodeArtifact, la ressource principale est un domaine. Dans une stratégie, vous utilisez un Amazon Resource Name (ARN) pour identifier la ressource à laquelle la stratégie s'applique. Les référentiels sont également des ressources auxquelles ils sont ARNs associés. Pour plus d'informations, consultez [Amazon Resource Names \(ARNs\)](#) dans le Référence générale d'Amazon Web Services.

Type de ressource	Format ARN
Domain	arn:aws:codeartifact: <i>region-ID :account-ID :domain/my_domain</i>
Référentiel	arn:aws:codeartifact: <i>region-ID :account-ID :repository/ my_domain /my_repo</i>
Groupe de packages	arn:aws:codeartifact: <i>region-ID :account-ID :package-group/ my_domain /encoded_package_group_pattern</i>
Package avec un espace de noms	arn:aws:codeartifact: <i>region-ID :account-ID :package/ my_domain /my_repo/package-format /namespace /my_package</i>
Package sans espace de noms	arn:aws:codeartifact: <i>region-ID :account-ID :package/ my_domain /my_repo/package-format //my_package</i>
Toutes les CodeArtifact ressources	arn:aws:codeartifact:*
Toutes les CodeArtifact ressources détenues par le compte spécifié dans la région AWS spécifiée	arn:aws:codeartifact: <i>region-ID :account-ID :*</i>

L'ARN de ressource que vous spécifiez dépend de l'action ou des actions auxquelles vous souhaitez contrôler l'accès.

Vous pouvez indiquer un domaine spécifique (*myDomain*) dans votre déclaration à l'aide de son ARN comme suit.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

Vous pouvez indiquer un repository (*myRepo*) spécifique dans votre instruction à l'aide de son ARN comme suit.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

Pour spécifier plusieurs ressources dans une seule instruction, séparez-les ARNs par des virgules. La déclaration suivante s'applique à tous les packages et référentiels d'un domaine spécifique.

```
"Resource": [
  "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain",
  "arn:aws:codeartifact:us-east-2:123456789012:repository/myDomain/*",
  "arn:aws:codeartifact:us-east-2:123456789012:package/myDomain/*"
]
```

Note

De nombreux AWS services considèrent les deux points (:) ou les barres obliques (/) comme le même caractère dans ARNs. Cependant, CodeArtifact utilise une correspondance exacte dans les modèles de ressources et les règles. Veillez à utiliser les caractères corrects lors de la création de modèles d'événements, afin qu'ils correspondent à la syntaxe ARN de la ressource.

AWS CodeArtifact Opérations et autorisation de l'API

Vous pouvez utiliser le tableau suivant comme référence lorsque vous configurez le contrôle d'accès et que vous rédigez des politiques d'autorisation que vous pouvez associer à une identité IAM (politiques basées sur l'identité).

Vous pouvez utiliser des AWS clés de condition larges dans vos AWS CodeArtifact politiques pour exprimer des conditions. Pour obtenir une liste, reportez-vous à la section [Référence des éléments de politique IAM JSON](#) dans le guide de l'utilisateur IAM.

Vous spécifiez les actions dans le champ Action de la politique. Pour spécifier une action, utilisez le préfixe codeartifact: suivi du nom de l'opération d'API (par exemple, codeartifact:CreateDomain ou codeartifact:AssociateExternalConnection). Pour spécifier plusieurs actions dans une même instruction, séparez-les par une virgule (par exemple, "Action": ["codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection"]).

Utilisation de caractères génériques

Vous spécifiez un ARN, avec ou sans caractère générique (*) comme valeur de ressource dans le champ Resource de la stratégie. Vous pouvez utiliser un caractère générique pour spécifier plusieurs actions ou ressources. Par exemple, codeartifact: * spécifie toutes les CodeArtifact actions et codeartifact:Describe* indique toutes les CodeArtifact actions commençant par le motDescribe.

Groupe de packages ARNs

Note

Cette section explique comment le codage des groupes de packages ARNs et des modèles est informatif. Il est recommandé ARNs de copier depuis la console ou de récupérer à ARNs l'aide de l'DescribePackageGroupAPI plutôt que de coder des modèles et de construire ARNs.

Les politiques IAM utilisent le caractère générique pour correspondre à plusieurs actions IAM ou à plusieurs ressources. * Les modèles de groupes de packages utilisent également le * caractère. Afin de rédiger plus facilement des politiques IAM correspondant à un seul groupe de packages, le format ARN du groupe de packages utilise une version codée du modèle de groupe de packages.

Plus précisément, le format ARN du groupe de packages est le suivant :

```
arn:aws:codeartifact:region:account-ID:package-group/my_domain/encoded_package_group_pattern
```

Où le modèle de groupe de packages codé est le modèle de groupe de packages, certains caractères spéciaux étant remplacés par leurs valeurs codées en pourcentage. La liste suivante contient les caractères et leurs valeurs codées en pourcentage correspondantes :

- * : %2a
- \$: %24
- % : %25

Par exemple, l'ARN d'un groupe de packages racine d'un domaine, /*), serait :

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

Notez que les caractères non inclus dans la liste ne peuvent pas être codés et qu' ARNs ils distinguent les majuscules et minuscules. Ils * doivent donc être codés tels %2a ou non%2A.

Résolution des problèmes AWS CodeArtifact d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec CodeArtifact IAM.

Rubriques

- [Je ne suis pas autorisé à effectuer une action dans CodeArtifact](#)
- [Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes CodeArtifact ressources](#)

Je ne suis pas autorisé à effectuer une action dans CodeArtifact

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM mateojackson tente d'utiliser la console pour afficher des informations détaillées sur une ressource *my-example-widget* fictive, mais ne dispose pas des autorisations `codeartifact:GetWidget` fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
codeartifact:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur mateojackson doit être mise à jour pour autoriser l'accès à la ressource *my-example-widget* à l'aide de l'action `codeartifact:GetWidget`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes CodeArtifact ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises CodeArtifact en charge, consultez [Comment AWS CodeArtifact 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.

Utilisation des points de terminaison Amazon VPC

Vous pouvez configurer CodeArtifact pour utiliser un point de terminaison de cloud privé virtuel (VPC) d'interface afin d'améliorer la sécurité de votre VPC.

Les points de terminaison VPC utilisent AWS PrivateLink un service qui vous permet d'y accéder CodeArtifact APIs via des adresses IP privées. AWS PrivateLink restreint tout le trafic réseau entre votre VPC CodeArtifact et le réseau AWS. Lorsque vous utilisez un point de terminaison VPC d'interface, vous n'avez pas besoin de passerelle Internet, de périphérique NAT ou de passerelle privée virtuelle. Pour de plus amples informations, consultez la section [Points de terminaison VPC](#) dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Important

- Les points de terminaison VPC ne prennent pas en charge les demandes interrégionales. AWS Assurez-vous de créer votre point de terminaison dans la même AWS région que celle à laquelle vous prévoyez d'émettre vos appels d'API CodeArtifact.
- Les points de terminaison d'un VPC prennent uniquement en charge le DNS fourni par Amazon via Amazon Route 53. Si vous souhaitez utiliser votre propre DNS, vous pouvez utiliser le transfert DNS conditionnel. Pour plus d'informations, consultez les [ensembles d'options DHCP](#) dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.
- Le groupe de sécurité attaché au point de terminaison d'un VPC doit autoriser les connexions entrantes sur le port 443 à partir du sous-réseau privé du VPC.

Rubriques

- [Créez des points de terminaison VPC pour CodeArtifact](#)
- [Créer le point de terminaison de la passerelle Amazon S3](#)
- [Utilisation CodeArtifact depuis un VPC](#)
- [Création d'une stratégie de point de terminaison de VPC pour CodeArtifact](#)

Créez des points de terminaison VPC pour CodeArtifact

Pour créer des points de terminaison de cloud privé virtuel (VPC) pour CodeArtifact, utilisez la commande Amazon EC2 `create-vpc-endpoint` AWS CLI Pour plus d'informations, consultez

[Interface VPC Endpoints \(AWS PrivateLink\)](#) dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

Deux points de terminaison VPC sont nécessaires pour que toutes les demandes CodeArtifact soient transmises au réseau. AWS Le premier point de terminaison est utilisé pour appeler CodeArtifact APIs (par exemple, `GetAuthorizationToken` et `CreateRepository`).

```
com.amazonaws.region.codeartifact.api
```

Le deuxième point de terminaison est utilisé pour accéder aux CodeArtifact référentiels à l'aide de gestionnaires de packages et d'outils de génération (par exemple, npm et Gradle).

```
com.amazonaws.region.codeartifact.repositories
```

La commande suivante crée un point de terminaison pour accéder aux CodeArtifact référentiels.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --vpc-endpoint-type Interface \  
--service-name com.amazonaws.region.codeartifact.api --subnet-ids subnetid \  
--security-group-ids groupid --private-dns-enabled
```

La commande suivante crée un point de terminaison pour accéder aux gestionnaires de packages et aux outils de génération.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --vpc-endpoint-type Interface \  
--service-name com.amazonaws.region.codeartifact.repositories --subnet-ids subnetid \  
--security-group-ids groupid --private-dns-enabled
```

Note

Lorsque vous créez un `codeartifact.repositories` point de terminaison, vous devez créer un nom d'hôte DNS privé à l'aide de l'`--private-dns-enabled` option. Si vous ne pouvez pas ou ne souhaitez pas créer de nom d'hôte DNS privé lorsque vous créez le `codeartifact.repositories` point de terminaison, vous devez suivre une étape de configuration supplémentaire pour utiliser votre gestionnaire de packages CodeArtifact depuis un VPC. Pour plus d'informations, consultez [Utiliser le `codeartifact.repositories` point de terminaison sans DNS privé](#).

Après avoir créé des points de terminaison VPC, vous devrez peut-être effectuer une configuration supplémentaire avec des règles de groupe de sécurité pour utiliser les points de terminaison. CodeArtifact Pour plus d'informations sur les groupes de sécurité dans Amazon VPC, consultez la section [Groupes de sécurité](#).

Si vous rencontrez des problèmes de connexion CodeArtifact, vous pouvez utiliser l'outil VPC Reachability Analyzer pour résoudre le problème. Pour plus d'informations, consultez [Qu'est-ce que VPC Reachability Analyzer ?](#)

Créer le point de terminaison de la passerelle Amazon S3

CodeArtifact utilise Amazon Simple Storage Service (Amazon S3) pour stocker les actifs du package. Pour extraire des packages CodeArtifact, vous devez créer un point de terminaison de passerelle pour Amazon S3. Lorsque votre processus de création ou de déploiement télécharge des packages depuis CodeArtifact, il doit y accéder CodeArtifact pour obtenir les métadonnées des packages et Amazon S3 pour télécharger les actifs des packages (par exemple, .jar les fichiers Maven).

Note

Un point de terminaison Amazon S3 n'est pas nécessaire lors de l'utilisation des formats de package Python ou Swift.

Pour créer le point de terminaison de la passerelle Amazon S3 pour CodeArtifact, utilisez la `EC2 create-vpc-endpoint` AWS CLI commande Amazon. Lorsque vous créez le point de terminaison, vous devez sélectionner les tables de routage pour votre VPC. Pour plus d'informations, consultez la section [Gateway VPC Endpoints](#) dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

La commande suivante crée un point de terminaison Amazon S3.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
--route-table-ids routetableid
```

Autorisations minimales de compartiment Amazon S3 pour AWS CodeArtifact

Le point de terminaison de la passerelle Amazon S3 utilise un document de politique IAM pour limiter l'accès au service. Pour n'autoriser que les autorisations minimales relatives au compartiment

Amazon S3 CodeArtifact, limitez l'accès au compartiment Amazon S3 CodeArtifact utilisé lorsque vous créez le document de politique IAM pour le point de terminaison.

Le tableau suivant décrit les compartiments Amazon S3 auxquels vous devez faire référence dans vos politiques pour autoriser l'accès CodeArtifact dans chaque région.

Région	ARN du compartiment Amazon S3
us-east-1	arn:aws:s3 ::assets-193858265520-us-east-1
us-east-2	arn:aws:s3 ::assets-250872398865-us-east-2
us-west-2	arn:aws:s3 ::assets-787052242323-us-west-2
eu-west-1	arn:aws:s3 ::assets-438097961670-eu-west-1
eu-west-2	arn:aws:s3 ::assets-247805302724-eu-west-2
eu-west-3	arn:aws:s3 ::assets-762466490029-eu-west-3
eu-north-1	arn:aws:s3 ::assets-611884512288-eu-north-1
eu-south-1	arn:aws:s3 ::assets-484130244270-eu-south-1
eu-central-1	arn:aws:s3 ::assets-769407342218-eu-central-1
ap-northeast-1	arn:aws:s3 ::assets-660291247815-ap-northeast-1
ap-southeast-1	arn:aws:s3 ::assets-421485864821-ap-southeast-1
ap-southeast-2	arn:aws:s3 ::assets-860415559748-ap-southeast-2
ap-south-1	arn:aws:s3 ::assets-681137435769-ap-south-1

Vous pouvez utiliser la `aws codeartifact describe-domain` commande pour récupérer le compartiment Amazon S3 utilisé par un CodeArtifact domaine.

```
aws codeartifact describe-domain --domain mydomain
```

```
{  
  "domain": {  
    "name": "mydomain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/mydomain",  
    "status": "Active",  
    "createdTime": 1583075193.861,  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/a73que8sq-ba...",  
    "repositoryCount": 13,  
    "assetSizeBytes": 513830295,  
    "s3BucketArn": "arn:aws:s3:::assets-787052242323-us-west-2"  
  }  
}
```

exemple

L'exemple suivant montre comment fournir l'accès aux compartiments Amazon S3 requis pour les CodeArtifact opérations dans la `us-east-1` région. Pour les autres régions, mettez à jour l'Resource entrée avec l'ARN d'autorisation correct pour votre région, conformément au tableau ci-dessus.

```
{  
  "Statement": [  
    {  
      "Sid": "Access-to-specific-bucket-only",  
      "Principal": "*",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Effect": "Allow",  
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]  
    }  
  ]  
}
```

Utilisation CodeArtifact depuis un VPC

Si vous ne pouvez pas ou ne souhaitez pas activer le DNS privé sur le point de terminaison du `com.amazonaws.region.codeartifact.repositories` VPC dans lequel vous l'avez créé [Créez des points de terminaison VPC pour CodeArtifact](#), vous devez utiliser une configuration différente de celle du point de terminaison des référentiels par rapport à CodeArtifact un VPC. Suivez les instructions [Utiliser le `codeartifact.repositories` point de terminaison sans DNS privé](#) pour configurer CodeArtifact si le DNS privé n'est pas activé sur le point de `com.amazonaws.region.codeartifact.repositories` terminaison.

Utiliser le `codeartifact.repositories` point de terminaison sans DNS privé

Si vous ne pouvez pas ou ne souhaitez pas activer le DNS privé sur le point de terminaison `com.amazonaws.region.codeartifact.repositories` VPC que vous avez créé [Créez des points de terminaison VPC pour CodeArtifact](#), vous devez suivre ces instructions pour configurer votre gestionnaire de packages avec l'URL correcte CodeArtifact .

1. Exécutez la commande suivante pour trouver un point de terminaison VPC à utiliser pour remplacer le nom d'hôte.

```
$ aws ec2 describe-vpc-endpoints --filters Name=service-name,Values=com.amazonaws.region.codeartifact.repositories \
--query 'VpcEndpoints[*].DnsEntries[*].DnsName'
```

Le résultat se présente comme suit.

```
[  
 [  
   "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"  
 ]  
 ]
```

2. Mettez à jour le chemin du point de terminaison VPC pour inclure le format du package, votre nom de CodeArtifact domaine et le nom du CodeArtifact référentiel. Consultez l'exemple suivant.

```
https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com/format/d/domain\_name-domain\_owner/repo\_name
```

Remplacez les champs suivants à partir de l'exemple de point de terminaison.

- ***format***: remplacez par un format de CodeArtifact package valide, par exemple, npm ou pypi.
- ***domain_name***: remplacez par le CodeArtifact domaine qui contient le CodeArtifact référentiel hébergeant vos packages.
- ***domain_owner***: Remplacez par l'ID du propriétaire du CodeArtifact domaine, par exemple 111122223333.
- ***repo_name***: remplacez par le CodeArtifact référentiel qui héberge vos packages.

L'URL suivante est un exemple de point de terminaison du référentiel npm.

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-west-2.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. Configurez votre gestionnaire de packages pour utiliser le point de terminaison VPC mis à jour à l'étape précédente. Vous devez configurer le gestionnaire de packages sans utiliser la CodeArtifact login commande. Pour les instructions de configuration pour chaque format de package, consultez la documentation suivante.

- npm : [Configuration de npm sans utiliser la commande de connexion](#)
- nuget : [configurer nuget ou dotnet sans la commande de connexion](#)
- pépin : [Configurer pip sans la commande de connexion](#)
- ficelle : [Configurer et utiliser Twine avec CodeArtifact](#)
- Gradle : [Utiliser CodeArtifact avec Gradle](#)
- MVN : [Utiliser CodeArtifact avec MVN](#)

Création d'une stratégie de point de terminaison de VPC pour CodeArtifact

Pour créer une politique de point de terminaison VPC pour CodeArtifact, spécifiez les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.

- Les ressources qui peuvent avoir des actions exécutées sur elles.

L'exemple de politique suivant indique que les principaux du compte 123456789012 peuvent appeler l'GetAuthorizationTokenAPI et récupérer des packages depuis un référentiel. CodeArtifact

```
{  
  "Statement": [  
    {  
      "Action": [  
        "codeartifact:GetAuthorizationToken",  
        "codeartifact:GetRepositoryEndpoint",  
        "codeartifact:ReadFromRepository",  
        "sts:GetServiceBearerToken"  
      ],  
      "Effect": "Allow",  
      "Resource": "*",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      }  
    }  
  ]  
}
```

Création de CodeArtifact ressources avec AWS CloudFormation

CodeArtifact est intégré à AWS CloudFormation un service qui vous aide à modéliser et à configurer vos AWS ressources afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. Vous créez un modèle qui décrit toutes les AWS ressources que vous souhaitez, et vous CloudFormation occupez de leur provisionnement et de leur configuration.

Lorsque vous l'utilisez CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos CodeArtifact ressources de manière cohérente et répétée. Décrivez simplement vos ressources une seule fois, puis fournissez les mêmes ressources à plusieurs reprises dans plusieurs comptes et AWS régions.

CodeArtifact et CloudFormation modèles

Pour fournir et configurer des ressources CodeArtifact et des services associés, vous devez comprendre les [CloudFormation modèles](#). Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez mettre à disposition dans vos CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser CloudFormation Designer pour vous aider à démarrer avec les CloudFormation modèles. Pour plus d'informations, consultez [Qu'est-ce qu'AWS CloudFormation Designer ?](#) dans le guide de AWS CloudFormation l'utilisateur.

CodeArtifact prend en charge la création de domaines, de référentiels et de groupes de packages dans CloudFormation. Pour plus d'informations, notamment des exemples de modèles JSON et YAML, consultez les rubriques suivantes du Guide de l'CloudFormation utilisateur :

- [AWS::CodeArtifact::Domain](#)
- [AWS::CodeArtifact::Repository](#)
- [AWS::CodeArtifact::PackageGroup](#)

Empêcher la suppression de CodeArtifact ressources

CodeArtifact les référentiels contiennent des dépendances applicatives critiques qui peuvent être difficiles à recréer en cas de perte. Pour protéger les CodeArtifact ressources contre toute

suppression accidentelle lors de la gestion CodeArtifact des ressources CloudFormation, incluez les `UpdateRetainPolicy` attributs `DeletionPolicy` et avec une valeur de `Retain` sur tous les domaines et référentiels. Cela empêchera la suppression si la ressource est supprimée du modèle de pile ou si la pile entière est supprimée accidentellement. L'extrait de code YAML suivant montre un domaine et un référentiel de base dotés des attributs suivants :

```
Resources:
  MyCodeArtifactDomain:
    Type: 'AWS::CodeArtifact::Domain'
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      DomainName: "my-domain"

  MyCodeArtifactRepository:
    Type: 'AWS::CodeArtifact::Repository'
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      RepositoryName: "my-repo"
      DomainName: !GetAtt MyCodeArtifactDomain.Name
```

Pour plus d'informations sur ces attributs, consultez [DeletionPolicy](#) et [UpdateReplacePolicy](#) dans le Guide de AWS CloudFormation l'utilisateur.

En savoir plus sur CloudFormation

Pour en savoir plus CloudFormation, consultez les ressources suivantes :

- [AWS CloudFormation](#)
- [AWS CloudFormation Guide de l'utilisateur](#)
- [AWS CloudFormation Guide de l'utilisateur de l'interface de ligne de commande](#)

Résolution des problèmes AWS CodeArtifact

Les informations suivantes peuvent vous aider à résoudre les problèmes courants liés à CodeArtifact.

Pour plus d'informations sur la résolution des problèmes spécifiques aux formats, consultez les rubriques suivantes :

- [Résolution des problèmes liés à Maven](#)
- [Résolution rapide des problèmes](#)

Je ne peux pas voir les notifications

Problème : lorsque vous êtes dans la console Outils pour développeurs et que vous choisissez Notifications sous Settings (Paramètres), une erreur d'autorisation s'affiche.

Correctifs possibles : Bien que les notifications soient une fonctionnalité de la console Developer Tools, les notifications CodeArtifact ne sont actuellement pas prises en charge. Aucune des politiques gérées n' CodeArtifact inclut les autorisations permettant aux utilisateurs de consulter ou de gérer les notifications. Si vous utilisez d'autres services dans la console Developer Tools et que ces services prennent en charge les notifications, les politiques gérées pour ces services incluent les autorisations requises pour consulter et gérer les notifications relatives à ces services.

Balisage des ressources

Une balise est une étiquette d'attribut personnalisée que vous attribuez ou AWS assignez à une AWS ressource. Chaque AWS étiquette comporte deux parties :

- Une clé de balise (par exemple, `CostCenter`, `Environment`, `Project` ou `Secret`). Les clés de balises sont sensibles à la casse.
- Un champ facultatif appelé valeur de balise (par exemple, `111122223333`, `Production` ou le nom d'une équipe). Omettre la valeur de balise équivaut à l'utilisation d'une chaîne vide. Les valeurs de balise sont sensibles à la casse, tout comme les clés de balise.

Ensemble, ces éléments sont connus sous le nom de paires clé-valeur.

Les balises vous aident à identifier et à organiser vos AWS ressources. De nombreux services AWS prennent en charge le balisage. Vous pouvez donc attribuer la même balise à des ressources à partir de différents services pour indiquer que les ressources sont liées. Par exemple, vous pouvez attribuer la même balise à un référentiel que celle que vous attribuez à un AWS CodeBuild projet.

Pour obtenir des conseils et des bonnes pratiques concernant l'utilisation des balises, consultez le livre blanc sur les [meilleures pratiques en matière de balisage AWS des ressources](#).

Vous pouvez baliser les types de ressources suivants dans CodeArtifact :

- [Marquer un dépôt dans CodeArtifact](#)
- [Marquer un domaine dans CodeArtifact](#)

Vous pouvez utiliser la console AWS CLI, CodeArtifact APIs, ou AWS SDKs pour :

- Ajoutez des balises à un domaine ou à un référentiel lorsque vous le créez*.
- Ajoutez, gérez et supprimez des balises pour un domaine ou un référentiel.

* Vous ne pouvez pas ajouter de balises à un domaine ou à un référentiel lorsque vous le créez dans la console.

Outre l'identification, l'organisation et le suivi de votre ressource à l'aide de balises, vous pouvez utiliser des balises dans les politiques IAM pour contrôler qui peut consulter votre ressource et

interagir avec elle. Pour obtenir des exemples de stratégies d'accès basées sur les balises, consultez [Utilisation de balises pour contrôler l'accès aux ressources CodeArtifact](#).

CodeArtifact répartition des coûts avec balises

Vous pouvez utiliser des balises pour répartir à la fois les coûts de stockage et de demande CodeArtifact.

Répartition des coûts de stockage des données dans CodeArtifact

Les coûts de stockage des données sont liés aux domaines. Par conséquent, pour répartir vos coûts de CodeArtifact stockage, vous pouvez utiliser toutes les balises appliquées à vos domaines. Pour plus d'informations sur l'ajout de balises aux domaines, consultez[Marquer un domaine dans CodeArtifact](#).

Répartition des coûts de demande dans CodeArtifact

L'utilisation de la plupart des requêtes est liée aux référentiels. Par conséquent, pour répartir les coûts de vos CodeArtifact demandes, vous pouvez utiliser toutes les balises appliquées à vos référentiels. Pour plus d'informations sur l'ajout de balises aux référentiels, consultez[Marquer un dépôt dans CodeArtifact](#).

Certains types de demandes sont associés à des domaines plutôt qu'à des référentiels, de sorte que l'utilisation des demandes et les coûts liés aux demandes seront alloués aux balises du domaine. Le meilleur moyen de déterminer si un type de demande est associé à un domaine ou à un référentiel est d'utiliser les [actions définies par](#) le AWS CodeArtifact tableau dans la référence d'autorisation de service. Recherchez le type de demande dans la colonne Actions et examinez la valeur dans la colonne Types de ressources correspondante. Si le type de ressource est un domaine, les demandes de ce type seront facturées au domaine. Si le type de ressource est un référentiel ou un package, les demandes de ce type seront facturées au référentiel. Certaines actions affichent les deux types de ressources. Pour ces actions, la ressource facturée dépend de la valeur transmise dans la demande.

Quotas dans AWS CodeArtifact

Le tableau suivant décrit les quotas de ressources dans CodeArtifact. Pour consulter les quotas de ressources ainsi que la liste des points de terminaison de service pour CodeArtifact, consultez les [quotas AWS de service](#) dans le Référence générale d'Amazon Web Services.

Vous pouvez [demander une augmentation des quotas de service](#) pour les quotas de CodeArtifact ressources suivants. Pour plus d'informations sur la demande d'augmentation des quotas de service, consultez la section [Quotas AWS de service](#).

Name	Par défaut	Ajustable	Description
Taille du fichier d'actifs	Chaque Région prise en charge : 5 giga-octets	Oui	Taille de fichier maximale par ressource.
Ressources par version de package	Chaque Région prise en charge : 150	Non	Le nombre maximum de ressources par version de package.
CopyPackageVersions demandes par seconde	Chaque Région prise en charge : 5	Oui	Le nombre maximum d'appels pouvant être effectués CopyPackageVersions par seconde.
Streams directs en amont par référentiel	Chaque Région prise en charge : 10	Non	Le nombre maximum de référentiels en amont direct par référentiel.
Domaines par AWS compte	Par région prise en charge : 10	Oui	Le nombre maximum de domaines pouvant être créés par AWS compte.
GetAuthorizationToken demandes par seconde	Chaque Région prise en charge : 40	Oui	Le nombre maximum de jetons d'autorisation récupérés par seconde.

Name	Par défaut	Ajustable	Description
GetPackageVersionAsset demandes par seconde	Chaque Région prise en charge : 50	Oui	Le nombre maximum d'appels pouvant être effectués GetPackageVersionAsset par seconde.
ListPackageVersionAssets demandes par seconde	Chaque région prise en charge : 200	Oui	Le nombre maximum d'appels pouvant être effectués ListPackageVersionAssets par seconde.
ListPackageVersions demandes par seconde	Chaque région prise en charge : 200	Oui	Le nombre maximum d'appels pouvant être effectués ListPackageVersions par seconde.
ListPackages demandes par seconde	Chaque région prise en charge : 200	Oui	Le nombre maximum d'appels pouvant être effectués ListPackages par seconde.
PublishPackageVersion demandes par seconde	Par région prise en charge : 10	Oui	Le nombre maximum d'appels pouvant être effectués PublishPackageVersion par seconde.
Requêtes de lecture par seconde à partir d'un seul AWS compte	Chaque région prise en charge : 800	Oui	Le nombre maximum de demandes de lecture provenant d'un AWS compte par seconde.
Référentiels par domaine	Chaque Région prise en charge : 1 000	Oui	Le nombre maximum de référentiels pouvant être créés par domaine.

Name	Par défaut	Ajustable	Description
Demandes par seconde à l'aide d'un seul jeton d'authentification	Chaque région prise en charge : 1 200	Non	Le nombre maximal de demandes par seconde utilisant un seul jeton d'authentification.
Demandes sans jeton d'authentification par adresse IP	Chaque région prise en charge : 600	Non	Le nombre maximal de demandes par seconde sans jeton d'authentification provenant d'une seule adresse IP.
Référentiels en amont recherchés	Chaque région prise en charge : 25	Non	Nombre maximal de référentiels en amont recherchés lors de la résolution d'un package.
Rédiger des demandes par seconde à partir d'un seul AWS compte	Chaque région prise en charge : 100	Oui	Le nombre maximum de demandes d'écriture provenant d'un AWS compte par seconde.

Note

En général, chaque demande de lecture est considérée comme une demande comptabilisée par rapport à un quota. CodeArtifact Cependant, pour le format de package Ruby, une seule demande de lecture adressée à l'`/api/v1/dependencies` opération peut demander des données sur plusieurs packages.

Par exemple, la demande peut ressembler à `https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1, gem2, gem3`. Dans cet exemple, la demande compte pour trois demandes par rapport au quota.

Notez que les demandes multiples ne s'appliquent qu'aux quotas de service, et non à la facturation. Dans l'exemple, vous ne serez facturé que pour une seule demande, bien que trois demandes soient prises en compte dans le quota de service.

AWS CodeArtifact historique du document du guide de l'utilisateur

Le tableau suivant décrit les modifications importantes apportées à la documentation de CodeArtifact.

Modification	Description	Date
<u>Documentation ajoutée pour configurer et utiliser Cargo avec CodeArtifact</u>	CodeArtifact prend désormais en charge les caisses Cargo. Ajout de documentation avec des conseils sur la configuration de Cargo pour utiliser des CodeArtifact référentiels. Pour de plus amples informations, veuillez consulter <u>Utilisation CodeArtifact avec Cargo.</u>	20 juin 2024
<u>Documentation ajoutée pour configurer et utiliser Ruby avec CodeArtifact</u>	CodeArtifact supporte désormais les gemmes Ruby. Ajout de documentation avec des conseils sur la configuration des gestionnaires de packages Ruby pour utiliser CodeArtifact des référentiels. Pour de plus amples informations, veuillez consulter <u>Utilisation CodeArtifact avec Ruby.</u>	30 avril 2024
<u>Ajout d'un exemple de politique clé pour la création de domaines avec une AWS KMS clé gérée par le client</u>	Ajout d'un exemple de politique clé qui peut être utilisé pour créer une clé KMS gérée par le client pour chiffrer les actifs dans CodeArtifact les domaines. Pour de plus amples informations, veuillez	18 avril 2024

consulter [Exemple de politique AWS KMS clé.](#)

[Ajout de documentation pour soutenir le lancement de groupes de packages.](#)

Ajout de documentation sur la gestion et l'utilisation des groupes de packages dans CodeArtifact. Pour de plus amples informations, veuillez consulter [Utilisation de groupes de packages dans CodeArtifact.](#)

21 mars 2024

[Ajout de gestionnaires de packages valides supplémentaires à la documentation sur la commande de connexion aws codeartifact.](#)

Ajouté dotnetnuget, et swift à la liste des gestionnaires de packages valides à utiliser avec la aws codeartifact login commande. Pour de plus amples informations, veuillez consulter [AWS CodeArtifact authentification et jetons.](#)

18 février 2024

[Ajout d'une entrée à la documentation de dépannage de Swift concernant le blocage de Xcode sur les machines CI](#)

Ajout d'informations, y compris une solution, concernant un problème susceptible de bloquer Xcode sur les machines CI en raison de l'invite de saisie du mot de passe par le trousseau. Pour de plus amples informations, veuillez consulter [Xcode se bloque sur la machine CI en raison de l'invite de saisie du mot de passe par le trousseau.](#)

6 février 2024

<u>Informations ajoutées sur la résolution des problèmes de lenteur des temps d'installation des packages npm avec npm 8.x ou supérieur</u>	Ajout d'informations sur la manière de contourner la lenteur des temps d'installation des packages npm à partir de CodeArtifact, ce qui pourrait entraîner des délais de construction lents. Pour de plus amples informations, veuillez consulter <u>Résolution des problèmes d'installation lents avec npm 8.x ou supérieur</u> .	29 décembre 2023
<u>Informations mises à jour sur le comportement des actifs et des métadonnées du package Python dans CodeArtifact</u>	Informations mises à jour sur la manière dont CodeArtifact les référentiels conservent et actualisent les actifs et les métadonnées des versions des packages Python. Pour de plus amples informations, veuillez consulter <u>Demande de packages Python depuis des connexions en amont et externes</u> .	14 décembre 2023
<u>Documentation réorganisée sur la surveillance CodeArtifact</u>	Informations réorganisées sur les CodeArtifact événements de surveillance et ajout d'informations sur l'affichage des CodeArtifact demandes avec CloudWatch les métriques Amazon. Pour de plus amples informations, veuillez consulter <u>Surveillance CodeArtifact</u> .	14 décembre 2023

<u>Ajout d'informations supplémentaires sur la gestion CodeArtifact des ressources avec CloudFormation</u>	Ajout de références et de liens vers la documentation sur la gestion CodeArtifact des ressources avec CloudFormation, y compris une section sur la prévention de la suppression des CodeArtifact ressources gérées avec CloudFormation. Pour de plus amples informations, veuillez consulter <u>Empêcher la suppression de CodeArtifact ressources.</u>	7 décembre 2023
<u>Ajout d'une documentation CodeArtifact détaillant le support des magasins de clés AWS KMS externes (XKS)</u>	Ajout d'une section contenant des informations sur CodeArtifact la prise en charge des clés KMS, y compris l'utilisation des clés XKS avec CodeArtifact. Pour de plus amples informations, veuillez consulter <u>Types de AWS KMS clés pris en charge dans CodeArtifact.</u>	31 octobre 2023
<u>Documentation de dépannage existante mise à jour et ajout d'une nouvelle documentation</u>	Ajout d'une rubrique de résolution des problèmes liés à Maven et inclusion de liens vers la documentation de dépannage de Swift et Maven dans la rubrique générale de résolution des problèmes. Pour de plus amples informations, veuillez consulter <u>Résolution des problèmes AWS CodeArtifact.</u>	28 septembre 2023

[Documentation mise à jour pour inclure la commande de publication de Swift Package Manager](#)

Swift 5.9 a introduit une `swift package-registry publish` commande permettant de créer et de publier un package Swift dans un référentiel de packages. Mise à jour de la documentation Swift pour inclure les instructions d'utilisation de cette commande. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec Swift.](#)

25 septembre 2023

[Ajout de documentation pour la configuration CodeArtifact avec Swift](#)

CodeArtifact prend désormais en charge les packages Swift. Ajout de documentation avec des conseils sur la configuration de Swift pour utiliser des CodeArtifact référentiels. Pour de plus amples informations, veuillez consulter [Utilisation CodeArtifact avec Swift.](#)

20 septembre 2023

[Ajout de conseils sur la façon de CodeArtifact gérer les versions de packages Python supprimées](#)

Ajout d'une documentation contenant des informations sur la façon de savoir si une version de package Python est supprimée, comment CodeArtifact gérer les versions de package supprimées et les réponses aux questions courantes. Pour de plus amples informations, veuillez consulter [Versions du package supprimées.](#)

02/08/2023

<u>Correction d'une commande de ligne de commande incorrecte dans la documentation de Yarn</u>	Correction d'une commande de ligne de commande incorrecte qui récupérait un jeton CodeArtifact d'autorisation et le stockait dans une variable d'environnement dans la <u>documentation Yarn</u> .	20 juillet 2023
<u>Ajouts mineurs et corrections d'un petit bogue à la documentation Python</u>	Ajout d'informations sur les points pip et twine dans leur documentation respective et correction de ce qui se passe lors de l'utilisation de la <code>codeartifact login</code> commande avec de la ficelle. Pour plus d'informations, consultez <u>Configurer et utiliser pip avec CodeArtifact</u> et <u>Configurer et utiliser Twine avec CodeArtifact</u> .	14 juillet 2023
<u>Correction de commandes dotnet incorrectes dans la documentation CodeBuild</u>	Les <code>dotnet add package</code> commandes de la <u>Utilisation de NuGet packages dans CodeBuild</u> documentation ont été corrigées.	13 juillet 2023
<u>Mise à jour AWS CodeArtifact et AWS Identity and Access Management documentation</u>	Révision de l'IAM dans la CodeArtifact documentation afin d'ajouter de la clarté et de la cohérence à la documentation des autres services. AWS Consultez <u>Identity and Access Management pour AWS CodeArtifact</u> .	24 mai 2023

<u>Ajout d'informations sur les versions de packages Python supprimées</u>	Ajout d'informations sur la façon CodeArtifact dont les métadonnées des versions de packages Python supprimées sont conservées. Pour plus d'informations, consultez <u>Versions du package supprimées</u> .	11 avril 2023
<u>Informations supplémentaires sur le support de Clojure</u>	Ajout d'informations sur le support de Clojure, notamment la gestion des dépendances pour les projets Clojure. Pour de plus amples informations, veuillez consulter <u>À utiliser CodeArtifact avec deps.edn</u> .	21 mars 2023
<u>Informations supplémentaires sur la publication de packages génériques</u>	Ajout d'informations sur les packages génériques et sur la façon de publier et de télécharger le contenu des packages avec le AWS CLI. Pour plus d'informations, consultez <u>Utilisation CodeArtifact avec des packages génériques, Publication et consommation de packages génériques et Commandes prises en charge pour les packages génériques</u> .	10 mars 2023
<u>Informations supplémentaires sur les limites de taille des actifs pour la publication</u>	Ajout d'une section à la publication de packages pour expliquer les limites de taille des actifs pour la publication.	21 juin 2022

[Refactorisation de la documentation sur les connexions externes](#)

Déplacement de la documentation sur les connexions externes et réorganisée afin de se concentrer sur l'objectif final de l'utilisateur, qui est de connecter son CodeArtifact référentiel aux référentiels de packages publics. Nous avons également ajouté plus de conseils et d'informations sur les différentes méthodes pour atteindre cet objectif. Pour de plus amples informations, veuillez consulter [Connect un CodeArtifact dépôt à un dépôt public](#).

9 mai 2022

[Mise à jour des informations sur les CodeArtifact événements pour Amazon CloudWatch Events](#)

Ajout d'informations supplémentaires au account champ et ajout du repositoryAdministrator champ. Pour de plus amples informations, veuillez consulter [CodeArtifact format d'événement et exemple](#).

7 mars 2022

[Ajout d'instructions de configuration pour une utilisation CodeArtifact depuis un VPC sans DNS privé](#)

Si vous ne pouvez pas ou ne souhaitez pas activer le DNS privé sur le point de terminaison de votre `codeartifact/repos` VPC, vous devez utiliser une configuration différente de celle du point de terminaison des référentiels par rapport à CodeArtifact un VPC. Pour plus d'informations, consultez [Utiliser le `codeartifact/repos` point de terminaison sans DNS privé.](#)

[Ajout d'une documentation détaillée pour mettre à jour l'état des versions des packages](#)

La documentation sur l'état de la version du package de mise à jour a été étendue dans sa propre rubrique. Ajout de documentation pour mettre à jour le statut d'une version de package, y compris les autorisations IAM requises, des exemples de AWS CLI commandes pour différents scénarios et les erreurs possibles. Pour plus d'informations, consultez [État de version du package de mise à jour.](#)

8 février 2022

1er septembre 2021

[Mise à jour de la documentation des versions du package de copie avec des informations plus détaillées sur les autorisations](#)

Ajout d'informations supplémentaires sur les autorisations IAM et de politique basées sur les ressources requises pour appeler la `aws codeartifact copy-package-versions` commande pour copier des versions de packages d'un référentiel à un autre au sein du même domaine dans. CodeArtifact Outre des informations supplémentaires, il existe désormais des exemples de politiques basées sur les ressources requises pour le référentiel source et de destination. Pour plus d'informations, consultez [Autorisations IAM requises pour copier des packages](#).

25 août 2021

[Documentation mise à jour pour exécuter une version Gradle dans IntelliJ IDEA](#)

Mise à jour de la documentation pour exécuter une version de Gradle dans IntelliJ IDEA avec des étapes de configuration de Gradle pour récupérer des plugins. CodeArtifact Une option a également été ajoutée pour créer un nouveau jeton CodeArtifact d'autorisation pour chaque nouvelle exécution avec un appel en ligne àaws codeartifact get-authorization-token . Pour plus d'informations, consultez [Exécuter une compilation Gradle dans IntelliJ IDEA.](#)

23 août 2021

[Ajout de documentation pour configurer et utiliser Yarn avec AWS CodeArtifact](#)

Ajout de documentation pour configurer et utiliser Yarn 1.X et Yarn 2.X pour gérer les packages npm avec. CodeArtifact Pour plus d'informations, consultez [Configurer et utiliser Yarn avec CodeArtifact.](#)

30 juillet 2021

<u>AWS CodeArtifact prend désormais en charge les NuGet packages</u>	CodeArtifact les utilisateurs peuvent désormais publier et consommer NuGet des packages. Ajout de documentation pour la configuration et l'utilisation de Visual Studio et d'outils de ligne de NuGet commande tels que nuget et dotnet avec CodeArtifact les référentiels. Pour plus d'informations, consultez <u>Utilisation CodeArtifact avec NuGet</u> .	19 novembre 2020
<u>Marquage des ressources dans AWS CodeArtifact</u>	Ajout de documentation sur le balisage des référentiels et des domaines dans AWS CodeArtifact Consultez <u>Balisage des ressources</u> .	30 octobre 2020
<u>CodeArtifact prend désormais en charge CloudFormation</u>	CodeArtifact les utilisateurs peuvent désormais utiliser des CloudFormation modèles pour créer des CodeArtifact référentiels et des domaines. Consultez <u>Création de CodeArtifact ressources avec AWS CloudFormation</u> pour plus d'informations et pour commencer.	8 octobre 2020

[Ajoutez des informations sur la création de points de terminaison de passerelle Amazon S3 à utiliser CodeArtifact avec Amazon VPC](#)

Ajout d'informations sur la création de points de terminaison de passerelle Amazon S3 à l'aide de la EC2 AWS CLI commande Amazon.

12 août 2020

Cette documentation contient également des informations sur les autorisations spécifiques qui CodeArtifact doivent être utilisées avec les environnements Amazon VPC. Consultez [Créer le point de terminaison de la passerelle Amazon S3.](#)

[Publication d'artefacts Maven avec curl et publication d'artefacts Maven tiers](#)

Ajout de conseils pour [Publier avec curl etPublier des artefacts tiers.](#)

10 août 2020

[Version de disponibilité générale \(GA\)](#)

Version initiale du guide de CodeArtifact l'utilisateur.

10 juin 2020

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.