CodeArtifact Benutzerleitfaden

CodeArtifact

CodeArtifact: CodeArtifact Benutzerleitfaden

Table of Contents

Was ist AWS CodeArtifact? ... 1
Wie CodeArtifact funktioniert das? ... 1
Konzepte ... 2
Komponente ... 2
Domain ... 3
Repository ... 3
Paket ... 3
Paketgruppe ... 4
Paket-Namespace ... 4
Paketversion ... 4
Revision der Paketversion ... 4
Upstream-Repository ... 5
Wie fange ich an mit CodeArtifact? ... 5
Einrichtung ... 6
Melde dich an für AWS ... 6
Installieren oder aktualisieren Sie und konfigurieren Sie dann das AWS CLI ... 7
Stellen Sie einen IAM-Benutzer bereit ... 8
Installieren Sie Ihren Paketmanager oder Ihr Build-Tool ... 10
Nächste Schritte ... 10
Erste Schritte ... 12
Voraussetzungen ... 12
Erste Schritte mit der Konsole ... 13
Erste Schritte mit dem AWS CLI ... 15
Arbeiten mit Repositorien ... 23
Erstellen eines -Repositorys ... 23
Erstellen Sie ein Repository (Konsole) ... 24
Erstellen Sie ein Repository (AWS CLI) ... 25
Erstellen Sie ein Repository mit einem Upstream-Repository ... 26
Herstellen einer Verbindung mit einem Repository ... 27
Verwenden Sie einen Paketmanager-Client ... 28
Löschen Sie ein Repository ... 28
Löschen Sie ein Repository (Konsole) ... 28
Löschen Sie ein Repository (AWS CLI) ... 29
Schützt Repositorys vor dem Löschen ... 29

Repositorys auflisten ... 31
Listet die Repositorys in einem Konto auf AWS ... 31
Listet Repositorys in der Domäne auf ... 32
Eine Repository-Konfiguration anzeigen oder ändern ... 34
Eine Repository-Konfiguration anzeigen oder ändern (Konsole) ... 34
Eine Repository-Konfiguration anzeigen oder ändern ()AWS CLI ... 36
Repository-Richtlinien ... 38
Erstellen Sie eine Ressourcenrichtlinie, um Lesezugriff zu gewähren ... 38
Legen Sie eine Richtlinie fest ... 40
Lesen Sie eine Richtlinie ... 41
Löschen Sie eine Richtlinie ... 42
Gewähren Sie den Hauptbenutzern Lesezugriff ... 42
Gewähren Sie Schreibzugriff auf Pakete ... 42
Gewähren Sie Schreibzugriff auf ein Repository ... 44
Interaktion zwischen Repository- und Domain-Richtlinien ... 45
Kennzeichnen Sie ein Repository ... 46
Tag-Repositorien (CLI) ... 46
Tag-Repositorys (Konsole) ... 50
Arbeiten mit Upstream-Repositorys ... 54
Was ist der Unterschied zwischen Upstream-Repositorys und externen Verbindungen? ... 55
Upstream-Repositorys hinzufügen oder entfernen ... 55
Upstream-Repositorys hinzufügen oder entfernen (Konsole) ... 55
Fügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI ... 56
Ein CodeArtifact Repository mit einem öffentlichen Repository Connect ... 59
Stellen Sie eine Connect zu einem externen Repository her (Konsole) ... 60
Stellen Sie eine Connect zu einem externen Repository her (CLI) ... 61
Unterstützte Repositorys für externe Verbindungen ... 62
Eine externe Verbindung entfernen (CLI) ... 63
Eine Paketversion mit Upstream-Repositorys anfordern ... 64
Aufbewahrung von Paketen aus Upstream-Repositorys ... 64
Ruft Pakete über eine Upstream-Beziehung ab ... 65
Aufbewahrung von Paketen in Zwischenrepositorien ... 67
Pakete von externen Verbindungen anfordern ... 68
Ruft Pakete von einer externen Verbindung ab ... 68
Latenz bei externen Verbindungen ... 70
CodeArtifact Verhalten, wenn ein externes Repository nicht verfügbar ist ... 71

Verfügbarkeit neuer Paketversionen ... 71
Paketversionen mit mehr als einem Asset werden importiert ... 72
Reihenfolge der Prioritäten des Upstream-Reposit ... 72
Einfaches Beispiel für die Prioritätsreihenfolge ... 73
Beispiel für eine komplexe Prioritätsreihenfolge ... 74
API-Verhalten bei Upstream-Repositorys ... 75
Mit -Paketen arbeiten ... 78
Überblick über Pakete ... 78
Unterstützte Paketformate ... 79
Veröffentlichen von Paketen ... 79
Status der Paketversion ... 82
Normalisierung von Paketnamen, Paketversion und Assetnamen ... 83
Listet Paketnamen auf ... 84
NPM-Paketnamen auflisten ... 85
Listet die Namen der Maven-Pakete auf ... 87
Python-Paketnamen auflisten ... 88
Filtern Sie nach dem Präfix des Paketnamens ... 88
Unterstützte Kombinationen von Suchoptionen ... 89
Ausgabe formatieren ... 89
Standardwerte und andere Optionen ... 90
Paketversionen auflisten ... 90
NPM-Paketversionen auflisten ... 92
Maven-Paketversionen auflisten ... 93
Versionen sortieren ... 93
Standardanzeigeversion ... 94
Ausgabe formatieren ... 94
Listet die Ressourcen der Paketversion auf ... 95
Listet die Assets eines NPM-Pakets auf ... 96
Listet die Ressourcen eines Maven-Pakets auf ... 97
Laden Sie die Ressourcen der Paketversion herunter ... 97
Pakete zwischen Repositorys kopieren ... 98
Erforderliche IAM-Berechtigungen zum Kopieren von Paketen ... 98
Kopieren Sie die Paketversionen ... 100
Kopiert ein Paket aus Upstream-Repositorys ... 101
Kopieren Sie ein NPM-Paket mit Gültigkeitsbereich ... 101
Kopieren Sie die Maven-Paketversionen ... 101

Versionen, die im Quell-Repository nicht existieren ... 102
Versionen, die bereits im Ziel-Repository vorhanden sind ... 103
Angabe einer Paketversionsrevision ... 104
Kopiere npm-Pakete ... 105
Löschen Sie ein Paket oder eine Paketversion ... 106
Löschen eines Pakets (AWS CLI) ... 106
Löschen eines Pakets (Konsole) ... 107
Löschen einer Paketversion (AWS CLI) ... 107
Löschen einer Paketversion (Konsole) ... 108
Löschen eines NPM-Pakets oder einer Paketversion ... 109
Löschen eines Maven-Pakets oder einer Paketversion ... 109
Bewährte Methoden zum Löschen von Paketen oder Paketversionen ... 110
Details und Abhängigkeiten der Paketversion anzeigen und aktualisieren ... 110
Details zur Paketversion anzeigen ... 110
Versionsdetails des npm-Pakets anzeigen ... 112
Versionsdetails des Maven-Pakets anzeigen ... 113
Abhängigkeiten von Paketversionen anzeigen ... 114
Readme-Datei zur Paketversion anzeigen ... 115
Aktualisiere den Status der Paketversion ... 115
Der Status der Paketversion wird aktualisiert ... 116
Erforderliche IAM-Berechtigungen, um den Status einer Paketversion zu aktualisieren ... 117
Der Status eines NPM-Pakets mit Gültigkeitsbereich wird aktualisiert ... 118
Status für ein Maven-Paket wird aktualisiert ... 118
Angabe einer Paketversionsrevision ... 118
Unter Verwendung des Parameters „Erwarteter Status“ ... 120
Fehler bei einzelnen Paketversionen ... 120
Entsorgung von Paketversionen ... 122
Die Einstellungen zur Herkunft des Pakets werden bearbeitet ... 124
Allgemeine Szenarien zur Paketzugriffskontrolle ... 124
Einstellungen zur Kontrolle des Paketursprungs ... 126
Standardeinstellungen für die Kontrolle des Paketursprungs ... 127
Wie die Kontrollen zur Herkunft von Paketen mit den Ursprungskontrollen für Paketgruppen interagieren ... 128
Die Einstellungen für den Paketursprung werden bearbeitet ... 129
Veröffentlichungs- und Upstream-Repositorys ... 131
Arbeiten mit Paketgruppen ... 132

Erstellen Sie eine Paketgruppe 133
Erstellen Sie eine Paketgruppe (Konsole) 133
Erstellen Sie eine Paketgruppe (AWS CLI) 134
Eine Paketgruppe anzeigen oder bearbeiten 135
Eine Paketgruppe anzeigen oder bearbeiten (Konsole) 135
Eine Paketgruppe anzeigen oder bearbeiten ()AWS CLI 136
Löschen Sie eine Paketgruppe 137
Löschen Sie eine Paketgruppe (Konsole) 138
Löschen Sie eine Paketgruppe (AWS CLI) 138
Herkunftskontrollen für Paketgruppen 138
Einstellungen für Einschränkungen 139
Listen zulässiger Repositorys 140
Bearbeiten der Origin Control-Einstellungen für Paketgruppen 141
Konfigurationsbeispiele für die Ursprungssteuerung von Paketgruppen 142
Wie die Einstellungen für die Ursprungssteuerung von Paketgruppen mit den Einstellungen für die Paketursprungskontrolle interagieren 145
Syntax und Abgleichverhalten der Paketgruppendefinition 145
Syntax und Beispiele für Paketgruppendefinitionen 145
Paketgruppenhierarchie und Musterspezifität 147
Wörter, Wortgrenzen und Präfixabgleich 147
Groß-/Kleinschreibung 148
Starkes und schwaches Spiel 149
Zusätzliche Varianten 150
Kennzeichnen Sie eine Paketgruppe 151
Tag-Paketgruppen (CLI) 151
Arbeiten mit Domänen 155
Überblick über die Domäne 155
Kontenübergreifende Domänen 156
Arten von AWS KMS Schlüsseln, die unterstützt werden in CodeArtifact 157
Domain erstellen 158
Erstellen Sie eine Domäne (Konsole) 158
Erstellen Sie eine Domäne (AWS CLI) 159
Beispiel für eine AWS KMS Schlüsselrichtlinie 160
Domäne löschen 162
Einschränkungen beim Löschen von Domains 162
Eine Domain löschen (Konsole) 163

Löschen Sie eine Domäne (AWS CLI) 163
Domain-Richtlinien 163
Aktivieren Sie den kontoübergreifenden Zugriff auf eine Domain 164
Beispiel für eine Domänenrichtlinie 166
Beispiel für eine Domänenrichtlinie mit AWS Organizations 167
Legen Sie eine Domain-Richtlinie fest 168
Lesen Sie eine Domänenrichtlinie 169
Löschen Sie eine Domänenrichtlinie 169
Kennzeichnen Sie eine Domain 170
Tag-Domains (CLI) 170
Domains taggen (Konsole) 174
Cargo verwenden 177
Cargo konfigurieren und verwenden 177
Konfigurieren Sie Cargo mit CodeArtifact 177
Installation von Frachtkisten 182
Frachtkisten veröffentlichen 183
Unterstützung von Cargo Command 183
Unterstützte Befehle, für die der Zugriff auf die Registrierung erforderlich ist 184
Befehle werden nicht unterstützt 184
Mit Maven 185
CodeArtifact Mit Gradle verwenden 185
Abhängigkeiten abrufen 186
Plugins abrufen 187
Veröffentlichen Sie Artefakte 188
Führen Sie einen Gradle-Build in IntelliJ IDEA aus 190
CodeArtifact Mit MVN verwenden 194
Abhängigkeiten abrufen 186
Artefakte veröffentlichen 188
Veröffentlichen Sie Artefakte von Drittanbietern 199
Beschränken Sie das Herunterladen von Maven-Abhängigkeiten auf ein Repository CodeArtifact 200
Informationen zum Apache Maven-Projekt 202
CodeArtifact Mit deps.edn verwenden 202
Abhängigkeiten abrufen 203
Artefakte veröffentlichen 204
Publizieren mit curl 205

Verwenden Sie Maven-Prüfsummen ... 207
Prüfsummenspeicher ... 208
Bei der Veröffentlichung stimmen die Prüfsummen nicht überein ... 209
Wiederherstellung nach nicht übereinstimmenden Prüfsummen ... 210
Verwenden Sie Maven-Snapshots ... 211
Veröffentlichung von Snapshots in CodeArtifact ... 211
Snapshot-Versionen werden konsumiert ... 214
Löschen von Snapshot-Versionen ... 214
Veröffentlichung von Snapshots mit Curl ... 214
Schnappschüsse und externe Verbindungen ... 217
Snapshots und Upstream-Repositorys ... 217
Maven-Pakete von Upstreams und externen Verbindungen anfordern ... 218
Import von Standard-Asset-Namen ... 218
Importieren von nicht standardmäßigen Asset-Namen ... 219
Die Herkunft von Assets wird überprüft ... 220
Import neuer Assets und Paketversionsstatus in Upstream-Repositorys ... 220
Fehlerbehebung in Maven ... 221
Deaktivieren Sie parallel Eingaben, um Fehler 429 zu beheben: Zu viele Anfragen ... 221
Verwenden von npm ... 222
Konfigurieren und verwenden Sie npm ... 222
Konfiguration von npm mit dem Login-Befehl ... 223
Konfiguration von npm ohne Verwendung des Login-Befehls ... 223
NPM-Befehle ausführen ... 226
Überprüfung der NPM-Authentifizierung und -Autorisierung ... 226
Zurück zur Standard-NPM-Registrierung ... 227
Fehlerbehebung bei langsamen Installationen mit npm 8.x oder höher ... 227
Konfiguriere und verwende Yarn ... 228
Konfigurieren Sie Yarn 1.X mit dem Befehl `aws codeartifact login` ... 228
Konfiguriere Yarn 2.X mit dem Befehl `yarn config set` ... 230
Unterstützung für npm-Befehle ... 232
Unterstützte Befehle, die mit einem Repository interagieren ... 232
Unterstützte clientseitige Befehle ... 234
Befehle werden nicht unterstützt ... 184
Handhabung von NPM-Tags ... 238
Bearbeiten Sie Tags mit dem NPM-Client ... 238
npm-Tags und die API CopyPackageVersions ... 239

npm-Tags und Upstream-Repositorys ... 239
Support für npm-kompatible Paketmanager ... 241
Benutzen NuGet ... 242
CodeArtifact Mit Visual Studio verwenden ... 242
Konfigurieren Sie Visual Studio mit dem CodeArtifact Credential Provider ... 243
Verwenden Sie die Visual Studio Package Manager-Konsole ... 244
CodeArtifact Mit Nuget oder Dotnet verwenden ... 245
Konfigurieren Sie die Nuget- oder Dotnet-CLI ... 245
NuGet Pakete konsumieren ... 251
NuGet Pakete veröffentlichen ... 252
CodeArtifact NuGet Referenz zum Credential Provider ... 253
CodeArtifact NuGet Versionen des Credential Providers ... 254
NuGet Normalisierung von Paketnamen, Version und Assetnamen ... 254
NuGet Kompatibilität ... 256
Allgemeine NuGet Kompatibilität ... 256
NuGet Unterstützung für die Befehlszeile ... 256
Verwenden von Python ... 257
Konfigurieren und verwenden Sie Pip mit CodeArtifact ... 257
Konfigurieren Sie pip mit dem Befehl `login` ... 257
Konfigurieren Sie pip ohne den Login-Befehl ... 258
Führen Sie pip aus ... 259
Konfigurieren und verwenden Sie Twine mit CodeArtifact ... 260
Konfigurieren Sie Twine mit dem Befehl `login` ... 260
Konfigurieren Sie Twine ohne den Befehl `login` ... 261
Führen Sie Twine aus ... 262
Normalisierung von Python-Paketnamen ... 262
Python-Kompatibilität ... 263
Unterstützung für Pip-Befehle ... 263
Python-Pakete von Upstreams und externen Verbindungen anfordern ... 264
Paketversionen wurden entfernt ... 265
Warum wird das Abrufen der neuesten Yankeed-Metadaten oder Assets für eine Paketversion CodeArtifact nicht unterstützt? ... 266
Ruby verwenden ... 268
und Bundler konfigurieren RubyGems und verwenden ... 268
Configure RubyGems (`gem`) und Bundler (`bundle`) mit CodeArtifact ... 268
Installation von Ruby Gems ... 274

Veröffentlichung von Ruby-Edelsteinen ... 275
RubyGems Befehlsunterstützung ... 276
Bundler-Kompatibilität ... 277
Bundler-Kompatibilität ... 277
Swift verwenden ... 278
Konfigurieren Sie Swift mit CodeArtifact ... 278
Konfigurieren Sie Swift mit dem Login-Befehl ... 278
Konfigurieren Sie Swift ohne den Login-Befehl ... 280
Swift-Pakete konsumieren und veröffentlichen ... 284
Swift-Pakete werden konsumiert ... 284
Swift-Pakete in Xcode konsumieren ... 286
Swift-Pakete veröffentlichen ... 287
Swift-Pakete abrufen von GitHub und erneutes Veröffentlichen von CodeArtifact ... 289
Schnelle Normalisierung von Paketnamen und Namespace ... 292
Schnelle Problembehebung ... 292
Ich erhalte einen 401-Fehler in Xcode, auch nachdem ich den Swift Package Manager konfiguriert habe ... 292
Xcode hängt auf dem CI-Computer, da der Schlüsselbund zur Eingabe des Passworts aufgefordert wird ... 293
Verwendung generischer Pakete ... 296
Überblick über generische Pakete ... 296
Allgemeine Paketeinschränkungen ... 296
Unterstützte Befehle ... 297
Generische Pakete veröffentlichen und konsumieren ... 298
Veröffentlichen eines generischen Pakets ... 298
Generische Paketressourcen auflisten ... 300
Generische Paket-Assets werden heruntergeladen ... 302
Verwenden CodeArtifact mit CodeBuild ... 304
Verwenden von NPM-Paketen in CodeBuild ... 304
Richten Sie Berechtigungen mit IAM-Rollen ein ... 304
Melden Sie sich an und verwenden Sie npm ... 305
Verwenden von Python-Paketen in CodeBuild ... 306
Richten Sie Berechtigungen mit IAM-Rollen ein ... 306
Melden Sie sich an und verwenden Sie Pip oder Twine ... 308
Verwenden von Maven-Paketen in CodeBuild ... 309
Richten Sie Berechtigungen mit IAM-Rollen ein ... 309

Verwenden Sie Gradle oder MVN 311
Verwenden von Paketen NuGet in CodeBuild 311
Richten Sie Berechtigungen mit IAM-Rollen ein 312
Verbrauchen Sie Pakete NuGet 313
Mit NuGet Paketen bauen 314
Veröffentlichen Sie NuGet Pakete 317
Zwischenspeichern von Abhängigkeiten 318
Überwachung CodeArtifact 320
CodeArtifact Ereignisse überwachen 320
CodeArtifact Veranstaltungsformat und Beispiel 322
Verwenden Sie ein Ereignis, um eine CodePipeline Ausführung zu starten 326
EventBridgeBerechtigungen konfigurieren 327
Erstellen Sie die EventBridge Regel 327
Erstellen Sie das EventBridge Regelziel 327
Verwenden Sie ein Ereignis, um eine Lambda-Funktion auszuführen 328
Erstellen Sie die EventBridge Regel 328
Erstellen Sie das EventBridge Regelziel 328
Berechtigungen konfigurieren EventBridge 329
Sicherheit 330
Datenschutz 331
Datenverschlüsselung 332
Datenschutz für Datenverkehr 332
Überwachen 333
Protokollierung von CodeArtifact API-Aufrufen mit AWS CloudTrail 333
Compliance-Validierung 337
Authentifizierung und Tokens 339
Mit dem Befehl erstellte Tokens `login` 340
Zum Aufrufen der `GetAuthorizationToken` API sind Berechtigungen erforderlich 342
Mit der `GetAuthorizationToken` API erstellte Tokens 342
Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen 343
Widerrufen von CodeArtifact Autorisierungstoken 344
Ausfallsicherheit 345
Sicherheit der Infrastruktur 345
Angriffe durch Substitution von Abhängigkeiten 346
Identitäts- und Zugriffsverwaltung 347
Zielgruppe 347

Authentifizierung mit Identitäten 348
Verwalten des Zugriffs mit Richtlinien 352
Wie AWS CodeArtifact funktioniert mit IAM 355
Beispiele für identitätsbasierte Richtlinien 363
Verwenden von Tags zur Steuerung des Zugriffs auf CodeArtifact-Ressourcen 372
AWS CodeArtifact Referenz zu Berechtigungen 377
Fehlerbehebung 381
Arbeiten mit VPC-Endpunkten 383
Erstellen von VPC-Endpunkten 383
Erstellen Sie den Amazon S3-Gateway-Endpunkt 385
Mindestberechtigungen für Amazon S3 S3-Buckets für AWS CodeArtifact 385
CodeArtifact Von einer VPC aus verwenden 387
Verwenden Sie den `codeartifact.repositories` Endpunkt ohne privates DNS 388
Erstellen einer VPC-Endpunktrichtlinie 389
AWS CloudFormation Ressourcen 391
CodeArtifact und AWS CloudFormation Vorlagen 391
Das Löschen von CodeArtifact Ressourcen verhindern 391
Erfahren Sie mehr über AWS CloudFormation 392
Fehlerbehebung 393
Ich kann keine Benachrichtigungen sehen 393
Taggen von -Ressourcen 394
CodeArtifact Kostenzuweisung mit Tags 395
Zuweisung der Datenspeicherkosten in CodeArtifact 395
Zuordnung der Anforderungskosten in CodeArtifact 395
Kontingente in AWS CodeArtifact 396
Dokumentverlauf 400
..... cdxiii

Was ist AWS CodeArtifact?

AWS CodeArtifact ist ein sicherer, hoch skalierbarer, verwalteter Artefakt-Repository-Service, der Unternehmen dabei unterstützt, Softwarepakete für die Anwendungsentwicklung zu speichern und gemeinsam zu nutzen. Sie können es CodeArtifact mit gängigen Build-Tools und Paketmanagern wie NuGet CLI, Maven, Gradle, npm, yarn, pip und Twine verwenden. CodeArtifact trägt dazu bei, dass Sie nicht mehr Ihr eigenes Speichersystem für Artefakte verwalten oder sich Gedanken über die Skalierung der Infrastruktur machen müssen. Es gibt keine Beschränkungen für die Anzahl oder Gesamtgröße der Pakete, die Sie in einem CodeArtifact Repository speichern können.

Sie können eine Verbindung zwischen Ihrem privaten CodeArtifact Repository und einem externen, öffentlichen Repository wie npmjs.com oder Maven Central herstellen. CodeArtifact ruft dann Pakete auf Anfrage aus dem öffentlichen Repository ab und speichert sie, wenn sie von einem Paketmanager angefordert werden. Dies macht es bequemer, Open-Source-Abhängigkeiten zu nutzen, die von Ihrer Anwendung verwendet werden, und trägt dazu bei, dass sie immer für Builds und Entwicklungen verfügbar sind. Sie können auch private Pakete in einem CodeArtifact Repository veröffentlichen. Auf diese Weise können Sie proprietäre Softwarekomponenten von mehreren Anwendungs- und Entwicklungsteams in Ihrem Unternehmen gemeinsam nutzen.

Weitere Informationen finden Sie unter AWS CodeArtifact.

Wie CodeArtifact funktioniert das?

CodeArtifact speichert Softwarepakete in Repositorien. Repositorys sind mehrsprachig — ein einzelnes Repository kann Pakete aller unterstützten Typen enthalten. Jedes CodeArtifact Repository ist Mitglied einer einzigen Domain. CodeArtifact Wir empfehlen, dass Sie eine Produktionsdomäne für Ihre Organisation mit einem oder mehreren Repositorys verwenden. Sie können beispielsweise jedes Repository für ein anderes Entwicklungsteam verwenden. Pakete in Ihren Repositorys können dann entdeckt und von Ihren Entwicklungsteams gemeinsam genutzt werden.

Um Pakete zu einem Repository hinzuzufügen, konfigurieren Sie einen Paketmanager wie npm oder Maven so, dass er den Repository-Endpunkt (URL) verwendet. Anschließend können Sie den Paketmanager verwenden, um Pakete im Repository zu veröffentlichen. Sie können Open-Source-Pakete auch in ein Repository importieren, indem Sie es mit einer externen Verbindung zu einem öffentlichen Repository wie npmjs, NuGet Gallery, Maven Central oder PyPI konfigurieren. Weitere Informationen finden Sie unter Ein CodeArtifact Repository mit einem öffentlichen Repository Connect.

Sie können Pakete in einem Repository für ein anderes Repository in derselben Domain verfügbar machen. Um dies zu tun, konfigurieren Sie ein Repository als Upstream-Repository für das andere. Alle Paketversionen, die für das Upstream-Repository verfügbar sind, sind auch für das Downstream-Repository verfügbar. Darüber hinaus sind alle Pakete, die für das Upstream-Repository über eine externe Verbindung zu einem öffentlichen Repository verfügbar sind, auch für das Downstream-Repository verfügbar. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.

CodeArtifact erfordert, dass sich Benutzer beim Dienst authentifizieren, um Paketversionen zu veröffentlichen oder zu nutzen. Sie müssen sich beim CodeArtifact Dienst authentifizieren, indem Sie mit Ihren AWS Anmeldeinformationen ein Autorisierungstoken erstellen. Pakete in CodeArtifact Repositorys können nicht öffentlich zugänglich gemacht werden. Weitere Hinweise zur Authentifizierung und zum Zugriff in finden Sie CodeArtifact unterAWS CodeArtifact Authentifizierung und Tokens.

AWS CodeArtifact-Konzepte

Im Folgenden finden Sie einige Konzepte und Begriffe, die Sie bei der Verwendung CodeArtifact kennen sollten.

Themen

- Komponente
- Domain
- Repository
- Paket
- Paketgruppe
- Paket-Namespace
- Paketversion
- Revision der Paketversion
- Upstream-Repository

Komponente

Ein Asset ist eine einzelne Datei CodeArtifact , die in einer Paketversion gespeichert ist, z. B. eine `.tgz` NPM-Datei oder Maven POM- und JAR-Dateien.

Domain

Repositorys werden zu einer übergeordneten Entität zusammengefasst, die als Domain bezeichnet wird. Alle Paketressourcen und Metadaten werden in der Domain gespeichert, aber sie werden über Repositorys genutzt. Ein bestimmtes Paket-Asset, z. B. eine Maven-JAR-Datei, wird einmal pro Domain gespeichert, unabhängig davon, in wie vielen Repositorys es vorhanden ist. Alle Assets und Metadaten in einer Domain werden mit demselben AWS KMS key (KMS-Schlüssel) verschlüsselt, der in AWS Key Management Service (AWS KMS) gespeichert ist.

Jedes Repository ist Mitglied einer einzelnen Domain und kann nicht in eine andere Domain verschoben werden.

Mithilfe einer Domain können Sie eine Unternehmensrichtlinie auf mehrere Repositorys anwenden. Mit diesem Ansatz legen Sie fest, welche Konten auf Repositorys in der Domain zugreifen können und welche öffentlichen Repositorys als Paketquellen verwendet werden können.

Obwohl eine Organisation mehrere Domänen haben kann, empfehlen wir eine einzige Produktionsdomäne, die alle veröffentlichten Artefakte enthält. Auf diese Weise können Teams Pakete innerhalb Ihrer Organisation finden und gemeinsam nutzen.

Repository

Ein CodeArtifact Repository enthält eine Reihe von Paketversionen, von denen jede einer Reihe von Assets zugeordnet ist. Repositorys sind mehrsprachig — ein einzelnes Repository kann Pakete aller unterstützten Typen enthalten. Jedes Repository stellt Endpunkte zum Abrufen und Veröffentlichen von Paketen mithilfe von Tools wie der Nuget-CLI, der NPM-CLI, der Maven-CLI (mvn) und pip zur Verfügung. Sie können bis zu 1.000 Repositorys pro Domain erstellen.

Paket

Ein Paket ist ein Paket aus Software und Metadaten, das zur Auflösung von Abhängigkeiten und zur Installation der Software erforderlich ist. In CodeArtifact besteht ein Paket aus einem Paketnamen, einem optionalen Namespace wie `@types` in`@types/node`, einer Reihe von Paketversionen und Metadaten auf Paketebene wie npm-Tags.

AWS CodeArtifact unterstützt die Paketformate Cargo, Generic, Maven, npm NuGet, PyPI, Ruby und Swift.

Paketgruppe

Paketgruppen können verwendet werden, um die Konfiguration auf mehrere Pakete anzuwenden, die einem definierten Muster entsprechen, wobei das Paketformat, der Paketnamespace und der Paketname verwendet werden. Sie können Paketgruppen verwenden, um die Steuerung des Paketursprungs für mehrere Pakete bequemer zu konfigurieren. Kontrollen zur Paketherkunft werden verwendet, um die Aufnahme oder Veröffentlichung neuer Paketversionen zu blockieren oder zuzulassen. Dadurch werden Benutzer vor böswilligen Aktionen geschützt, die als Abhängigkeitsersetzungsangriffe bezeichnet werden.

Paket-Namespace

Einige Paketformate unterstützen hierarchische Paketnamen, um Pakete in logische Gruppen zu organisieren und Namenskollisionen zu vermeiden. Zum Beispiel unterstützt npm Bereiche. Weitere Informationen finden Sie in der Dokumentation zu npm-Scopes. Das npm-Paket `@types/node` hat den Gültigkeitsbereich `@types` und den Namen. `node` Es gibt viele andere Paketnamen im `@types` Gültigkeitsbereich. CodeArtifactIn wird der Bereich („Typen“) als Paket-Namespace und der Name („Knoten“) als Paketname bezeichnet. Bei Maven-Paketen entspricht der Paket-Namespace der Maven-GroupID. Das Maven-Paket `org.apache.logging.log4j:log4j` hat eine GroupID (Paket-Namespace) von `org.apache.logging.log4j` und die ArtifactID (Paketname). `log4j` Für generische Pakete ist ein Namespace erforderlich. Einige Paketformate wie PyPI unterstützen keine hierarchischen Namen mit einem ähnlichen Konzept wie npm scope oder Maven GroupID. Ohne eine Möglichkeit, Paketnamen zu gruppieren, kann es schwieriger sein, Namenskollisionen zu vermeiden.

Paketversion

Eine Paketversion identifiziert die spezifische Version eines Pakets, z. B. `@types/node 12.6.9` Das Format und die Semantik der Versionsnummer variieren je nach Paketformat. Beispielsweise müssen npm-Paketversionen der Semantic Versioning-Spezifikation entsprechen. In CodeArtifact besteht eine Paketversion aus der Versionskennung, Metadaten auf Paketversionsebene und einer Reihe von Assets.

Revision der Paketversion

Eine Paketversionsrevision ist eine Zeichenfolge, die einen bestimmten Satz von Ressourcen und Metadaten für eine Paketversion identifiziert. Jedes Mal, wenn eine Paketversion aktualisiert wird, wird eine neue Paketversionsrevision erstellt. Sie könnten beispielsweise ein Quellverteilungsarchiv (sdist) für eine Python-Paketversion veröffentlichen und später ein Python-Rad hinzufügen, das

kompilierten Code zu derselben Version enthält. Wenn Sie das Rad veröffentlichen, wird eine neue Version der Paketversion erstellt.

Upstream-Repository

Ein Repository ist einem anderen vorgelagert, wenn auf die darin enthaltenen Paketversionen vom Repository-Endpunkt des Downstream-Repositorys aus zugegriffen werden kann. Bei diesem Ansatz werden die Inhalte der beiden Repositorien aus Sicht eines Kunden effektiv zusammengeführt. Mithilfe CodeArtifact können Sie eine Upstream-Beziehung zwischen zwei Repositorys erstellen.

Wie fange ich an mit CodeArtifact?

Wir empfehlen, dass Sie zuerst die folgenden Schritte ausführen:

1. Erfahren Sie mehr darüber, CodeArtifact indem Sie lesenAWS CodeArtifact-Konzepte.
2. Richten Sie Ihren AWS-Konto AWS CLI, den und einen IAM-Benutzer ein, indem Sie die Schritte unter befolgen. Einrichtung mit AWS CodeArtifact
3. Folgen Sie zur Verwendung den Anweisungen CodeArtifact unterErste Schritte mit CodeArtifact.

Einrichtung mit AWS CodeArtifact

Wenn Sie bereits bei Amazon Web Services (AWS) registriert sind, können Sie AWS CodeArtifact sofort verwenden. Sie können die CodeArtifact Konsole öffnen, eine Domain und ein Repository erstellen auswählen und den Schritten im Startassistenten folgen, um Ihre erste Domain und Ihr erstes Repository zu erstellen.

Wenn Sie sich AWS noch nicht angemeldet haben oder Hilfe beim Erstellen Ihrer ersten Domain und Ihres ersten Repositorys benötigen, führen Sie die folgenden Aufgaben aus, um die Nutzung einzurichten CodeArtifact:

Themen

- Melde dich an für AWS
- Installieren oder aktualisieren Sie und konfigurieren Sie dann das AWS CLI
- Stellen Sie einen IAM-Benutzer bereit
- Installieren Sie Ihren Paketmanager oder Ihr Build-Tool

Melde dich an für AWS

Wenn Sie sich für Amazon Web Services (AWS) registrieren, werden Ihnen nur die Dienste und Ressourcen in Rechnung gestellt, die Sie nutzen, einschließlich AWS CodeArtifact.

Wenn Sie bereits eine haben AWS-Konto, fahren Sie mit der nächsten Aufgabe fortInstallieren oder aktualisieren Sie und konfigurieren Sie dann das AWS CLI. Wenn Sie noch keine haben AWS-Konto, gehen Sie wie folgt vor, um eine zu erstellen.

Um ein zu erstellen AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte

Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Installieren oder aktualisieren Sie und konfigurieren Sie dann das AWS CLI

Um CodeArtifact Befehle von AWS Command Line Interface (AWS CLI) auf einem lokalen Entwicklungscomputer aufzurufen, müssen Sie den installieren AWS CLI.

Wenn Sie eine ältere Version von AWS CLI installiert haben, müssen Sie sie aktualisieren, damit die CodeArtifact Befehle verfügbar sind. CodeArtifact Befehle sind in den folgenden AWS CLI Versionen verfügbar:

1. AWS CLI 1: 1.18.77 und neuer
2. AWS CLI 2:2.0.21 und neuer

Verwenden Sie den `aws --version` Befehl, um die Version zu überprüfen.

Um das zu installieren und zu konfigurieren AWS CLI

1. Installieren oder aktualisieren Sie das AWS CLI mit den Anweisungen unter Installation von AWS Command Line Interface.
2. Konfigurieren Sie AWS CLI die mit dem Befehl configure wie folgt.

   ```
   aws configure
   ```

   Wenn Sie dazu aufgefordert werden, geben Sie den AWS Zugriffsschlüssel und den AWS geheimen Zugriffsschlüssel des IAM-Benutzers an, den Sie mit CodeArtifact verwenden möchten. Wenn Sie zur Eingabe des AWS-Region Standardnamens aufgefordert werden, geben Sie die Region an, in der Sie die Pipeline erstellen möchten, z. B. `us-east-2` Wenn Sie nach dem standardmäßigen Ausgabeformat gefragt werden, geben Sie `json` an.

> **Important**
>
> Wenn Sie den konfigurieren AWS CLI, werden Sie aufgefordert, einen anzugeben AWS-Region. Wählen Sie eine der unterstützten Regionen aus, die unter Region und Endpoints in der Allgemeine AWS-Referenzaufgeführt sind.

Weitere Informationen finden Sie unter Konfiguration AWS Command Line Interface und Verwaltung der Zugriffsschlüssel für IAM-Benutzer.

3. Rufen Sie den folgenden Befehl vom auf, um die Installation oder das Upgrade zu überprüfen. AWS CLI

```
aws codeartifact help
```

Bei Erfolg zeigt dieser Befehl eine Liste der verfügbaren CodeArtifact Befehle an.

Als Nächstes können Sie einen IAM-Benutzer erstellen und diesem Benutzer Zugriff auf CodeArtifact gewähren. Weitere Informationen finden Sie unter Stellen Sie einen IAM-Benutzer bereit.

Stellen Sie einen IAM-Benutzer bereit

Folgen Sie diesen Anweisungen, um einen IAM-Benutzer auf die Verwendung vorzubereiten. CodeArtifact

Um einen IAM-Benutzer bereitzustellen

1. Erstellen Sie einen IAM-Benutzer oder verwenden Sie einen, der mit Ihrem verknüpft ist. AWS-Konto Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers und Überblick über AWS IAM-Richtlinien im IAM-Benutzerhandbuch.
2. Gewähren Sie dem IAM-Benutzer Zugriff auf. CodeArtifact

   - Option 1: Erstellen Sie eine benutzerdefinierte IAM-Richtlinie. Mit einer benutzerdefinierten IAM-Richtlinie können Sie die erforderlichen Mindestberechtigungen bereitstellen und die Gültigkeitsdauer von Authentifizierungstoken ändern. Weitere Informationen und Beispielrichtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact.

- Option 2: Verwenden Sie die `AWSCodeArtifactAdminAccess` AWS verwaltete Richtlinie. Der folgende Ausschnitt zeigt den Inhalt dieser Richtlinie.

> **Important**
>
> Diese Richtlinie gewährt allen Zugriff. CodeArtifact APIs Es wird empfohlen, immer die Mindestberechtigungen zu verwenden, die für die Ausführung Ihrer Aufgabe erforderlich sind. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

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

> **Note**
>
> Die `sts:GetServiceBearerToken` Berechtigung muss der IAM-Benutzer- oder Rollenrichtlinie hinzugefügt werden. Sie kann zwar zu einer CodeArtifact Domain- oder

Repository-Ressourcenrichtlinie hinzugefügt werden, hat jedoch keine Auswirkung auf Ressourcenrichtlinien.

Die `sts:GetServiceBearerToken` Genehmigung ist erforderlich, um die CodeArtifact `GetAuthorizationToken` API aufzurufen. Diese API gibt ein Token zurück, das verwendet werden muss, wenn Sie einen Paketmanager wie `npm` oder `pip` mit verwenden CodeArtifact. Um einen Paketmanager mit einem CodeArtifact Repository zu verwenden, muss Ihr IAM-Benutzer oder Ihre IAM-Rolle dies zulassen, `sts:GetServiceBearerToken` wie im vorherigen Richtlinienbeispiel gezeigt.

Falls Sie den Paketmanager oder das Build-Tool, das Sie verwenden möchten, noch nicht installiert haben CodeArtifact, finden Sie weitere Informationen unterInstallieren Sie Ihren Paketmanager oder Ihr Build-Tool.

Installieren Sie Ihren Paketmanager oder Ihr Build-Tool

Um Pakete von zu veröffentlichen oder zu nutzen CodeArtifact, müssen Sie einen Paketmanager verwenden. Für jeden Pakettyp gibt es unterschiedliche Paketmanager. Die folgende Liste enthält einige Paketmanager, die Sie mit verwenden können CodeArtifact. Falls Sie dies noch nicht getan haben, installieren Sie die Paketmanager für den Pakettyp, den Sie verwenden möchten.

- Verwenden Sie für npm die npm-CLI oder pnpm.
- Verwenden Sie für Maven entweder Apache Maven () oder Gradle. `mvn`
- Verwenden Sie für Python pip, um Pakete zu installieren, und Twine, um Pakete zu veröffentlichen.
- Verwenden Sie für NuGet das Toolkit for Visual Studio in Visual Studio oder das Nuget oder Dotnet. CLIs
- Verwenden Sie für generische Pakete das AWS CLIoder SDK, um Paketinhalte zu veröffentlichen und herunterzuladen.

Nächste Schritte

Ihre nächsten Schritte hängen davon ab, mit welchem Pakettyp oder welchen Pakettypen Sie arbeiten CodeArtifact, und vom Status Ihrer CodeArtifact Ressourcen.

Wenn Sie CodeArtifact für sich selbst, Ihr Team oder Ihre Organisation zum ersten Mal damit beginnen, finden Sie in der folgenden Dokumentation allgemeine Informationen zu den ersten Schritten und Hilfe bei der Erstellung der benötigten Ressourcen.

- Erste Schritte mit der Konsole
- Erste Schritte mit dem AWS CLI

Wenn Ihre Ressourcen bereits erstellt wurden und Sie bereit sind, Ihren Paketmanager so zu konfigurieren, dass er Pakete per Push in ein Repository überträgt oder Pakete aus einem CodeArtifact Repository installiert, lesen Sie in der Dokumentation nach, die Ihrem Pakettyp oder Paketmanager entspricht.

- CodeArtifact Mit npm verwenden
- CodeArtifact Mit Python verwenden
- Verwendung CodeArtifact mit Maven
- Verwenden CodeArtifact mit NuGet
- Verwendung CodeArtifact mit generischen Paketen

Erste Schritte mit CodeArtifact

In diesem Tutorial für die ersten Schritte erstellen CodeArtifact Sie Folgendes:

- Eine Domain namens`my-domain`.
- Ein Repository namens`my-repo`, das enthalten ist in`my-domain`.
- Ein Repository mit dem Namen`npm-store`, das enthalten ist in`my-domain`. Das `npm-store` hat eine externe Verbindung zum öffentlichen NPM-Repository. Diese Verbindung wird verwendet, um ein npm-Paket in das Repository aufzunehmen. `my-repo`

Bevor Sie mit diesem Tutorial beginnen, empfehlen wir Ihnen, es zu lesen. CodeArtifact AWS CodeArtifact-Konzepte

> **Note**
>
> In diesem Tutorial müssen Sie Ressourcen erstellen, die möglicherweise Kosten für Ihr AWS -Konto verursachen. Weitere Informationen finden Sie unter CodeArtifact Preise.

Themen

- Voraussetzungen
- Erste Schritte mit der Konsole
- Erste Schritte mit dem AWS CLI

Voraussetzungen

Sie können dieses Tutorial mit dem AWS Management Console oder dem AWS Command Line Interface (AWS CLI) abschließen. Um dem Tutorial folgen zu können, müssen Sie zunächst die folgenden Voraussetzungen erfüllen:

- Führen Sie die Schritte unter Einrichtung mit AWS CodeArtifact aus.
- Installieren Sie die npm-CLI. Weitere Informationen finden Sie unter Node.js und npm herunterladen und installieren in der npm-Dokumentation.

Erste Schritte mit der Konsole

Führen Sie die folgenden Schritte aus, um mit der CodeArtifact AWS Management Console Verwendung von zu beginnen. In diesem Handbuch wird der `npm` Paketmanager verwendet. Wenn Sie einen anderen Paketmanager verwenden, müssen Sie einige der folgenden Schritte ändern.

1. Melden Sie sich bei https://console.aws.amazon.com/codesuite/codeartifact/start an AWS Management Console und öffnen Sie die AWS CodeArtifact Konsole. Weitere Informationen finden Sie unter Einrichtung mit AWS CodeArtifact.
2. Wählen Sie Repository erstellen aus.
3. Geben Sie im Feld Repository-Name den Wert ein. **`my-repo`**
4. (Optional) Geben Sie unter Repository-Beschreibung eine optionale Beschreibung für Ihr Repository ein.
5. Wählen Sie unter Öffentliche Upstream-Repositorys die Option npm-store aus, um ein mit npmjs verbundenes Repository zu erstellen, das Ihrem Repository vorgeschaltet ist. `my-repo`

   CodeArtifact weist diesem Repository den Namen `npm-store` für Sie zu. Alle Pakete, die im Upstream-Repository verfügbar sind, `npm-store` sind auch für das entsprechende Downstream-Repository verfügbar. `my-repo`
6. Wählen Sie Weiter.
7. Wählen Sie unter AWS-Konto die Option Dieses AWS-Konto aus.
8. Geben Sie im Feld Domainname den Wert ein**`my-domain`**.
9. Erweitern Sie Additional configuration (Zusätzliche Konfiguration).
10. Sie müssen einen AWS KMS key (KMS-Schlüssel) verwenden, um alle Ressourcen in Ihrer Domain zu verschlüsseln. Sie können einen Von AWS verwalteter Schlüssel oder einen KMS-Schlüssel verwenden, den Sie verwalten:

    - Wählen Sie AWS-verwalteten Schlüssel, wenn Sie den Standard verwenden möchten Von AWS verwalteter Schlüssel.
    - Wählen Sie Vom Kunden verwalteter Schlüssel, wenn Sie einen von Ihnen verwalteten KMS-Schlüssel verwenden möchten. Um einen KMS-Schlüssel zu verwenden, den Sie verwalten, suchen Sie unter ARN für vom Kunden verwalteten Schlüssel nach dem KMS-Schlüssel und wählen Sie ihn aus.

Weitere Informationen finden Sie unter Von AWS verwalteter Schlüsselund vom Kunden verwalteter Schlüssel im AWS Key Management Service Entwicklerhandbuch.

11. Wählen Sie Weiter.
12. Prüfen Sie unter Überprüfen und erstellen, CodeArtifact was für Sie erstellt wurde.

    - Der Paketfluss zeigt `my-domain`my-repo, wie, und `npm-store` zusammenhängen.
    - Schritt 1: Repository erstellen zeigt Details zu `my-repo` und`npm-store`.
    - Schritt 2: Domain auswählen zeigt Details zu`my-domain`.

    Wenn Sie bereit sind, wählen Sie Repository erstellen.
13. Wählen Sie auf der Seite my-repo die Option Verbindungsanweisungen anzeigen und dann npm aus.
14. Verwenden Sie den AWS CLI , um den unter Konfigurieren Sie Ihren NPM-Client mit diesem `login` Befehl angezeigten Befehl auszuführen. AWS CLI CodeArtifact

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

    Sie sollten eine Ausgabe erhalten, die bestätigt, dass Ihre Anmeldung erfolgreich war.

    ```
    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
    ```

    Wenn Sie den Fehler erhalten`Could not connect to the endpoint URL`, stellen Sie sicher, dass Ihre konfiguriert AWS CLI ist und dass Ihr Standardregionsname auf dieselbe Region gesetzt ist, in der Sie Ihr Repository erstellt haben. Weitere Informationen finden Sie unter Konfiguration der AWS-Befehlszeilenschnittstelle.

    Weitere Informationen finden Sie unter Konfigurieren und verwenden Sie npm mit CodeArtifact
15. Verwenden Sie die npm-CLI, um ein npm-Paket zu installieren. Verwenden Sie beispielsweise den folgenden Befehl, um das beliebte npm-Paket `lodash` zu installieren.

    ```
    npm install lodash
    ```

16. Kehren Sie zur CodeArtifact Konsole zurück. Wenn Ihr my-repo-Repository geöffnet ist, aktualisieren Sie die Seite. Andernfalls wählen Sie im Navigationsbereich Repositories und dann my-repo aus.

    Unter Pakete sollten Sie die NPM-Bibliothek oder das Paket sehen, das Sie installiert haben. Sie können den Namen des Pakets wählen, um dessen Version und Status anzuzeigen. Sie können die neueste Version auswählen, um Paketdetails wie Abhängigkeiten, Ressourcen und mehr anzuzeigen.

    > **Note**
    >
    > Es kann zu einer Verzögerung zwischen der Installation des Pakets und der Aufnahme in Ihr Repository kommen.

17. Um weitere AWS Gebühren zu vermeiden, löschen Sie die Ressourcen, die Sie in diesem Tutorial verwendet haben:

    > **Note**
    >
    > Eine Domain, die Repositorys enthält, kann nicht gelöscht werden. Deshalb müssen Sie `npm-store` zuerst `my-repo` und löschen`my-domain`.

    a. Wählen Sie im Navigationsbereich Repositories aus.
    b. Wählen Sie npm-store, wählen Sie Löschen und folgen Sie dann den Schritten zum Löschen des Repositorys.
    c. Wählen Sie my-repo, wählen Sie Löschen und folgen Sie dann den Schritten zum Löschen des Repositorys.
    d. Wählen Sie im Navigationsbereich Domains aus.
    e. Wählen Sie my-domain und anschließend Delete aus, und folgen Sie dann den Schritten zum Löschen der Domain.

Erste Schritte mit dem AWS CLI

Führen Sie die folgenden Schritte aus, um mit der CodeArtifact Verwendung von AWS Command Line Interface (AWS CLI) zu beginnen. Weitere Informationen finden Sie unter Installieren oder

aktualisieren Sie und konfigurieren Sie dann das AWS CLI. In diesem Handbuch wird der npm Paketmanager verwendet. Wenn Sie einen anderen Paketmanager verwenden, müssen Sie einige der folgenden Schritte ändern.

1. Verwenden Sie den AWS CLI , um den create-domain Befehl auszuführen.

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

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrer neuen Domain angezeigt.

```
{
    "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
    }
}
```

Wenn Sie den Fehler erhaltenCould not connect to the endpoint URL, stellen Sie sicher, dass Ihre konfiguriert AWS CLI ist und dass Ihr Standardregionsname auf dieselbe Region gesetzt ist, in der Sie Ihr Repository erstellt haben. Weitere Informationen finden Sie unter Konfiguration der AWS-Befehlszeilenschnittstelle.

2. Verwenden Sie den create-repository Befehl, um ein Repository in Ihrer Domain zu erstellen.

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

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrem neuen Repository angezeigt.

```
{
    "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. Verwenden Sie den create-repository Befehl, um ein Upstream-Repository für Ihr Repository zu erstellen. `my-repo`

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

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrem neuen Repository angezeigt.

```
{
    "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. Verwenden Sie den associate-external-connection Befehl, um Ihrem Repository eine externe Verbindung zum öffentlichen NPM-Repository hinzuzufügen. `npm-store`

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

In der Ausgabe werden Daten im JSON-Format mit Details zum Repository und seiner neuen externen Verbindung angezeigt.

```
{
```

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

Weitere Informationen finden Sie unter Ein CodeArtifact Repository mit einem öffentlichen Repository Connect.

5. Verwenden Sie den update-repository Befehl, um das `npm-store` Repository dem Repository als Upstream-Repository zuzuordnen. `my-repo`

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

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrem aktualisierten Repository angezeigt, einschließlich des neuen Upstream-Repositorys.

```
{
    "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": []
    }
}
```

Weitere Informationen finden Sie unter Fügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI.

6. Verwenden Sie den login Befehl, um Ihren npm-Paketmanager mit Ihrem Repository zu konfigurieren. `my-repo`

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

Sie sollten eine Ausgabe erhalten, die bestätigt, dass Ihre Anmeldung erfolgreich war.

```
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
```

Weitere Informationen finden Sie unter Konfigurieren und verwenden Sie npm mit CodeArtifact.

7. Verwenden Sie die npm-CLI, um ein npm-Paket zu installieren. Verwenden Sie beispielsweise den folgenden Befehl, um das beliebte npm-Paket `lodash` zu installieren.

```
npm install lodash
```

8. Verwenden Sie den list-packages Befehl, um das Paket anzuzeigen, das Sie gerade in Ihrem `my-repo` Repository installiert haben.

> **Note**
>
> Zwischen dem Abschluss des `npm install` Installationsbefehls und dem Zeitpunkt, an dem das Paket in Ihrem Repository sichtbar ist, kann es zu einer Verzögerung kommen. Einzelheiten zur typischen Latenz beim Abrufen von Paketen aus öffentlichen Repositorien finden Sie unter. Latenz bei externen Verbindungen

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

JSON-formatierte Daten werden in der Ausgabe mit dem Format und dem Namen des Pakets angezeigt, das Sie installiert haben.

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

Sie haben jetzt drei Ressourcen: CodeArtifact

- Die Domain`my-domain`.
- Das Repository`my-repo`, das in enthalten ist`my-domain`. Für dieses Repository steht ein npm-Paket zur Verfügung.
- Das Repository`npm-store`, das in `my-domain` enthalten ist. Dieses Repository hat eine externe Verbindung zum öffentlichen NPM-Repository und ist dem Repository als Upstream-Repository zugeordnet. `my-repo`

9. Um weitere AWS Gebühren zu vermeiden, löschen Sie die Ressourcen, die Sie in diesem Tutorial verwendet haben:

> **Note**
>
> Eine Domain, die Repositorys enthält, kann nicht gelöscht werden. Deshalb müssen Sie `npm-store` zuerst `my-repo` und löschen`my-domain`.

a. Verwenden Sie den delete-repository Befehl, um das `npm-store` Repository zu löschen.

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

In der Ausgabe werden Daten im JSON-Format mit Details zum gelöschten Repository angezeigt.

```
{
    "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. Verwenden Sie den delete-repository Befehl, um das Repository zu löschen. `npm-store`

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

In der Ausgabe werden Daten im JSON-Format mit Details zum gelöschten Repository angezeigt.

```
{
    "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. Verwenden Sie den delete-domain Befehl, um das Repository zu löschen. `my-domain`

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

In der Ausgabe werden Daten im JSON-Format mit Details zur gelöschten Domain angezeigt.

```
{
    "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
    }
}
```

Arbeiten mit Repositorien in CodeArtifact

In diesen Themen erfahren Sie, AWS CLI wie Sie die CodeArtifact Konsole verwenden und CodeArtifact APIs Repositorys erstellen, auflisten, aktualisieren und löschen.

Themen

- Erstellen eines -Repositorys
- Herstellen einer Verbindung mit einem Repository
- Löschen Sie ein Repository
- Repositorys auflisten
- Eine Repository-Konfiguration anzeigen oder ändern
- Repository-Richtlinien
- Kennzeichnen Sie ein Repository in CodeArtifact

Erstellen eines -Repositorys

Da alle Pakete in Repositorys gespeichert CodeArtifact sind, müssen Sie zur Verwendung CodeArtifact eines erstellen. Sie können ein Repository mithilfe der CodeArtifact Konsole, der Taste AWS Command Line Interface (AWS CLI) oder AWS CloudFormation erstellen. Jedes Repository ist dem AWS Konto zugeordnet, das Sie bei der Erstellung verwenden. Sie können mehrere Repositorys haben, und diese werden erstellt und in Domänen gruppiert. Wenn Sie ein Repository erstellen, enthält es keine Pakete. Repositorys sind polyglot, was bedeutet, dass ein einzelnes Repository Pakete aller unterstützten Typen enthalten kann.

Hinweise zu CodeArtifact Dienstbeschränkungen, wie z. B. der maximal zulässigen Anzahl von Repositorys in einer einzelnen Domäne, finden Sie unter. Kontingente in AWS CodeArtifact Wenn Sie die maximale Anzahl erlaubter Repositorys erreicht haben, können Sie Repositorys löschen, um Platz für weitere zu schaffen.

Einem Repository können ein oder mehrere CodeArtifact Repositorys als Upstream-Repositorys zugeordnet sein. Auf diese Weise kann ein Paketmanager-Client über einen einzigen URL-Endpunkt auf die Pakete zugreifen, die in mehr als einem Repository enthalten sind. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.

Weitere Hinweise zur Verwaltung von CodeArtifact Repositorys mit finden Sie CloudFormation unter CodeArtifact Ressourcen erstellen mit AWS CloudFormation.

> **Note**
>
> Nachdem Sie ein Repository erstellt haben, können Sie seinen Namen, sein zugeordnetes AWS Konto oder seine Domäne nicht mehr ändern.

Themen

- Erstellen Sie ein Repository (Konsole)
- Erstellen Sie ein Repository (AWS CLI)
- Erstellen Sie ein Repository mit einem Upstream-Repository

Erstellen Sie ein Repository (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Repositories und dann Create repository aus.
3. Geben Sie unter Repository-Name einen Namen für Ihr Repository ein.
4. (Optional) Geben Sie unter Repository-Beschreibung eine optionale Beschreibung für Ihr Repository ein.
5. (Optional) Fügen Sie unter Upstream-Repositorys veröffentlichen Zwischenrepositorys hinzu, die Ihre Repositorys mit Paketbehörden wie Maven Central oder npmjs.com verbinden.
6. Wählen Sie Weiter.
7. Wählen Sie im AWS-Konto die Option Dieses AWS-Konto aus, wenn Sie bei dem Konto angemeldet sind, dem die Domain gehört. Wählen Sie Anderes AWS-Konto, wenn ein anderes AWS-Konto die Domain besitzt.
8. Wählen Sie unter Domain die Domain aus, in der das Repository erstellt werden soll.

   Wenn das Konto keine Domänen enthält, müssen Sie eine erstellen. Geben Sie den Namen für die neue Domain im Feld Domainname ein.

   Erweitern Sie Additional configuration (Zusätzliche Konfiguration).

   Sie müssen einen AWS KMS key (KMS-Schlüssel) verwenden, um alle Ressourcen in Ihrer Domain zu verschlüsseln. Sie können einen Von AWS verwalteter Schlüssel oder einen KMS-Schlüssel verwenden, den Sie verwalten:

> **Important**
>
> CodeArtifact unterstützt nur symmetrische KMS-Schlüssel. Sie können keinen asymmetrischen KMS-Schlüssel verwenden, um Ihre Domänen zu verschlüsseln. CodeArtifact Wie Sie feststellen, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, erfahren Sie unter Erkennen symmetrischer und asymmetrischer KMS-Schlüssel.

- Wählen Sie AWS-verwalteten Schlüssel, wenn Sie den Standard verwenden möchten Von AWS verwalteter Schlüssel.
- Wählen Sie Vom Kunden verwalteter Schlüssel, wenn Sie einen von Ihnen verwalteten KMS-Schlüssel verwenden möchten. Um einen KMS-Schlüssel zu verwenden, den Sie verwalten, suchen Sie unter ARN für vom Kunden verwalteten Schlüssel nach dem KMS-Schlüssel und wählen Sie ihn aus.

Weitere Informationen finden Sie unter Von AWS verwaltete SchlüsselVom Kunden verwalteter Schlüssel im AWS Key Management Service Entwicklerhandbuch.

9. Wählen Sie Weiter.
10. Prüfen Sie unter Überprüfen und erstellen, CodeArtifact was für Sie erstellt wurde.

    - Der Paketfluss zeigt, wie Ihre Domain und Ihre Repositorys miteinander verbunden sind.
    - Schritt 1: Repository erstellen zeigt Details zum Repository und zu optionalen Upstream-Repositorys, die erstellt werden.
    - Schritt 2: Domain auswählen zeigt Details zu`my_domain`.

    Wenn Sie bereit sind, wählen Sie Repository erstellen.

Erstellen Sie ein Repository (AWS CLI)

Verwenden Sie den `create-repository` Befehl, um ein Repository in Ihrer Domain zu erstellen.

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

Beispielausgabe:

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

Ein neues Repository enthält keine Pakete. Jedes Repository ist dem AWS Konto zugeordnet, mit dem Sie bei der Erstellung des Repositorys authentifiziert wurden.

Erstellen Sie ein Repository mit Tags

Um ein Repository mit Tags zu erstellen, fügen Sie den `--tags` Parameter zu Ihrem `create-domain` Befehl hinzu.

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

Erstellen Sie ein Repository mit einem Upstream-Repository

Sie können ein oder mehrere Upstream-Repositorys angeben, wenn Sie ein Repository erstellen.

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

Beispielausgabe:

```
{
    "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": [
            {
                "repositoryName": "my-upstream-repo"
            }
        ],
        "externalConnections"" "[]"
    }
}
```

> **Note**
>
> Um ein Repository mit einem Upstream-Repository zu erstellen, benötigen Sie die Erlaubnis für die `AssociateWithDownstreamRepository` Aktion im Upstream-Repository.

Informationen zum Hinzufügen eines Upstream-Objekts zu einem Repository, nachdem es erstellt wurde, finden Sie unter Upstream-Repositorys hinzufügen oder entfernen (Konsole) undFügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI.

Herstellen einer Verbindung mit einem Repository

Nachdem Sie Ihr Profil und Ihre Anmeldeinformationen für die Authentifizierung bei Ihrem AWS Konto konfiguriert haben, entscheiden Sie, in welchem Repository Sie es verwenden möchten. CodeArtifact Ihnen stehen folgende Optionen zur Verfügung:

- Erstellen Sie ein -Repository. Weitere Informationen finden Sie unter Erstellen eines Repositorys.
- Verwenden Sie ein Repository, das bereits in Ihrem Konto vorhanden ist. Sie können den `list-repositories` Befehl verwenden, um die in Ihrem AWS Konto erstellten Repositorys zu finden. Weitere Informationen finden Sie unter Repositorys auflisten.
- Verwenden Sie ein Repository in einem anderen AWS Konto. Weitere Informationen finden Sie unter Repository-Richtlinien.

Verwenden Sie einen Paketmanager-Client

Wenn Sie wissen, welches Repository Sie verwenden möchten, schauen Sie sich eines der folgenden Themen an.

- Verwendung CodeArtifact mit Maven
- Verwendung CodeArtifact mit npm
- Verwenden mit CodeArtifact NuGet
- CodeArtifact Mit Python verwenden

Löschen Sie ein Repository

Sie können ein Repository mit der CodeArtifact Konsole oder dem löschen AWS CLI. Nachdem ein Repository gelöscht wurde, können Sie keine Pakete mehr dorthin übertragen oder Pakete daraus abrufen. Alle Pakete im Repository sind dauerhaft nicht mehr verfügbar und können nicht wiederhergestellt werden. Sie können ein Repository mit demselben Namen erstellen, dessen Inhalt jedoch leer ist.

> **Important**
>
> Das Löschen eines Repositorys kann nicht rückgängig gemacht werden. Nachdem Sie ein Repository gelöscht haben, können Sie es nicht mehr wiederherstellen und es kann auch nicht wiederhergestellt werden.

Themen

- Löschen Sie ein Repository (Konsole)
- Löschen Sie ein Repository (AWS CLI)
- Schützt Repositorys vor dem Löschen

Löschen Sie ein Repository (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.

2. Wählen Sie im Navigationsbereich Repositories und dann das Repository aus, das Sie löschen möchten.
3. Wählen Sie Löschen und folgen Sie dann den Schritten zum Löschen der Domain.

Löschen Sie ein Repository (AWS CLI)

Verwenden Sie den `delete-repository` Befehl, um ein Repository zu löschen.

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

Beispielausgabe:

```
{
    "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": []
    }
}
```

Schützt Repositorys vor dem Löschen

Du kannst verhindern, dass ein Repository versehentlich gelöscht wird, indem du eine Domain-Richtlinie einfügst, die der folgenden ähnelt:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyRepositoryDeletion",
            "Action": [
                "codeartifact:DeleteRepository"
            ],
```

```
            "Effect": "Deny",
            "Resource": "*",
            "Principal": *
        }
    ]
}
```

Diese Richtlinie verhindert, dass alle Prinzipale das Repository löschen. Wenn Sie jedoch später entscheiden, dass Sie das Repository löschen müssen, können Sie dies tun, indem Sie die folgenden Schritte ausführen:

1. Aktualisieren Sie die Richtlinie in der Domänenrichtlinie wie folgt:

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

   *repository-arn*Ersetzen Sie es durch den ARN des Repositorys, das Sie löschen möchten.
2. Wählen Sie in der AWS CodeArtifact Konsole Repositories und löschen Sie das gewählte Repository.
3. Nachdem Sie das Repository gelöscht haben, können Sie die Richtlinie wieder ändern, um versehentliche Löschungen zu verhindern.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "DenyRepositoryDeletion",
               "Action": [
                   "codeartifact:DeleteRepository"
   ```

```
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Principal": *
        }
    ]
}
```

Alternativ können Sie dieselbe Ablehnungsaussage in eine Repository-Richtlinie aufnehmen. Auf diese Weise haben Sie mehr Flexibilität, um hochwertige Repositorys vor dem Löschen zu schützen.

Repositorys auflisten

Verwenden Sie die Befehle in diesem Thema, um Repositorys in einem AWS Konto oder einer Domain aufzulisten.

Listet die Repositorys in einem Konto auf AWS

Verwenden Sie diesen Befehl, um alle Repositorys in Ihrem AWS Konto aufzulisten.

```
aws codeartifact list-repositories
```

Beispielausgabe:

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

Sie können die Antwort `list-repositories` mithilfe der Parameter `--max-results` und `--next-token` paginieren. Geben Sie für `--max-results` eine Ganzzahl zwischen 1 und 1000 an, um die Anzahl der auf einer einzelnen Seite zurückgegebenen Ergebnisse anzugeben. Die Standardeinstellung ist 50. Um nachfolgende Seiten zurückzugeben, führen Sie den `list-repositories` Vorgang erneut aus und übergeben Sie den in der vorherigen Befehlsausgabe empfangenen `nextToken` Wert an`--next-token`. Wenn die `--next-token` Option nicht verwendet wird, wird immer die erste Ergebnisseite zurückgegeben.

Listet Repositorys in der Domäne auf

`list-repositories-in-domain`Dient zum Abrufen einer Liste aller Repositorys in einer Domain.

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

Die Ausgabe zeigt, dass einige der Repositorys von unterschiedlichen AWS Konten verwaltet werden.

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

Sie können die Antwort `list-repositories-in-domain` mit den Parametern `--max-results` und `--next-token` paginieren. Geben Sie für `--max-results` eine Ganzzahl zwischen 1 und 1000 an, um die Anzahl der auf einer einzelnen Seite zurückgegebenen Ergebnisse anzugeben. Die Standardeinstellung ist 50. Um nachfolgende Seiten zurückzugeben, führen Sie den `list-repositories-in-domain` Vorgang erneut aus und übergeben Sie den in der vorherigen Befehlsausgabe empfangenen `nextToken` Wert an`--next-token`. Wenn die `--next-token` Option nicht verwendet wird, wird immer die erste Ergebnisseite zurückgegeben.

Um die Repository-Namen in einer kompakteren Liste auszugeben, versuchen Sie es mit dem folgenden Befehl.

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

Beispielausgabe:

```
repo1
repo2
repo3
```

Im folgenden Beispiel wird zusätzlich zum Repository-Namen die Konto-ID ausgegeben.

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

Beispielausgabe:

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

Weitere Informationen zu dem `--query` Parameter finden Sie ListRepositoriesin der CodeArtifact API-Referenz.

Eine Repository-Konfiguration anzeigen oder ändern

Sie können Details zu Ihrem Repository mit der CodeArtifact Konsole oder der AWS Command Line Interface (AWS CLI) anzeigen und aktualisieren.

> **Note**
>
> Nachdem Sie ein Repository erstellt haben, können Sie seinen Namen, sein zugeordnetes AWS Konto oder seine Domäne nicht mehr ändern.

Themen

- Eine Repository-Konfiguration anzeigen oder ändern (Konsole)
- Eine Repository-Konfiguration anzeigen oder ändern ()AWS CLI

Eine Repository-Konfiguration anzeigen oder ändern (Konsole)

In der CodeArtifact Konsole können Sie Details zu Ihrem Repository anzeigen und es aktualisieren.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Repositorys und dann den Repository-Namen aus, den Sie anzeigen oder ändern möchten.
3. Erweitern Sie Details, um Folgendes zu sehen:

   - Die Domain des Repositorys. Wählen Sie den Domainnamen, um mehr darüber zu erfahren.
   - Die Ressourcenrichtlinie des Repositorys. Wählen Sie Repository-Richtlinie anwenden aus, um eine hinzuzufügen.
   - Der Amazon-Ressourcenname (ARN) des Repositorys.
   - Wenn Ihr Repository über eine externe Verbindung verfügt, können Sie die Verbindung auswählen, um mehr darüber zu erfahren. Ein Repository kann nur eine externe Verbindung haben. Weitere Informationen finden Sie unter Ein CodeArtifact Repository mit einem öffentlichen Repository Connect.
   - Wenn dein Repository über Upstream-Repositorys verfügt, kannst du eines auswählen, um dessen Details zu sehen. Ein Repository kann bis zu 10 direkte Upstream-Repositorys haben. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.

   > **Note**
   >
   > Ein Repository kann eine externe Verbindung oder Upstream-Repositorys haben, aber nicht beides.

4. Unter Pakete können Sie alle Pakete sehen, die für dieses Repository verfügbar sind. Wählen Sie ein Paket aus, um mehr darüber zu erfahren.
5. Wählen Sie Verbindungsanweisungen anzeigen und wählen Sie dann einen Paketmanager aus, um zu erfahren, wie Sie ihn konfigurieren können CodeArtifact.
6. Wählen Sie Repository-Richtlinie anwenden, um Ihr Repository zu aktualisieren oder eine Ressourcenrichtlinie hinzuzufügen. Weitere Informationen finden Sie unter Repository-Richtlinien.
7. Wählen Sie Bearbeiten, um Folgendes hinzuzufügen oder zu aktualisieren.

   - Die Beschreibung des Repositorys.
   - Mit dem Repository verknüpfte Tags.

- Wenn dein Repository über eine externe Verbindung verfügt, kannst du ändern, mit welchem öffentlichen Repository es verbunden ist. Andernfalls können Sie ein oder mehrere bestehende Repositorys als Upstream-Repositorys hinzufügen. Ordnen Sie sie in der Reihenfolge an, in der sie priorisiert werden sollen, CodeArtifact wenn ein Paket angefordert wird. Weitere Informationen finden Sie unter Reihenfolge der Prioritäten des Upstream-Reposit.

Eine Repository-Konfiguration anzeigen oder ändern ()AWS CLI

Verwenden Sie den `describe-repository` Befehl CodeArtifact, um die aktuelle Konfiguration eines Repositorys in anzuzeigen.

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

Beispielausgabe:

```
{
    "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": []
    }
}
```

Ändern Sie die Upstream-Konfiguration eines Repositor

Ein Upstream-Repository ermöglicht es einem Paketmanager-Client, über einen einzigen URL-Endpunkt auf die Pakete zuzugreifen, die in mehr als einem Repository enthalten sind. Verwenden Sie den `update-repository` Befehl, um die Upstream-Beziehung eines Repositorys hinzuzufügen oder zu ändern.

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

Beispielausgabe:

```
{
    "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**
>
> Um ein Upstream-Repository hinzuzufügen, benötigen Sie die Berechtigung für die `AssociateWithDownstreamRepository` Aktion im Upstream-Repository.

Um die Upstream-Beziehung eines Repositorys zu entfernen, verwenden Sie eine leere Liste als Argument für die `--upstreams` Option.

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

Beispielausgabe:

```
{
    "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": []
    }
}
```

Repository-Richtlinien

CodeArtifact verwendet ressourcenbasierte Berechtigungen, um den Zugriff zu kontrollieren. Mit ressourcenbasierten Berechtigungen können Sie angeben, welche Benutzer auf ein Repository zugreifen können und welche Aktionen ausführbar sind. Standardmäßig hat nur der Repository-Besitzer Zugriff auf das Repository. Sie können ein Richtliniendokument anwenden, das anderen IAM-Prinzipalen den Zugriff auf Ihr Repository ermöglicht.

Weitere Informationen finden Sie unter Ressourcenbasierte Richtlinien und Identitätsbasierte Richtlinien und Ressourcenbasierte Richtlinien.

Erstellen Sie eine Ressourcenrichtlinie, um Lesezugriff zu gewähren

Eine Ressourcenrichtlinie ist eine Textdatei im JSON-Format. Die Datei muss einen Prinzipal (Akteur), eine oder mehrere Aktionen und einen Effekt (`Allow`oder`Deny`) angeben. Die folgende Ressourcenrichtlinie gewährt dem Konto beispielsweise die `123456789012` Erlaubnis, Pakete aus dem Repository herunterzuladen.

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

Da die Richtlinie nur für Operationen mit dem Repository ausgewertet wird, an das sie angehängt ist, müssen Sie keine Ressource angeben. Da die Ressource impliziert ist, können Sie den Wert `Resource` auf * setzen. Damit ein Paketmanager ein Paket aus diesem Repository herunterladen kann, muss auch eine Domänenrichtlinie für den kontoübergreifenden Zugriff erstellt werden. Die Domänenrichtlinie muss dem Prinzipal mindestens die `codeartifact:GetAuthorizationToken` Erlaubnis erteilen. Ein Beispiel für eine vollständige Domänenrichtlinie zur Gewährung von kontenübergreifendem Zugriff finden Sie hierBeispiel für eine Domänenrichtlinie.

> **Note**
>
> Die `codeartifact:ReadFromRepository` Aktion kann nur für eine Repository-Ressource verwendet werden. Sie können den Amazon-Ressourcennamen (ARN) eines Pakets nicht als Ressource angeben`codeartifact:ReadFromRepository`, um Lesezugriff auf eine Teilmenge von Paketen in einem Repository zu ermöglichen. Ein bestimmter Principal kann entweder alle Pakete in einem Repository lesen oder keines davon.

Weil die einzige im Repository angegebene Aktion darin besteht`ReadFromRepository`, dass Benutzer und Rollen aus dem Konto Pakete aus dem Repository herunterladen `1234567890` können. Sie können jedoch keine anderen Aktionen an ihnen ausführen (z. B. das Auflisten von Paketnamen und -versionen). In der Regel gewähren Sie in der folgenden Richtlinie außerdem Berechtigungen, `ReadFromRepository` weil ein Benutzer, der Pakete aus einem Repository herunterlädt, auch auf andere Weise mit diesem interagieren muss.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:DescribePackageVersion",
                "codeartifact:DescribeRepository",
                "codeartifact:GetPackageVersionReadme",
                "codeartifact:GetRepositoryEndpoint",
                "codeartifact:ListPackages",
                "codeartifact:ListPackageVersions",
                "codeartifact:ListPackageVersionAssets",
                "codeartifact:ListPackageVersionDependencies",
                "codeartifact:ReadFromRepository"
            ],
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "*"
        }
    ]
}
```

Legen Sie eine Richtlinie fest

Nachdem Sie ein Richtliniendokument erstellt haben, verwenden Sie den `put-repository-permissions-policy` Befehl, um es an ein Repository anzuhängen:

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

Wenn Sie aufrufen`put-repository-permissions-policy`, wird die Ressourcenrichtlinie für das Repository bei der Auswertung der Berechtigungen ignoriert. Dadurch wird sichergestellt, dass sich der Besitzer einer Domain nicht selbst aus dem Repository aussperren kann, was ihn daran hindern würde, die Ressourcenrichtlinie zu aktualisieren.

> **Note**
>
> Sie können einem anderen AWS Konto keine Berechtigungen zur Aktualisierung der Ressourcenrichtlinie für ein Repository mithilfe einer Ressourcenrichtlinie gewähren, da die Ressourcenrichtlinie beim Aufrufen ignoriert wird put-repository-permissions-policy.

Beispielausgabe:

```
{
    "policy": {
        "resourceArn": "arn:aws:codeartifact:region-
id:111122223333:repository/my_domain/my_repo",
        "document": "{ ...policy document content...}",
        "revision": "MQlyyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxx="
    }
}
```

Die Ausgabe des Befehls enthält den Amazon-Ressourcennamen (ARN) der Repository-Ressource, den vollständigen Inhalt des Richtliniendokuments und eine Revisions-ID. Sie können die Revisions-ID an die `put-repository-permissions-policy` Verwendung der `--policy-revision` Option übergeben. Dadurch wird sichergestellt, dass eine bekannte Version des Dokuments überschrieben wird und nicht eine neuere Version, die von einem anderen Autor festgelegt wurde.

Lesen Sie eine Richtlinie

Verwenden Sie den `get-repository-permissions-policy` Befehl, um eine vorhandene Version eines Richtliniendokuments zu lesen. Verwenden Sie das `--output` und `--query policy.document` zusammen mit dem `json.tool` Python-Modul, um die Ausgabe lesbar zu formatieren.

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

Beispielausgabe:

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

```
}
```

Löschen Sie eine Richtlinie

Verwenden Sie den `delete-repository-permissions-policy` Befehl, um eine Richtlinie aus einem Repository zu löschen.

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

Das Format der Ausgabe entspricht dem des `get-repository-permissions-policy` Befehls.

Gewähren Sie den Hauptbenutzern Lesezugriff

Wenn Sie in einem Richtliniendokument den Root-Benutzer eines Kontos als Hauptbenutzer angeben, gewähren Sie allen Benutzern und Rollen in diesem Konto Zugriff. Um den Zugriff auf ausgewählte Benutzer oder Rollen zu beschränken, verwenden Sie deren ARN im `Principal` Abschnitt der Richtlinie. Verwenden Sie beispielsweise Folgendes, um dem bob IAM-Benutzerkonto `123456789012` Lesezugriff zu gewähren.

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

Gewähren Sie Schreibzugriff auf Pakete

Die `codeartifact:PublishPackageVersion` Aktion wird verwendet, um die Erlaubnis zur Veröffentlichung neuer Versionen eines Pakets zu kontrollieren. Bei der mit dieser Aktion

verwendeten Ressource muss es sich um ein Paket handeln. Das Format des CodeArtifact Pakets ARNs ist wie folgt.

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

Das folgende Beispiel zeigt den ARN für ein npm-Paket mit Bereich `@parity` und Namen `ui` im `my_repo` Repository in der Domäne`my_domain`.

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

Der ARN für ein npm-Paket ohne Gültigkeitsbereich enthält die leere Zeichenfolge für das Namespace-Feld. Das Folgende ist beispielsweise der ARN für ein Paket ohne Geltungsbereich und mit einem Namen `react` im `my_repo` Repository in der Domäne`my_domain`.

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

Die folgende Richtlinie gewährt dem Konto die `123456789012` Berechtigung, Versionen von `@parity/ui` im `my_repo` Repository zu veröffentlichen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:PublishPackageVersion"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "arn:aws:codeartifact:region-id:111122223333:package/my_domain/my_repo/npm/parity/ui"
        }
    ]
}
```

> **Important**
>
> Um die Erlaubnis zur Veröffentlichung von Maven- und NuGet Paketversionen zu erteilen, fügen Sie zusätzlich zu `codeartifact:PublishPackageVersion` die folgenden Berechtigungen hinzu.
>
> 1. NuGet: `codeartifact:ReadFromRepository` und geben Sie die Repository-Ressource an
> 2. Maven: `codeartifact:PutPackageMetadata`

Da diese Richtlinie eine Domain und ein Repository als Teil der Ressource festlegt, erlaubt sie das Veröffentlichen nur, wenn sie an dieses Repository angehängt sind.

Gewähren Sie Schreibzugriff auf ein Repository

Sie können Platzhalter verwenden, um Schreibberechtigungen für alle Pakete in einem Repository zu erteilen. Verwenden Sie beispielsweise die folgende Richtlinie, um einem Konto die Berechtigung zu erteilen, in alle Pakete im `my_repo` Repository zu schreiben.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:PublishPackageVersion"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "arn:aws:codeartifact:region-
id:111122223333:package/my_domain/my_repo/*"
        }
    ]
}
```

Interaktion zwischen Repository- und Domain-Richtlinien

CodeArtifact unterstützt Ressourcenrichtlinien für Domänen und Repositorys. Ressourcenrichtlinien sind optional. Jede Domäne kann über eine Richtlinie verfügen und jedes Repository in der Domäne kann über eine eigene Repository-Richtlinie verfügen. Wenn sowohl eine Domänenrichtlinie als auch eine Repository-Richtlinie vorhanden sind, werden beide bei der Entscheidung, ob eine Anfrage an ein CodeArtifact Repository zugelassen oder abgelehnt wird, bewertet. Domänen- und Repository-Richtlinien werden anhand der folgenden Regeln bewertet:

- Bei der Ausführung von Vorgängen auf Kontoebene wie ListDomainsoder werden keine Ressourcenrichtlinien bewertet. ListRepositories
- Bei Operationen auf Domänenebene wie oder werden keine Repository-Richtlinien bewertet. DescribeDomainListRepositoriesInDomain
- Die Domänenrichtlinie wird bei der Ausführung nicht bewertet. PutDomainPermissionsPolicy Beachten Sie, dass diese Regel Aussperrungen verhindert.
- Die Domänenrichtlinie wird bei der Ausführung bewertet PutRepositoryPermissionsPolicy, die Repository-Richtlinie wird jedoch nicht bewertet.
- Eine ausdrückliche Ablehnung in einer Richtlinie hat Vorrang vor einer Zulassung in einer anderen Richtlinie.
- Eine ausdrückliche Zulassung ist nur in einer Ressourcenrichtlinie erforderlich. Das Auslassen einer Aktion in einer Repository-Richtlinie führt nicht zu einer impliziten Ablehnung, wenn die Domänenrichtlinie die Aktion zulässt.
- Wenn keine Ressourcenrichtlinie eine Aktion zulässt, ist das Ergebnis eine implizite Ablehnung, es sei denn, das Konto des aufrufenden Prinzipals ist das Konto des Domänenbesitzers oder des Repository-Administrators und eine identitätsbasierte Richtlinie erlaubt die Aktion.

Ressourcenrichtlinien sind optional, wenn sie verwendet werden, um Zugriff in einem Szenario mit einem einzigen Konto zu gewähren, in dem das für den Zugriff auf ein Repository verwendete Anruferkonto mit dem Domäneninhaber- und Repository-Administratorkonto identisch ist. Ressourcenrichtlinien sind erforderlich, um Zugriff in einem kontenübergreifenden Szenario zu gewähren, in dem das Konto des Anrufers nicht mit dem Konto des Domaininhabers oder des Repository-Administrators identisch ist. Der kontenübergreifende Zugriff CodeArtifact folgt den allgemeinen IAM-Regeln für den kontoübergreifenden Zugriff, wie im IAM-Benutzerhandbuch unter Feststellen, ob eine kontoübergreifende Anfrage zulässig ist beschrieben.

- Einem Principal im Domaininhaberkonto kann über eine identitätsbasierte Richtlinie Zugriff auf jedes Repository in der Domain gewährt werden. Beachten Sie, dass in diesem Fall keine ausdrückliche Genehmigung in einer Domänen- oder Repository-Richtlinie erforderlich ist.
- Einem Prinzipal im Domaininhaberkonto kann über eine Domain- oder Repository-Richtlinie Zugriff auf jedes Repository gewährt werden. Beachten Sie, dass in diesem Fall keine ausdrückliche Genehmigung in einer identitätsbasierten Richtlinie erforderlich ist.
- Einem Prinzipal im Administratorkonto des Repositorys kann über eine identitätsbasierte Richtlinie Zugriff auf das Repository gewährt werden. Beachten Sie, dass in diesem Fall keine ausdrückliche Genehmigung in einer Domänen- oder Repository-Richtlinie erforderlich ist.
- Einem Prinzipal in einem anderen Konto wird nur dann Zugriff gewährt, wenn dies durch mindestens eine Ressourcenrichtlinie und mindestens eine identitätsbasierte Richtlinie erlaubt wird, ohne dass eine Richtlinie die Aktion ausdrücklich verweigert.

Kennzeichnen Sie ein Repository in CodeArtifact

Tags sind mit AWS-Ressourcen verknüpfte Schlüssel-Wert-Paare. Sie können Tags auf Ihre Repositorys in CodeArtifact anwenden. Informationen zur Kennzeichnung von CodeArtifact Ressourcen, zu Anwendungsfällen, Einschränkungen für Tagschlüssel und -werte sowie zu unterstützten Ressourcentypen finden Sie unter. Taggen von -Ressourcen

Sie können die CLI verwenden, um Tags anzugeben, wenn Sie ein Repository erstellen. Sie können die Konsole oder CLI verwenden, um Tags hinzuzufügen oder zu entfernen und die Werte von Tags in einem Repository zu aktualisieren. Sie können jedem Repository bis zu 50 Tags hinzufügen.

Themen

- Tag-Repositorien (CLI)
- Tag-Repositorys (Konsole)

Tag-Repositorien (CLI)

Sie können die CLI verwenden, um Repository-Tags zu verwalten.

Themen

- Hinzufügen von Tags zu einem Repository (CLI)
- Tags für ein Repository anzeigen (CLI)

- Tags für ein Repository bearbeiten (CLI)
- Tags aus einem Repository entfernen (CLI)

Hinzufügen von Tags zu einem Repository (CLI)

Sie können die Konsole oder die AWS CLI zum Markieren von Repositorys verwenden.

Informationen darüber, wie Sie beim Erstellen eines Repository ein Tag hinzufügen können, finden Sie unter Erstellen eines -Repositorys.

Bei diesen Schritten wird davon ausgegangen, dass Sie bereits eine aktuelle Version der AWS CLI installiert oder eine Aktualisierung auf die aktuelle Version vorgenommen haben. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface.

Führen Sie am Terminal oder über die Befehlszeile den Befehl tag-resource aus und geben Sie dabei den Amazon-Ressourcennamen (ARN) des Repositorys an, für den Sie Tags hinzufügen möchten, sowie den Schlüssel und Wert des hinzuzufügenden Tags.

> **Note**
>
> Führen Sie den `describe-repository` folgenden Befehl aus, um den ARN des Repositorys abzurufen:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Sie können mehrere Tags zu einem Repository hinzufügen. Um beispielsweise ein *my_repo* in einer Domain benanntes Repository *my_domain* mit zwei Tags zu taggen, einem Tag-Schlüssel *key1* mit dem Tag-Wert von *value1* und einem Tag-Schlüssel *key2* mit dem Tag-Wert von*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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags für ein Repository anzeigen (CLI)

Gehen Sie wie folgt vor AWS CLI , um die AWS Tags für ein Repository anzuzeigen. Wenn keine Tags hinzugefügt wurden, ist die zurückgegebene Liste leer.

Führen Sie am Terminal oder über die Befehlszeile den Befehl list-tags-for-resource aus.

> **Note**
>
> Führen Sie den `describe-repository` folgenden Befehl aus, um den ARN des Repositorys abzurufen:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Um beispielsweise eine Liste von Tag-Schlüsseln und Tag-Werten für ein Repository anzuzeigen, das *my_repo* in einer Domain benannt ist, die *my_domain* mit dem `arn:aws:codeartifact:`*us-west-2*`:`*111122223333*`:repository/`*my_domain*`/`*my_repo* ARN-Wert benannt ist:

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

Bei erfolgreicher Ausführung gibt dieser Befehl etwa wie folgt aussehende Informationen zurück:

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

Tags für ein Repository bearbeiten (CLI)

Gehen Sie wie folgt vor, AWS CLI um mit dem ein Tag für ein Repository zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen.

Führen Sie im Terminal oder in der Befehlszeile den tag-resource Befehl aus und geben Sie den ARN des Repositorys an, in dem Sie ein Tag aktualisieren möchten, und geben Sie den Tag-Schlüssel und den Tag-Wert an.

> **Note**
>
> Führen Sie den `describe-repository` folgenden Befehl aus, um den ARN des Repositorys abzurufen:
>
> ```
> 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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags aus einem Repository entfernen (CLI)

Gehen Sie wie folgt vor, AWS CLI um mit dem ein Tag aus einem Repository zu entfernen.

> **Note**
>
> Wenn Sie ein -Repository löschen, werden alle Tag-Zuordnungen aus dem Repository gelöscht. Sie müssen vor dem Löschen eines Repositorys keine Tags entfernen.

Führen Sie den Befehl im Terminal oder in der untag-resource Befehlszeile aus und geben Sie den ARN des Repositorys an, in dem Sie Tags entfernen möchten, und den Tag-Schlüssel des Tags, das Sie entfernen möchten.

> **Note**
>
> Führen Sie den `describe-repository` folgenden Befehl aus, um den ARN des Repositorys abzurufen:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Um beispielsweise mehrere Tags aus einem Repository zu entfernen, das *my_repo* in einer Domain benannt ist, die *my_domain* mit den Tag-Schlüsseln benannt ist, *key1* und*key2*:

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

Bei Erfolg hat dieser Befehl keine Ausgabe. Nach dem Entfernen von Tags können Sie mit dem `list-tags-for-resource` Befehl die verbleibenden Tags im Repository anzeigen.

Tag-Repositorys (Konsole)

Sie können Ressourcen über die Konsole oder CLI mit Tags markieren.

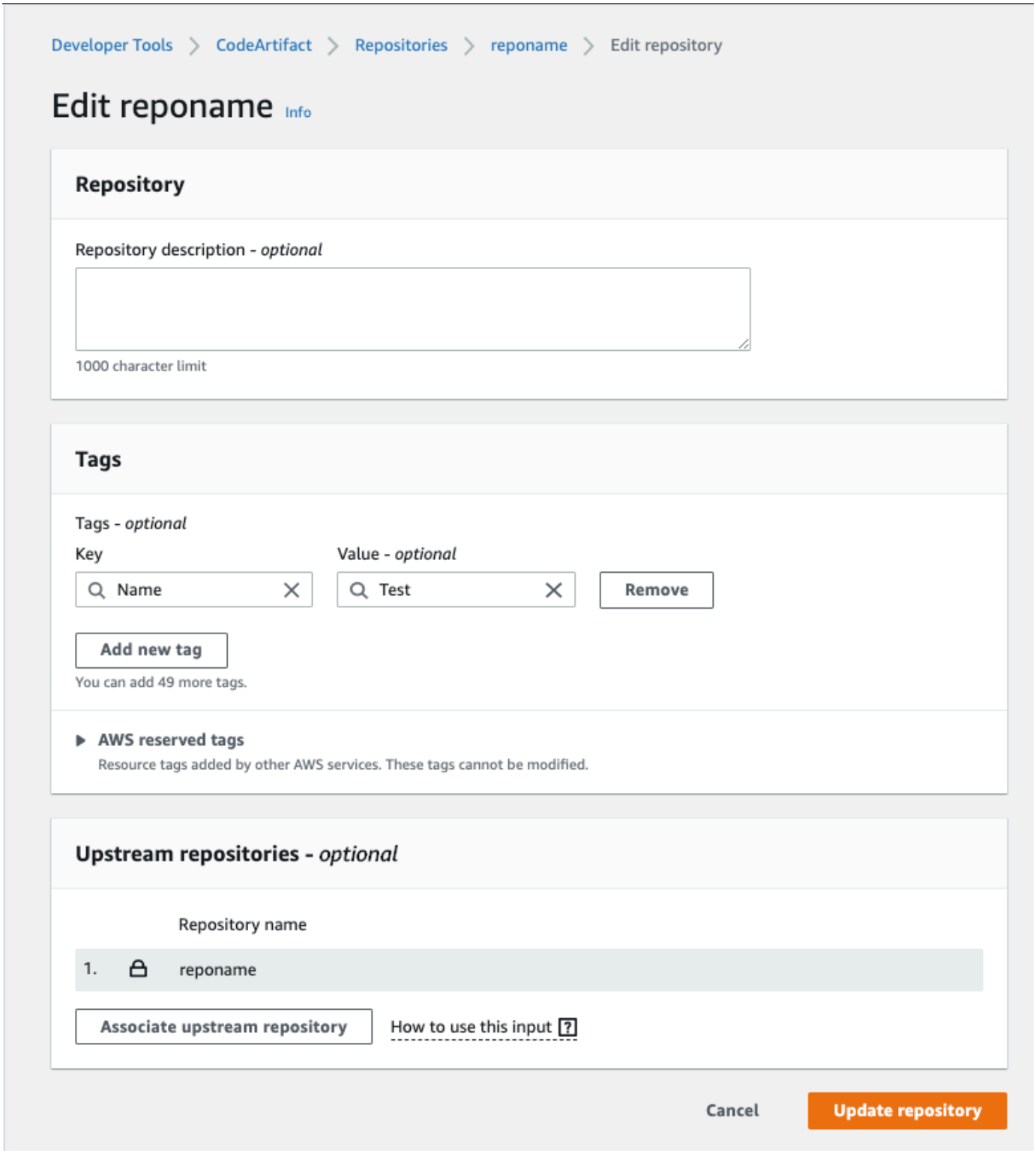
Themen

- Fügen Sie Tags zu einem Repository hinzu (Konsole)
- Tags für ein Repository anzeigen (Konsole)
- Bearbeiten Sie die Tags für ein Repository (Konsole)
- Entferne Tags aus einem Repository (Konsole)

Fügen Sie Tags zu einem Repository hinzu (Konsole)

Sie können die Konsole verwenden, um Tags zu einem vorhandenen Repository hinzuzufügen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Repositorys das Repository aus, dem Sie Tags hinzufügen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wenn das Repository keine Tags enthält, wählen Sie unter Repository-Tags die Option Repository-Tags hinzufügen aus. Wenn das Repository Tags enthält, wählen Sie Repository-Tags anzeigen und bearbeiten aus.
5. Wählen Sie Neues Tag hinzufügen aus.
6. Geben Sie in den Feldern Schlüssel und Wert den Text für jedes Tag ein, das Sie hinzufügen möchten. (Das Feld Value (Wert) ist optional.) Geben Sie beispielsweise für Key (Schlüssel) **`Name`** ein. Geben Sie unter Value (Wert) **`Test`** ein.



7. (Optional) Wählen Sie Add tag (Tag hinzufügen) aus, um weitere Zeilen hinzuzufügen und weitere Tags einzugeben.
8. Wählen Sie Repository aktualisieren.

Tags für ein Repository anzeigen (Konsole)

Sie können die Konsole verwenden, um Tags für bestehende Repositorys aufzulisten.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Repositorys das Repository aus, in dem Sie Tags anzeigen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Repository-Tags die Option Repository-Tags anzeigen und bearbeiten aus.

> **Note**
>
> Wenn diesem Repository keine Tags hinzugefügt wurden, wird in der Konsole Repository-Tags hinzufügen angezeigt.

Bearbeiten Sie die Tags für ein Repository (Konsole)

Sie können die Konsole verwenden, um Tags zu bearbeiten, die dem Repository hinzugefügt wurden.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Repositorys das Repository aus, in dem Sie Tags aktualisieren möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Repository-Tags die Option Repository-Tags anzeigen und bearbeiten aus.

> **Note**
>
> Wenn diesem Repository keine Tags hinzugefügt wurden, wird in der Konsole Repository-Tags hinzufügen angezeigt.

5. Aktualisieren Sie in den Feldern Key (Schlüssel) und Value (Wert) die Werte nach Bedarf. Ändern Sie beispielsweise für den Schlüssel **`Name`** unter Value (Wert) die Angabe **`Test`** in **`Prod`**.
6. Wählen Sie Repository aktualisieren.

Entferne Tags aus einem Repository (Konsole)

Sie können die Konsole verwenden, um Tags aus Repositorys zu löschen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Repositorys das Repository aus, in dem Sie Tags entfernen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Repository-Tags die Option Repository-Tags anzeigen und bearbeiten aus.

   > **Note**
   >
   > Wenn diesem Repository keine Tags hinzugefügt wurden, wird in der Konsole Repository-Tags hinzufügen angezeigt.

5. Wählen Sie neben dem Schlüssel und dem Wert für jedes Tag, das Sie löschen möchten, die Option Entfernen aus.
6. Wählen Sie Repository aktualisieren aus.

Arbeiten mit Upstream-Repositorys in CodeArtifact

Ein Repository kann andere AWS CodeArtifact Repositorys als Upstream-Repositorys haben. Dadurch kann ein Paketmanager-Client über einen einzigen Repository-Endpunkt auf die Pakete zugreifen, die in mehr als einem Repository enthalten sind.

Mit dem SDK, oder können Sie einem AWS CodeArtifact Repository ein oder mehrere Upstream-Repositorys hinzufügen. AWS Management Console AWS CLI Um ein Repository mit einem Upstream-Repository zu verknüpfen, benötigen Sie die entsprechenden Rechte für die `AssociateWithDownstreamRepository` Aktion im Upstream-Repository. Weitere Informationen erhalten Sie unter Erstellen Sie ein Repository mit einem Upstream-Repository und Upstream-Repositorys hinzufügen oder entfernen.

Wenn ein Upstream-Repository eine externe Verbindung zu einem öffentlichen Repository hat, können die Repositorys, die diesem nachgelagert sind, Pakete aus diesem öffentlichen Repository abrufen. Nehmen wir zum Beispiel an, dass das Repository `my_repo` ein Upstream-Repository mit dem Namen `upstream` `upstream` hat und dass es eine externe Verbindung zu einem öffentlichen NPM-Repository hat. In diesem Fall `my_repo` kann ein Paketmanager, mit dem eine Verbindung besteht, Pakete aus dem öffentlichen NPM-Repository abrufen. Weitere Hinweise zum Anfordern von Paketen aus Upstream-Repositorys oder externen Verbindungen finden Sie unter Eine Paketversion mit Upstream-Repositorys anfordern oder. Pakete von externen Verbindungen anfordern

Themen

- Was ist der Unterschied zwischen Upstream-Repositorys und externen Verbindungen?
- Upstream-Repositorys hinzufügen oder entfernen
- Ein CodeArtifact Repository mit einem öffentlichen Repository Connect
- Eine Paketversion mit Upstream-Repositorys anfordern
- Pakete von externen Verbindungen anfordern
- Reihenfolge der Prioritäten des Upstream-Reposit
- API-Verhalten bei Upstream-Repositorys

Was ist der Unterschied zwischen Upstream-Repositorys und externen Verbindungen?

In CodeArtifact: Upstream-Repositorys und externe Verbindungen verhalten sich größtenteils gleich, es gibt jedoch einige wichtige Unterschiede.

1. Sie können einem Repository bis zu 10 Upstream-Repositorys hinzufügen. CodeArtifact Sie können nur eine externe Verbindung hinzufügen.
2. Es gibt separate API-Aufrufe, um ein Upstream-Repository oder eine externe Verbindung hinzuzufügen.
3. Das Verhalten bei der Aufbewahrung von Paketen ist etwas anders, da Pakete, die von Upstream-Repositorys angefordert wurden, in diesen Repositorys aufbewahrt werden. Weitere Informationen finden Sie unter Aufbewahrung von Paketen in Zwischenrepositorien.

Upstream-Repositorys hinzufügen oder entfernen

Folgen Sie den Schritten in den folgenden Abschnitten, um Upstream-Repositorys zu einem Repository hinzuzufügen oder aus einem CodeArtifact Repository zu entfernen. Weitere Informationen zu Upstream-Repositorys finden Sie unter. Arbeiten mit Upstream-Repositorys in CodeArtifact

Dieses Handbuch enthält Informationen zur Konfiguration anderer CodeArtifact Repositorys als Upstream-Repositorys. Informationen zur Konfiguration einer externen Verbindung zu öffentlichen Repositorys wie npmjs.com, Nuget Gallery, Maven Central oder PyPI finden Sie unter Hinzufügen einer externen Verbindung.

Upstream-Repositorys hinzufügen oder entfernen (Konsole)

Führen Sie die Schritte im folgenden Verfahren aus, um mithilfe der CodeArtifact Konsole ein Repository als Upstream-Repository hinzuzufügen. Hinweise zum Hinzufügen eines Upstream-Repositorys mit dem AWS CLI finden Sie unterFügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI.

So fügen Sie mithilfe der CodeArtifact Konsole ein Upstream-Repository hinzu

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.

2. Wählen Sie im Navigationsbereich Domains und dann den Domainnamen aus, der Ihr Repository enthält.
3. Wählen Sie den Namen Ihres Repositorys.
4. Wählen Sie Edit (Bearbeiten) aus.
5. Wählen Sie unter Upstream-Repositorys die Option Upstream-Repository zuordnen aus und fügen Sie das Repository hinzu, das Sie als Upstream-Repository hinzufügen möchten. Sie können nur Repositorys in derselben Domain wie Upstream-Repositorys hinzufügen.
6. Wählen Sie Repository aktualisieren.

Um ein Upstream-Repository mithilfe der CodeArtifact Konsole zu entfernen

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann den Domainnamen aus, der Ihr Repository enthält.
3. Wählen Sie den Namen Ihres Repositorys.
4. Wählen Sie Edit (Bearbeiten) aus.
5. Suchen Sie unter Upstream-Repositorys den Listeneintrag des Upstream-Repositorys, das Sie entfernen möchten, und wählen Sie Disassociate aus.

   > **Important**
   >
   > Sobald Sie ein Upstream-Repository aus einem CodeArtifact Repository entfernen, haben Paketmanager keinen Zugriff mehr auf Pakete im Upstream-Repository oder einem seiner Upstream-Repositorys.

6. Wählen Sie Repository aktualisieren.

Fügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI

Mit dem AWS Command Line Interface (AWS CLI) können Sie die CodeArtifact Upstream-Repositorys eines Repositorys hinzufügen oder entfernen. Verwenden Sie dazu den `update-repository` Befehl und geben Sie die Upstream-Repositorys mit dem `--upstreams` Parameter an.

Sie können nur Repositorys in derselben Domain wie Upstream-Repositorys hinzufügen.

Um Upstream-Repositorys hinzuzufügen ()AWS CLI

1. Falls nicht, folgen Sie den Schritten unter Einrichtung mit AWS CodeArtifact Einrichten und Konfigurieren von AWS CLI mit CodeArtifact.
2. Verwenden Sie den `aws codeartifact update-repository` Befehl mit der `--upstreams` Markierung, um Upstream-Repositorys hinzuzufügen.

> **Note**
>
> Beim Aufrufen des `update-repository` Befehls werden die vorhandenen konfigurierten Upstream-Repositorys durch die Liste der Repositorys ersetzt, die mit dem Flag bereitgestellt werden. `--upstreams` Wenn Sie Upstream-Repositorys hinzufügen und die vorhandenen behalten möchten, müssen Sie die vorhandenen Upstream-Repositorys in den Aufruf einbeziehen.

Der folgende Beispielbefehl fügt zwei Upstream-Repositorys zu einem Repository mit dem Namen hinzu`my_repo`, das sich in einer Domäne mit dem Namen befindet. `my_domain` Die Reihenfolge der Upstream-Repositorys im `--upstreams` Parameter bestimmt deren Suchpriorität, wenn ein Paket aus dem `my_repo` Repository CodeArtifact angefordert wird. Weitere Informationen finden Sie unter Reihenfolge der Prioritäten des Upstream-Reposit.

Hinweise zum Herstellen einer Verbindung zu öffentlichen, externen Repositorys wie npmjs.com oder Maven Central finden Sie unter. Ein CodeArtifact Repository mit einem öffentlichen Repository Connect

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

Die Ausgabe enthält die Upstream-Repositorys wie folgt.

```
{
       "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": []
        }
    }
```

Um ein Upstream-Repository zu entfernen ()AWS CLI

1. Falls nicht, folgen Sie den Schritten unter Einrichtung mit AWS CodeArtifact Einrichten und Konfigurieren von AWS CLI mit CodeArtifact.
2. Verwenden Sie den `update-repository` Befehl mit der `--upstreams` Markierung, um CodeArtifact Upstream-Repositorys aus einem Repository zu entfernen. Die Liste der Repositorys, die dem Befehl zur Verfügung gestellt wird, ist der neue Satz von Upstream-Repositorys für das Repository. CodeArtifact Schließen Sie bestehende Upstream-Repositorys ein, die Sie behalten möchten, und lassen Sie die Upstream-Repositorys weg, die Sie entfernen möchten.

   Um alle Upstream-Repositorys aus einem Repository zu entfernen, verwenden Sie den `update-repository` Befehl und include `--upstreams` ohne Argument. Im Folgenden werden Upstream-Repositorys aus einem Repository mit dem Namen entfernt`my_repo`, das in einer Domäne mit dem Namen enthalten ist. `my_domain`

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

   Die Ausgabe zeigt, dass die Liste von leer `upstreams` ist.

```
{
       "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": []
       }
   }
```

Ein CodeArtifact Repository mit einem öffentlichen Repository Connect

Sie können eine externe Verbindung zwischen einem CodeArtifact Repository und einem externen, öffentlichen Repository wie https://npmjs.comdem Maven Central-Repository hinzufügen. Wenn Sie dann ein Paket aus dem CodeArtifact Repository anfordern, das noch nicht im Repository vorhanden ist, kann das Paket von der externen Verbindung abgerufen werden. Dadurch ist es möglich, Open-Source-Abhängigkeiten zu nutzen, die von Ihrer Anwendung verwendet werden.

Die beabsichtigte Art CodeArtifact, externe Verbindungen zu verwenden, besteht darin, ein Repository pro Domain mit einer externen Verbindung zu einem bestimmten öffentlichen Repository zu haben. Wenn Sie beispielsweise eine Verbindung zu npmjs.com herstellen möchten, konfigurieren Sie ein Repository in Ihrer Domain mit einer externen Verbindung zu npmjs.com und konfigurieren Sie alle anderen Repositorys mit einer Upstream-Verbindung. Auf diese Weise können alle Repositorys die Pakete verwenden, die bereits von npmjs.com abgerufen wurden, anstatt sie erneut abzurufen und zu speichern.

Themen

- Stellen Sie eine Connect zu einem externen Repository her (Konsole)
- Stellen Sie eine Connect zu einem externen Repository her (CLI)
- Unterstützte Repositorys für externe Verbindungen
- Eine externe Verbindung entfernen (CLI)

Stellen Sie eine Connect zu einem externen Repository her (Konsole)

Wenn Sie die Konsole verwenden, um eine Verbindung zu einem externen Repository hinzuzufügen, passiert Folgendes:

1. Ein `-store` Repository für das externe Repository wird in Ihrer CodeArtifact Domain erstellt, falls noch keines vorhanden ist. Diese `-store` Repositorys verhalten sich wie Zwischenrepositorien zwischen Ihrem Repository und dem externen Repository und ermöglichen es Ihnen, eine Verbindung zu mehr als einem externen Repository herzustellen.
2. Das entsprechende `-store` Repository wird deinem Repository als Upstream hinzugefügt.

Die folgende Liste enthält jedes `-store` Repository CodeArtifact und das jeweilige externe Repository, mit dem sie eine Verbindung herstellen.

1. `cargo-store`ist mit crates.io verbunden.
2. `clojars-store`ist mit Clojars Repository verbunden.
3. `commonsware-store`ist mit dem CommonsWare Android Repository verbunden.
4. `google-android-store`ist mit Google Android verbunden.
5. `gradle-plugins-store`ist mit Gradle-Plugins verbunden.
6. `maven-central-store`ist mit dem Maven Central Repository verbunden.
7. `npm-store`ist mit npmjs.com verbunden.
8. `nuget-store`ist mit nuget.org verbunden.
9. `pypi-store`ist mit der Python Packaging Authority verbunden.

10`rubygems-store`ist mit RubyGems .org verbunden.

Um eine Verbindung zu einem externen Repository (Konsole) herzustellen

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann den Domainnamen aus, der Ihr Repository enthält.
3. Wählen Sie den Namen Ihres Repositorys.
4. Wählen Sie Edit (Bearbeiten) aus.

5. Wählen Sie unter Upstream-Repositorys die Option Upstream-Repository zuordnen aus und fügen Sie das entsprechende `-store` Repository hinzu, das als Upstream-Repository verbunden ist.
6. Wählen Sie Repository aktualisieren aus.

Nachdem das `-store` Repository als Upstream-Repository hinzugefügt wurde, können Paketmanager, die mit Ihrem CodeArtifact Repository verbunden sind, Pakete aus dem jeweiligen externen Repository abrufen.

Stellen Sie eine Connect zu einem externen Repository her (CLI)

Sie können das verwenden AWS CLI , um Ihr CodeArtifact Repository mit einem externen Repository zu verbinden, indem Sie eine externe Verbindung direkt zum Repository hinzufügen. Auf diese Weise können Benutzer, die mit dem CodeArtifact Repository oder einem seiner nachgelagerten Repositorys verbunden sind, Pakete aus dem konfigurierten externen Repository abrufen. Jedes CodeArtifact Repository kann nur eine externe Verbindung haben.

Es wird empfohlen, ein Repository pro Domain mit einer externen Verbindung zu einem bestimmten öffentlichen Repository zu haben. Um andere Repositorien mit dem öffentlichen Repository zu verbinden, fügen Sie ihnen das Repository mit der externen Verbindung als Upstream hinzu. Wenn Sie oder eine andere Person in Ihrer Domain bereits externe Verbindungen in der Konsole konfiguriert haben, verfügt Ihre Domain wahrscheinlich bereits über ein `-store` Repository mit einer externen Verbindung zu dem öffentlichen Repository, zu dem Sie eine Verbindung herstellen möchten. Weitere Informationen zu `-store` Repositorys und zur Verbindung mit der Konsole finden Sie unterStellen Sie eine Connect zu einem externen Repository her (Konsole).

Um eine externe Verbindung zu einem CodeArtifact Repository hinzuzufügen (CLI)

- Wird verwendet`associate-external-connection`, um eine externe Verbindung hinzuzufügen. Das folgende Beispiel verbindet ein Repository mit der öffentlichen NPM-Registry npmjs.com. Eine Liste der unterstützten externen Repositorys finden Sie unter. Unterstützte Repositorys für externe Verbindungen

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

  Beispielausgabe:

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

Nachdem Sie eine externe Verbindung hinzugefügt haben, finden Sie unter Informationen Pakete von externen Verbindungen anfordern zum Anfordern von Paketen aus einem externen Repository mit einer externen Verbindung.

Unterstützte Repositorys für externe Verbindungen

CodeArtifact unterstützt eine externe Verbindung zu den folgenden öffentlichen Repositorys. Um mit der CodeArtifact CLI eine externe Verbindung anzugeben, verwenden Sie den Wert in der Spalte Name für den `--external-connection` Parameter, wenn Sie den `associate-external-connection` Befehl ausführen.

Repository-Typ	Beschreibung	Name
Maven	Das Repository von Clojar	`public:maven-clojars`
Maven	CommonsWare Android-Repository	`public:maven-commonsware`

Repository-Typ	Beschreibung	Name
Maven	Google Android-Repository	`public:maven-googleandroid`
Maven	Repository für Gradle-Plugins	`public:maven-gradleplugins`
Maven	Maven Central	`public:maven-central`
npm	öffentliches Register von npm	`public:npmjs`
NuGet	NuGet Bildergalerie	`public:nuget-org`
Python	Python-Paketindex	`public:pypi`
Ruby	RubyGems.org	`public:ruby-gems-org`
Rust	Crates.io	`public:crates-io`

Eine externe Verbindung entfernen (CLI)

Um eine externe Verbindung zu entfernen, die mit dem `associate-external-connection` Befehl in der hinzugefügt wurde AWS CLI, verwenden Sie`disassociate-external-connection`.

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

Beispielausgabe:

```
{
    "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": []
    }
}
```

Eine Paketversion mit Upstream-Repositorys anfordern

Wenn ein Client (zum Beispiel npm) eine Paketversion von einem CodeArtifact Repository mit dem Namen anfordert`my_repo`, das mehrere Upstream-Repositorys hat, kann Folgendes passieren:

- Wenn sie die angeforderte Paketversion `my_repo` enthält, wird sie an den Client zurückgegeben.
- Wenn `my_repo` es die angeforderte Paketversion nicht enthält, CodeArtifact sucht es in `my_repo` den Upstream-Repositorys danach. Wenn die Paketversion gefunden wird, wird ein Verweis darauf `my_repo` kopiert und die Paketversion wird an den Client zurückgegeben.
- Wenn `my_repo` weder die Paketversion noch die Upstream-Repositorys die Paketversion enthalten, wird eine HTTP `Not Found` 404-Antwort an den Client zurückgegeben.

Wenn Sie Upstream-Repositorys mit dem `update-repository` Befehl `create-repository` oder hinzufügen, bestimmt die Reihenfolge, in der sie an den `--upstreams` Parameter übergeben werden, ihre Priorität, wenn eine Paketversion angefordert wird. Geben Sie die Upstream-Repositorys `--upstreams` in der Reihenfolge an, die Sie verwenden CodeArtifact möchten, wenn eine Paketversion angefordert wird. Weitere Informationen finden Sie unter Reihenfolge der Prioritäten des Upstream-Reposit.

Die maximale Anzahl von direkten Upstream-Repositorys, die für ein Repository zulässig sind, ist 10. Da Direct-Upstream-Repositorys auch eigene Direct-Upstream-Repositorys haben CodeArtifact können, können mehr als 10 Repositorien nach Paketversionen durchsucht werden. Die maximale Anzahl von Repositorys, die CodeArtifact durchsucht werden, wenn eine Paketversion angefordert wird, ist 25.

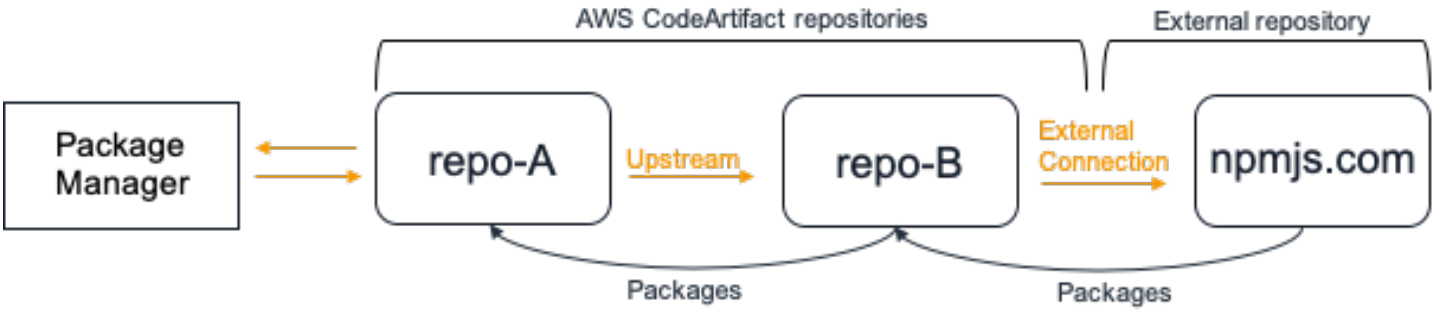
Aufbewahrung von Paketen aus Upstream-Repositorys

Wenn eine angeforderte Paketversion in einem Upstream-Repository gefunden wird, wird ein Verweis darauf beibehalten und ist im Downstream-Repository immer verfügbar. Die beibehaltene Paketversion ist von keinem der folgenden Faktoren betroffen:

- Das Upstream-Repository wird gelöscht.
- Trennen des Upstream-Repositorys vom Downstream-Repository.
- Löschen der Paketversion aus dem Upstream-Repository.
- Bearbeiten der Paketversion im Upstream-Repository (z. B. durch Hinzufügen eines neuen Assets).

Ruft Pakete über eine Upstream-Beziehung ab

Wenn ein CodeArtifact Repository eine Upstream-Beziehung zu einem Repository hat, das über eine externe Verbindung verfügt, werden Anfragen für Pakete, die sich nicht im Upstream-Repository befinden, aus dem externen Repository kopiert. Stellen Sie sich zum Beispiel die folgende Konfiguration vor: Ein Repository mit dem Namen `repo-A` hat ein Upstream-Repository mit dem Namen`repo-B`. `repo-B`hat eine externe Verbindung zu https://npmjs.com.



Wenn npm es für die Verwendung des `repo-A` Repositorys konfiguriert ist, `npm install` löst das Ausführen das Kopieren von Paketen aus dem Verzeichnis aus https://npmjs.com. `repo-B` Die installierten Versionen werden ebenfalls abgerufen`repo-A`. Das folgende Beispiel wird installiert`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
```

`repo-A`Enthält nach der Ausführung `npm install` nur die neueste Version (`lodash 4.17.20`), da dies die Version ist, die npm von `repo-A` abgerufen wurde.

```
aws codeartifact list-package-versions --repository repo-A --domain my_domain \
```

```
            --domain-owner 111122223333 --format npm --package lodash
```

Beispielausgabe:

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

Da es eine externe Verbindung zu `repo-B` hat https://npmjs.com, werden alle Paketversionen, aus https://npmjs.comdenen importiert wurde, in `repo-B` gespeichert. Diese Paketversionen könnten von jedem Downstream-Repository mit einer Upstream-Beziehung zu `repo-B` abgerufen worden sein.

Der Inhalt von `repo-B` bietet eine Möglichkeit, alle Pakete und Paketversionen zu sehen, aus denen https://npmjs.comim Laufe der Zeit importiert wurde. Um beispielsweise alle Versionen des `lodash` Pakets zu sehen, die im Laufe der Zeit importiert wurden, können Sie Folgendes verwenden`list-package-versions`.

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

Beispielausgabe:

```
{
    "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",
```

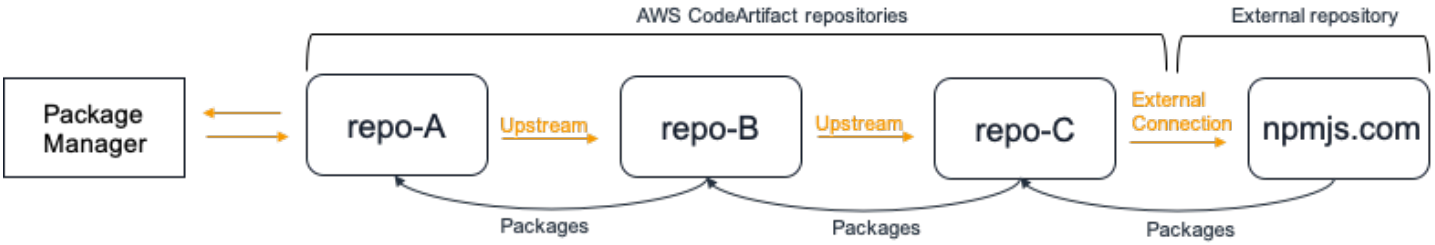
```
            "status": "Published"
        },
        {
            "version": "0.2.0",
            "revision": "REVISION-3-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.2.1",
            "revision": "REVISION-4-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.1.0",
            "revision": "REVISION-5-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        }
    ],
    "nextToken": "eyJsaXN0UGFja2FnZVZlcnNpb25zVG9rZW4iOiIwLjIuMiJ9"
}
```


Aufbewahrung von Paketen in Zwischenrepositorien

CodeArtifact ermöglicht die Verkettung von Upstream-Repositorys. `repo-A`Kann zum Beispiel `repo-B` als Upstream haben und `repo-B` kann `repo-C` als Upstream haben. Diese Konfiguration macht die Paketversionen in `repo-B` und `repo-C` verfügbar von`repo-A`.

Wenn ein Paketmanager eine Verbindung zum Repository herstellt `repo-A` und eine Paketversion aus dem Repository abruft`repo-C`, wird die Paketversion nicht im Repository gespeichert. `repo-B` Die Paketversion wird nur im Repository gespeichert, das sich am weitesten unten befindet, in diesem Beispiel. `repo-A` Sie wird nicht in zwischengeschalteten Repositorien aufbewahrt. Dies gilt auch für längere Ketten. Wenn es beispielsweise vier Repositorien`repo-A`,`repo-B`, und gäbe `repo-D` und ein Paketmanager verbunden wäre`repo-C`, von dem eine Paketversion `repo-A` abgerufen wurde`repo-D`, würde die Paketversion in oder beibehalten, `repo-A` aber nicht in oder. `repo-B` `repo-C`

Das Verhalten bei der Paketaufbewahrung ist ähnlich, wenn eine Paketversion aus einem externen Repository abgerufen wird, mit der Ausnahme, dass die Paketversion immer in dem Repository aufbewahrt wird, an das die externe Verbindung angeschlossen ist. `repo-A`Hat zum Beispiel `repo-B` als Upstream. `repo-B`hat `repo-C` als Upstream und hat `repo-C` auch npmjs.com als externe Verbindung konfiguriert; siehe das folgende Diagramm.



Wenn ein Paketmanager, der eine Verbindung herstellt, eine Paketversion **`repo-A`** anfordert, zum Beispiel Lodash 4.17.20, und die Paketversion in keinem der drei Repositorys vorhanden ist, wird sie von npmjs.com abgerufen. Wenn Lodash 4.17.20 abgerufen wird, wird es beibehalten, da es sich um das am meisten nachgeschaltete Repository handelt und **`repo-A`** da die externe Verbindung zu npmjs.com angeschlossen ist. **`repo-C`** lodash 4.17.20 wird nicht gespeichert, da es sich um ein Zwischenrepository handelt. `repo-B`

Pakete von externen Verbindungen anfordern

In den folgenden Abschnitten wird beschrieben, wie ein Paket über eine externe Verbindung angefordert wird. Außerdem wird CodeArtifact das erwartete Verhalten bei der Anforderung eines Pakets beschrieben.

Themen

- Ruft Pakete von einer externen Verbindung ab
- Latenz bei externen Verbindungen
- CodeArtifact Verhalten, wenn ein externes Repository nicht verfügbar ist
- Verfügbarkeit neuer Paketversionen
- Paketversionen mit mehr als einem Asset werden importiert

Ruft Pakete von einer externen Verbindung ab

Um Pakete von einer externen Verbindung abzurufen, nachdem Sie sie zu Ihrem CodeArtifact Repository hinzugefügt haben, wie unter beschriebenEin CodeArtifact Repository mit einem

öffentlichen Repository Connect, konfigurieren Sie Ihren Paketmanager so, dass er Ihr Repository verwendet, und installieren Sie die Pakete.

> **Note**
>
> Die folgenden Anweisungen finden Sie unternpm, oder CodeArtifact Mit Python verwenden, um die Konfiguration und Verwendung von Anweisungen für andere Pakettypen Verwendung CodeArtifact mit Maven Verwenden CodeArtifact mit NuGet einzusehen.

Um Pakete von einer externen Verbindung abzurufen

1. Konfigurieren und authentifizieren Sie Ihren Paketmanager mit Ihrem CodeArtifact Repository. Verwenden Sie für npm den folgenden `aws codeartifact login`-Befehl:

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

2. Fordere das Paket aus dem öffentlichen Repository an. Verwenden Sie für npm den folgenden `npm install` Befehl und *lodash* ersetzen Sie ihn durch das Paket, das Sie installieren möchten.

   ```
   npm install lodash
   ```

3. Nachdem das Paket in Ihr CodeArtifact Repository kopiert wurde, können Sie es mit den `list-package-versions` Befehlen `list-packages` und anzeigen.

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

   Beispielausgabe:

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

Der `list-package-versions` Befehl listet alle Versionen des Pakets auf, die in Ihr CodeArtifact Repository kopiert wurden.

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

Beispielausgabe:

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

Latenz bei externen Verbindungen

Beim Abrufen eines Pakets aus einem öffentlichen Repository über eine externe Verbindung kommt es zu einer Verzögerung zwischen dem Abrufen des Pakets aus dem öffentlichen Repository und dem Speichern in Ihrem CodeArtifact Repository. Nehmen wir zum Beispiel an, Sie haben Version 1.2.5 des npm-Pakets „lodash“ installiert, wie unter beschrieben. Ruft Pakete von einer externen Verbindung ab Obwohl der Befehl `npm install lodash` lodash erfolgreich abgeschlossen wurde, erscheint die Paketversion möglicherweise noch nicht in Ihrem Repository. CodeArtifact Normalerweise dauert es etwa 3 Minuten, bis die Paketversion in Ihrem Repository erscheint, obwohl es gelegentlich länger dauern kann.

Aufgrund dieser Latenz haben Sie möglicherweise erfolgreich eine Paketversion abgerufen, können die Version jedoch möglicherweise noch nicht in Ihrem Repository in der CodeArtifact Konsole oder beim Aufrufen der ListPackages und ListPackageVersions API-Operationen sehen. Sobald CodeArtifact die Paketversion asynchron persistiert wurde, ist sie in der Konsole und über API-Anfragen sichtbar.

CodeArtifact Verhalten, wenn ein externes Repository nicht verfügbar ist

Gelegentlich kommt es in einem externen Repository zu einem Ausfall, was bedeutet, dass Pakete CodeArtifact nicht abgerufen werden können oder dass das Abrufen von Paketen viel langsamer als normal ist. In diesem Fall stehen Paketversionen, die bereits aus einem externen Repository (z. B. npmjs.com) abgerufen und in einem CodeArtifact Repository gespeichert wurden, weiterhin zum Herunterladen zur Verfügung. CodeArtifact Pakete, die noch nicht gespeichert sind, sind jedoch CodeArtifact möglicherweise nicht verfügbar, selbst wenn eine externe Verbindung zu diesem Repository konfiguriert wurde. Ihr CodeArtifact Repository könnte beispielsweise die npm-Paketversion enthalten, `lodash 4.17.19` da Sie diese bisher in Ihrer Anwendung verwendet haben. Wenn Sie ein Upgrade durchführen möchten`4.17.20`, CodeArtifact wird diese neue Version normalerweise von npmjs.com abgerufen und in Ihrem Repository gespeichert. CodeArtifact Wenn bei npmjs.com jedoch ein Ausfall auftritt, ist diese neue Version nicht verfügbar. Die einzige Problemumgehung besteht darin, es später erneut zu versuchen, sobald npmjs.com wiederhergestellt ist.

Ausfälle externer Repositorys können sich auch auf die Veröffentlichung neuer Paketversionen auswirken. CodeArtifact In einem Repository mit konfigurierter externer Verbindung ist CodeArtifact die Veröffentlichung einer Paketversion, die bereits im externen Repository vorhanden ist, nicht möglich. Weitere Informationen finden Sie unter Überblick über Pakete. In seltenen Fällen kann ein Ausfall eines externen Repositorys jedoch bedeuten, dass CodeArtifact keine up-to-date Informationen darüber vorliegen, welche Pakete und Paketversionen in einem externen Repository vorhanden sind. In diesem Fall CodeArtifact könnte es möglich sein, eine Paketversion zu veröffentlichen, die normalerweise verweigert würde.

Verfügbarkeit neuer Paketversionen

Damit eine Paketversion in einem öffentlichen Repository wie npmjs.com über ein CodeArtifact Repository verfügbar ist, muss sie zuerst einem regionalen Paketmetadaten-Cache hinzugefügt werden. Dieser Cache wird CodeArtifact in jeder AWS Region verwaltet und enthält Metadaten, die den Inhalt der unterstützten öffentlichen Repositorien beschreiben. Aufgrund dieses Caches gibt es eine Verzögerung zwischen der Veröffentlichung einer neuen Paketversion in einem öffentlichen Repository und dem Zeitpunkt, an dem CodeArtifact sie verfügbar ist. Diese Verzögerung variiert je nach Pakettyp.

Bei npm-, Python- und Nuget-Paketen kann es zu einer Verzögerung von bis zu 30 Minuten kommen, wenn eine neue Paketversion auf npmjs.com, pypi.org oder nuget.org veröffentlicht wird und sie aus einem Repository installiert werden kann. CodeArtifact CodeArtifact synchronisiert automatisch

Metadaten aus diesen beiden Repositorys, um sicherzustellen, dass der Cache auf dem neuesten Stand ist.

Bei Maven-Paketen kann es zu einer Verzögerung von bis zu 3 Stunden zwischen der Veröffentlichung einer neuen Paketversion in einem öffentlichen Repository und dem Zeitpunkt kommen, an dem sie aus einem Repository installiert werden kann. CodeArtifact CodeArtifact sucht höchstens einmal alle 3 Stunden nach neuen Versionen eines Pakets. Die erste Anfrage nach einem bestimmten Paketnamen nach Ablauf der 3-stündigen Cache-Lebensdauer führt dazu, dass alle neuen Versionen dieses Pakets in den regionalen Cache importiert werden.

Bei häufig verwendeten Maven-Paketen werden neue Versionen normalerweise alle 3 Stunden importiert, da aufgrund der hohen Anzahl von Anfragen der Cache häufig aktualisiert wird, sobald die Cache-Lebensdauer abgelaufen ist. Bei selten verwendeten Paketen enthält der Cache erst dann die neueste Version, wenn eine Version des Pakets aus einem CodeArtifact Repository angefordert wird. Bei der ersten Anfrage sind nur zuvor importierte Versionen verfügbar CodeArtifact, aber diese Anfrage führt dazu, dass der Cache aktualisiert wird. Bei nachfolgenden Anfragen werden die neuen Versionen des Pakets dem Cache hinzugefügt und stehen zum Download zur Verfügung.

Paketversionen mit mehr als einem Asset werden importiert

Sowohl Maven- als auch Python-Pakete können mehrere Assets pro Paketversion haben. Dies macht das Importieren von Paketen dieser Formate komplexer als das von npm und NuGet Paketen, die nur ein Asset pro Paketversion haben. Eine Beschreibung der Elemente, die für diese Pakettypen importiert werden und wie neu hinzugefügte Elemente verfügbar gemacht werden, finden Sie unter und. Python-Pakete von Upstreams und externen Verbindungen anfordern Maven-Pakete von Upstreams und externen Verbindungen anfordern

Reihenfolge der Prioritäten des Upstream-Reposit

Wenn Sie eine Paketversion aus einem Repository mit einem oder mehreren Upstream-Repositorys anfordern, entspricht deren Priorität der Reihenfolge, in der sie beim Aufruf des Befehls `create-repository` or `update-repository` aufgeführt wurden. Wenn die angeforderte Paketversion gefunden wird, wird die Suche beendet, auch wenn nicht alle Upstream-Repositorys durchsucht wurden. Weitere Informationen finden Sie unter Fügen Sie Upstream-Repositorys hinzu oder entfernen Sie sie ()AWS CLI.

Verwenden Sie den `describe-repository` Befehl, um die Prioritätsreihenfolge zu sehen.

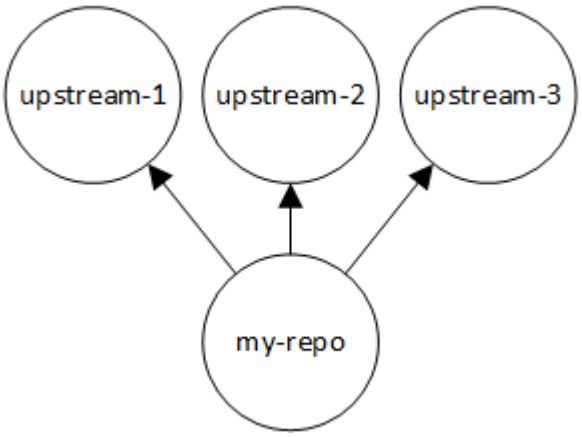
```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-
owner 111122223333
```

Das Ergebnis könnte das Folgende sein. Es zeigt, dass die Priorität des Upstream-Repositorys an `upstream-1` erster, `upstream-2` zweiter und `upstream-3` dritter Stelle liegt.

```
{
    "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": []
    }
}
```

Einfaches Beispiel für die Prioritätsreihenfolge

In der folgenden Abbildung hat das `my_repo` Repository drei Upstream-Repositorys. Die Prioritätsreihenfolge der Upstream-Repositorys ist`upstream-1`,,`upstream-2`. `upstream-3`



Eine Anfrage nach einer Paketversion in `my_repo` durchsucht die Repositorys in der folgenden Reihenfolge, bis sie gefunden wird oder bis eine HTTP `Not Found` 404-Antwort an den Client zurückgegeben wird:

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

Wenn die Paketversion gefunden wird, stoppt die Suche, auch wenn sie nicht in allen Upstream-Repositorys gesucht hat. Wenn die Paketversion beispielsweise in gefunden wird`upstream-1`, stoppt die Suche und sucht CodeArtifact nicht in `upstream-2` oder`upstream-3`.

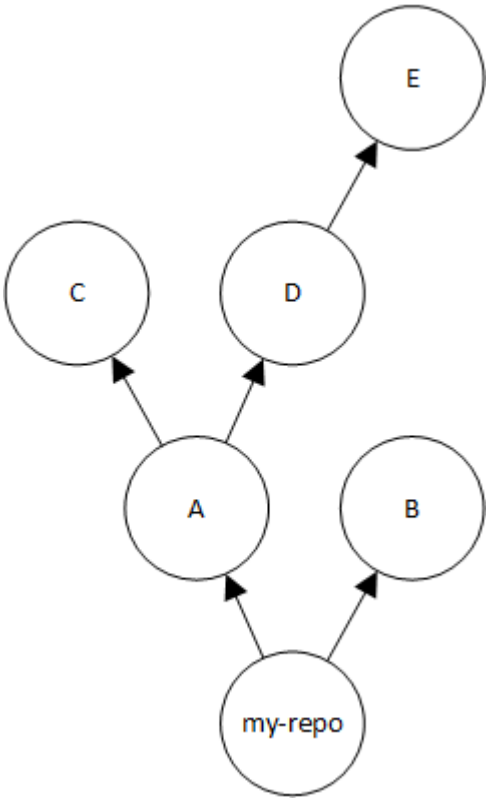
Wenn Sie den AWS CLI Befehl verwenden`list-package-versions`, um Paketversionen aufzulisten`my_repo`, sucht er nur in`my_repo`. Es listet keine Paketversionen in Upstream-Repositorys auf.

Beispiel für eine komplexe Prioritätsreihenfolge

Wenn ein Upstream-Repository seine eigenen Upstream-Repositorys hat, wird dieselbe Logik verwendet, um eine Paketversion zu finden, bevor zum nächsten Upstream-Repository gewechselt wird. Nehmen wir zum Beispiel an, dass Ihr `my_repo` Repository zwei Upstream-Repositorys hat, und`A`. `B` Wenn das Projektarchiv `A` über Upstream-Repositorys verfügt, sucht eine Anfrage nach einer Paketversion `my_repo` zuerst in`A`, dann in den Upstream-Repositorys von `A` usw. `my_repo`

In der folgenden Abbildung enthält das Repository `my_repo` Upstream-Repositorys. Das Upstream-Repository `A` hat zwei Upstream-Repositorys und `D` ein Upstream-Repository. Upstream-Repositorys

auf derselben Ebene im Diagramm werden in ihrer Prioritätsreihenfolge von links nach rechts angezeigt (Repository `A` hat eine höhere Prioritätsreihenfolge als Repository `B` und Repository `C` hat eine höhere Prioritätsreihenfolge als Repository`D`).



In diesem Beispiel `my_repo` durchsucht eine Anfrage nach einer Paketversion in den Repositorys in der folgenden Reihenfolge, bis sie gefunden wird oder bis ein Paketmanager eine HTTP `Not Found` 404-Antwort an den Client zurückgibt:

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

API-Verhalten bei Upstream-Repositorys

Wenn Sie bestimmte CodeArtifact APIs On-Repositorys aufrufen, die mit Upstream-Repositorys verbunden sind, kann das Verhalten unterschiedlich sein, je nachdem, ob die Pakete oder

Paketversionen im Ziel-Repository oder im Upstream-Repository gespeichert sind. Das Verhalten dieser APIs ist hier dokumentiert.

Weitere Informationen dazu CodeArtifact APIs finden Sie in der CodeArtifact API-Referenz.

Die meisten APIs , die auf ein Paket oder eine Paketversion verweisen, geben einen `ResourceNotFound` Fehler zurück, wenn die angegebene Paketversion nicht im Ziel-Repository vorhanden ist. Dies gilt auch dann, wenn das Paket oder die Paketversion in einem Upstream-Repository vorhanden ist. Tatsächlich werden Upstream-Repositorys ignoriert, wenn sie aufgerufen werden. APIs Das APIs sind:

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

Um dieses Verhalten zu demonstrieren, haben wir zwei Repositorien: `target-repo` und`upstream-repo`. `target-repo`ist leer und wurde als Upstream-Repository `upstream-repo` konfiguriert. `upstream-repo`enthält das npm-Paket`lodash`.

Wenn wir die `DescribePackageVersion` API aufrufen`upstream-repo`, die das `lodash` Paket enthält, erhalten wir die folgende Ausgabe:


```
{
    "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+FilPs=",
        "status": "Published"
    }
```

Wenn wir dieselbe API aufrufen`target-repo`, die leer ist, aber als Upstream `upstream-repo` konfiguriert wurde, erhalten wir die folgende Ausgabe:

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

Die `CopyPackageVersions` API verhält sich anders. Standardmäßig kopiert die `CopyPackageVersions` API nur Paketversionen, die im Ziel-Repository gespeichert sind. Wenn eine Paketversion im Upstream-Repository, aber nicht im Ziel-Repository gespeichert ist, wird sie nicht kopiert. Um Paketversionen von Paketen einzubeziehen, die nur im Upstream-Repository gespeichert sind, setzen Sie `true` in Ihrer API-Anfrage den Wert auf. `includeFromUpstream`

Weitere Informationen zur `CopyPackageVersions` API finden Sie unterPakete zwischen Repositorys kopieren.

Arbeiten Sie mit Paketen in CodeArtifact

In den folgenden Themen erfahren Sie, wie Sie mithilfe der CodeArtifact CLI und der API Aktionen für Pakete ausführen.

Themen

- Überblick über Pakete
- Listet Paketnamen auf
- Paketversionen auflisten
- Listet die Ressourcen der Paketversion auf
- Laden Sie die Ressourcen der Paketversion herunter
- Pakete zwischen Repositorys kopieren
- Löschen Sie ein Paket oder eine Paketversion
- Details und Abhängigkeiten der Paketversion anzeigen und aktualisieren
- Aktualisiere den Status der Paketversion
- Die Einstellungen zur Herkunft des Pakets werden bearbeitet

Überblick über Pakete

Ein Paket ist ein Softwarepaket und die Metadaten, die zur Auflösung von Abhängigkeiten und zur Installation der Software erforderlich sind. In CodeArtifact besteht ein Paket aus einem Paketnamen, einem optionalen Namespace wie `@types` in`@types/node`, einer Reihe von Paketversionen und Metadaten auf Paketebene wie npm-Tags.

Inhalt

- Unterstützte Paketformate
- Veröffentlichen von Paketen
  - Berechtigungen für das Veröffentlichen
  - Paket-Assets überschreiben
  - Private Pakete und öffentliche Repositorys
  - Veröffentlichen gepatchter Paketversionen
  - Beschränkungen der Asset-Größe für die Veröffentlichung

  - Latenz bei der Veröffentlichung
- Status der Paketversion
- Normalisierung von Paketnamen, Paketversion und Assetnamen

Unterstützte Paketformate

AWS CodeArtifact unterstützt die Paketformate Cargo, Generic, Maven, npm NuGet, PyPI, Ruby und Swift.

Veröffentlichen von Paketen

Sie können neue Versionen jedes unterstützten Paketformats in einem CodeArtifact Repository veröffentlichen, indem Sie Tools wie`npm`,`twine`,`Maven`, `Gradle`nuget, und verwenden`dotnet`.

Berechtigungen für das Veröffentlichen

Ihr AWS Identity and Access Management (IAM-) Benutzer oder Ihre Rolle muss über Berechtigungen zum Veröffentlichen im Ziel-Repository verfügen. Die folgenden Berechtigungen sind erforderlich, um Pakete zu veröffentlichen:

- Fracht: `codeartifact:PublishPackageVersion`
- generisch: `codeartifact:PublishPackageVersion`
- Maven: `codeartifact:PublishPackageVersion` und `codeartifact:PutPackageMetadata`
- npm: `codeartifact:PublishPackageVersion`
- NuGet: `codeartifact:PublishPackageVersion` und `codeartifact:ReadFromRepository`
- Python: `codeartifact:PublishPackageVersion`
- Rubin: `codeartifact:PublishPackageVersion`
- Schnell: `codeartifact:PublishPackageVersion`

In der obigen Liste von Berechtigungen muss Ihre IAM-Richtlinie die `package` Ressource für die `codeartifact:PutPackageMetadata` Berechtigungen `codeartifact:PublishPackageVersion` und angeben. Sie muss auch die `repository` Ressource für die `codeartifact:ReadFromRepository` Berechtigung angeben.

Weitere Hinweise zu Berechtigungen in CodeArtifact finden Sie unterAWS CodeArtifact Referenz zu Berechtigungen.

Paket-Assets überschreiben

Sie können ein bereits vorhandenes Paket-Asset mit einem anderen Inhalt nicht erneut veröffentlichen. Nehmen wir beispielsweise an, Sie haben bereits ein Maven-Paket mit einem JAR-Asset veröffentlicht. `mypackage-1.0.jar` Sie können dieses Asset nur dann erneut veröffentlichen, wenn die Prüfsumme der alten und neuen Assets identisch ist. Um dasselbe Asset mit neuem Inhalt erneut zu veröffentlichen, löschen Sie zuerst die Paketversion, indem Sie den delete-package-versions Befehl verwenden. Der Versuch, denselben Asset-Namen mit anderem Inhalt erneut zu veröffentlichen, führt zu einem HTTP-409-Konfliktfehler.

Für Paketformate, die mehrere Assets unterstützen (Generic, PyPI und Maven), können Sie einer vorhandenen Paketversion neue Assets mit unterschiedlichen Namen hinzufügen, vorausgesetzt, Sie verfügen über die erforderlichen Berechtigungen. Bei generischen Paketen können Sie neue Assets hinzufügen, solange sich die Paketversion im Status befindet. `Unfinished` Da npm nur ein einzelnes Asset pro Paketversion unterstützt, müssen Sie, um eine veröffentlichte Paketversion auf irgendeine Weise zu ändern, diese zuerst mit dem folgenden Befehl delete-package-versions löschen.

Wenn Sie versuchen, ein bereits vorhandenes Objekt erneut zu veröffentlichen (z. B.`mypackage-1.0.jar`) und der Inhalt des veröffentlichten Elements und des neuen Elements identisch sind, ist der Vorgang erfolgreich, da der Vorgang idempotent ist.

Private Pakete und öffentliche Repositorys

CodeArtifact veröffentlicht keine in Repositorys gespeicherten Pakete in öffentlichen CodeArtifact Repositorys wie npmjs.com oder Maven Central. CodeArtifact importiert Pakete aus öffentlichen Repositorys in ein CodeArtifact Repository, verschiebt Pakete jedoch nie in die andere Richtung. Pakete, die Sie in CodeArtifact Repositorys veröffentlichen, bleiben privat und stehen nur den AWS Konten, Rollen und Benutzern zur Verfügung, denen Sie Zugriff gewährt haben.

Veröffentlichen gepatchter Paketversionen

Manchmal möchten Sie vielleicht eine modifizierte Paketversion veröffentlichen, möglicherweise eine, die in einem öffentlichen Repository verfügbar ist. Möglicherweise haben Sie einen Fehler in einer kritischen Anwendungsabhängigkeit namens gefundenmydep 1.1, und Sie müssen ihn beheben, bevor der Paketanbieter die Änderung überprüfen und akzeptieren kann. Wie bereits beschrieben,

CodeArtifact verhindert es, dass Sie `mydep 1.1` in Ihrem CodeArtifact Repository veröffentlichen, wenn das öffentliche Repository von Ihrem CodeArtifact Repository aus über Upstream-Repositorys und eine externe Verbindung erreichbar ist.

Um dieses Problem zu umgehen, veröffentlichen Sie die Paketversion in einem anderen CodeArtifact Repository, in dem das öffentliche Repository nicht erreichbar ist. Verwenden Sie dann die `copy-package-versions` API, um die gepatchte Version von in das CodeArtifact Repository `mydep 1.1` zu kopieren, von wo aus Sie sie verwenden werden.

Beschränkungen der Asset-Größe für die Veröffentlichung

Die maximale Größe eines Paket-Assets, das veröffentlicht werden kann, wird durch das maximale Kontingent für die Asset-Dateigröße begrenzt, das unter angezeigt wirdKontingente in AWS CodeArtifact. Sie können beispielsweise kein Maven-JAR- oder Python-Rad veröffentlichen, das das maximale Kontingent für die aktuelle Asset-Dateigröße überschreitet. Wenn Sie größere Assets speichern müssen CodeArtifact, fordern Sie eine Erhöhung des Kontingents an.

Zusätzlich zum maximalen Kontingent für die Größe der Asset-Datei beträgt die maximale Größe einer Veröffentlichungsanforderung für NPM-Pakete 2 GB. Dieses Limit ist unabhängig vom maximalen Kontingent für die Größe der Asset-Datei und kann nicht durch eine Erhöhung des Kontingents erhöht werden. In einer NPM-Veröffentlichungsanforderung (HTTP PUT) werden Paketmetadaten und der Inhalt des NPM-Paket-Tar-Archivs gebündelt. Aus diesem Grund variiert die tatsächliche maximale Größe eines npm-Pakets, das veröffentlicht werden kann, und hängt von der Größe der enthaltenen Metadaten ab.

> **Note**
>
> Veröffentlichte npm-Pakete sind auf eine maximale Größe von weniger als 2 GB beschränkt.

Latenz bei der Veröffentlichung

In einem CodeArtifact Repository veröffentlichte Paketversionen können häufig in weniger als einer Sekunde heruntergeladen werden. Wenn Sie beispielsweise eine npm-Paketversion in CodeArtifact with veröffentlichen`npm publish`, sollte diese Version in weniger als einer Sekunde für einen `npm install` Befehl verfügbar sein. Die Veröffentlichung kann jedoch inkonsistent sein und manchmal länger dauern. Wenn Sie unmittelbar nach der Veröffentlichung eine Paketversion verwenden müssen, versuchen Sie es erneut, um sicherzustellen, dass der Download zuverlässig

ist. Wiederholen Sie den Download beispielsweise nach dem Veröffentlichen der Paketversion bis zu dreimal, wenn die gerade veröffentlichte Paketversion beim ersten Download-Versuch zunächst nicht verfügbar ist.

> **Note**
>
> Das Importieren einer Paketversion aus einem öffentlichen Repository dauert in der Regel länger als das Veröffentlichen. Weitere Informationen finden Sie unter Latenz bei externen Verbindungen.

Status der Paketversion

Jede Paketversion in CodeArtifact hat einen Status, der den aktuellen Status und die Verfügbarkeit der Paketversion beschreibt. Sie können den Status der Paketversion im SDK AWS CLI und SDK ändern. Weitere Informationen finden Sie unter Aktualisiere den Status der Paketversion.

Die folgenden Werte sind für den Paketversionsstatus möglich:

- Veröffentlicht — Die Paketversion wurde erfolgreich veröffentlicht und kann mit einem Paketmanager angefordert werden. Die Paketversion wird in die Paketversionslisten aufgenommen, die an die Paketmanager zurückgegeben werden, z. B. in der Ausgabe von`npm view <package-name> versions`. Alle Ressourcen der Paketversion sind im Repository verfügbar.
- Unfertig — Der Client hat ein oder mehrere Elemente für eine Paketversion hochgeladen, sie aber noch nicht fertiggestellt, indem er sie in den `Published` Status verschoben hat. Derzeit können nur generische Versionen und Maven-Paketversionen den Status haben. `Unfinished` Bei Maven-Paketen kann dies der Fall sein, wenn der Client ein oder mehrere Assets für eine Paketversion hochlädt, aber keine `maven-metadata.xml` Datei für das Paket veröffentlicht, die diese Version enthält. Wenn eine Maven-Paketversion unfertig ist, wird sie nicht in Versionslisten aufgenommen, die an solche `mvn` Clients zurückgegeben werden`gradle`, sodass sie nicht als Teil eines Builds verwendet werden kann. Generische Pakete können bewusst im `Unfinished` Status belassen werden, indem das `unfinished` Flag beim Aufruf der PublishPackageVersionAPI angegeben wird. Ein generisches Paket kann in den `Published` Status geändert werden, indem das `unfinished` Flag weggelassen oder die API aufgerufen wird. UpdatePackageVersionsStatus
- Nicht gelistet — Die Inhalte der Paketversion stehen im Repository zum Herunterladen zur Verfügung, aber die Paketversion ist nicht in der Liste der Versionen enthalten, die an

die Paketmanager zurückgegeben werden. Bei einem npm-Paket enthält die Ausgabe von beispielsweise nicht `npm view <package-name> versions` die Paketversion. Dies bedeutet, dass die Logik zur Abhängigkeitsauflösung von npm die Paketversion nicht auswählt, da die Version nicht in der Liste der verfügbaren Versionen erscheint. Wenn jedoch bereits in einer `npm package-lock.json` Datei auf die Paketversion „Nicht gelistet“ verwiesen wird, kann sie dennoch heruntergeladen und installiert werden, z. B. während der Ausführung. `npm ci`

- Archiviert — Die Ressourcen der Paketversion können nicht mehr heruntergeladen werden. Die Paketversion wird nicht in die Liste der Versionen aufgenommen, die an die Paketmanager zurückgegeben werden. Da die Ressourcen nicht verfügbar sind, wird die Nutzung der Paketversion durch Clients blockiert. Wenn der Build Ihrer Anwendung von einer Version abhängt, die auf Archiviert aktualisiert wurde, schlägt der Build fehl, vorausgesetzt, die Paketversion wurde nicht lokal zwischengespeichert. Sie können einen Paketmanager oder ein Build-Tool nicht verwenden, um eine archivierte Paketversion erneut zu veröffentlichen, da sie immer noch im Repository vorhanden ist. Sie können jedoch den Status der Paketversion mit der API wieder auf Nicht gelistet oder Veröffentlicht ändern. UpdatePackageVersionsStatus
- Verworfen — Die Paketversion erscheint nicht in den Auflistungen und die Inhalte können nicht aus dem Repository heruntergeladen werden. Der Hauptunterschied zwischen „Verworfen“ und „Archiviert“ besteht darin, dass bei einem Status von „Verworfen“ die Inhalte der Paketversion dauerhaft von gelöscht werden CodeArtifact. Aus diesem Grund können Sie eine Paketversion nicht von „Verworfen“ in „Archiviert“, „Nicht gelistet“ oder „Veröffentlicht“ verschieben. Die Paketversion kann nicht mehr verwendet werden, da die Assets gelöscht wurden. Nachdem eine Paketversion als entsorgt markiert wurde, wird Ihnen die Speicherung der Paketressourcen nicht mehr in Rechnung gestellt.

Paketversionen aller Status werden standardmäßig zurückgegeben, wenn sie list-package-versions ohne `--status` Parameter aufgerufen werden.

Abgesehen von den zuvor aufgeführten Status kann eine Paketversion auch mit der DeletePackageVersionsAPI gelöscht werden. Nach dem Löschen befindet sich eine Paketversion nicht mehr im Repository und Sie können diese Paketversion mit einem Paketmanager oder einem Build-Tool nach Belieben erneut veröffentlichen. Nachdem eine Paketversion gelöscht wurde, wird Ihnen die Speicherung der Inhalte dieser Paketversion nicht mehr in Rechnung gestellt.

Normalisierung von Paketnamen, Paketversion und Assetnamen

CodeArtifact normalisiert Paketnamen, Paketversionen und Assetnamen, bevor sie gespeichert werden, was bedeutet, dass sich die Namen oder Versionen in denen CodeArtifact möglicherweise

von dem Namen oder der Version unterscheiden, die bei der Veröffentlichung des Pakets angegeben wurden. Weitere Informationen darüber, wie Namen und Versionen CodeArtifact für jeden Pakettyp normalisiert werden, finden Sie in der folgenden Dokumentation:

- Normalisierung von Python-Paketnamen
- NuGet Normalisierung von Paketnamen, Version und Assetnamen

CodeArtifact führt keine Normalisierung für andere Paketformate durch.

Listet Paketnamen auf

Verwenden Sie den `list-packages` Befehl in CodeArtifact , um eine Liste aller Paketnamen in einem Repository abzurufen. Dieser Befehl gibt nur die Paketnamen zurück, nicht die Versionen.

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

Beispielausgabe:

```
{
    "nextToken": "eyJidWNrZXRJZCI6I...",
    "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"
            }
        },
        {
            "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"
            }
        }
    ]
}
```

NPM-Paketnamen auflisten

Um nur die Namen der npm-Pakete aufzulisten, setzen Sie den Wert der `--format` Option auf. `npm`

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

Um npm-Pakete in einem Namespace (npm-Bereich) aufzulisten, verwenden Sie die Optionen und. `--namespace` `--format`

> **Important**
>
> Der Wert für die `--namespace` Option sollte den Anfang nicht enthalten. `@` Um nach dem Namespace zu suchen`@types`, setzen Sie den Wert auf*types*.

> **Note**
>
> Die `--namespace` Option filtert nach dem Namespace-Präfix. Jedes npm-Paket mit einem Bereich, der mit dem an die `--namespace` Option übergebenen Wert beginnt, wird in der `list-packages` Antwort zurückgegeben.

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

Beispielausgabe:

```
{
    "nextToken": "eyJidWNrZXRJZ...",
    "packages": [
        {
            "package": "3d-bin-packing",
            "namespace": "types",
            "format": "npm"

        },
        {
            "package": "a-big-triangle",
            "namespace": "types",
            "format": "npm"
```

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

        }
    ]
}
```

Listet die Namen der Maven-Pakete auf

Um nur die Namen der Maven-Pakete aufzulisten, setzen Sie den Wert der `--format` Option auf. `maven` Sie müssen in der Option auch die Maven-Gruppen-ID angeben. `--namespace`

> **Note**
>
> Die `--namespace` Option filtert nach dem Namespace-Präfix. Jedes npm-Paket mit einem Bereich, der mit dem an die `--namespace` Option übergebenen Wert beginnt, wird in der `list-packages` Antwort zurückgegeben.

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

Beispielausgabe:

```
{
    "nextToken": "eyJidWNrZXRJZ...",
    "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"

        }
    ]
}
```

Python-Paketnamen auflisten

Um nur die Namen von Python-Paketen aufzulisten, setzen Sie den Wert der `--format` Option auf`pypi`.

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

Filtern Sie nach dem Präfix des Paketnamens

Um Pakete zurückzugeben, die mit einer bestimmten Zeichenfolge beginnen, können Sie die `--package-prefix` Option verwenden.

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

Beispielausgabe:

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

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

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

        }
    ]
}
```

Unterstützte Kombinationen von Suchoptionen

Sie können die `--package-prefix` Optionen `--format``--namespace`, und in jeder beliebigen Kombination verwenden, außer dass diese nicht alleine verwendet werden `--namespace` kann. Für die Suche nach allen npm-Paketen mit einem Bereich, der mit 1 beginnt, `@types` muss die `--format` Option angegeben werden. Die `--namespace` alleinige Verwendung führt zu einem Fehler.

Die Verwendung keiner der drei Optionen wird auch von unterstützt `list-packages` und gibt alle Pakete aller Formate zurück, die im Repository vorhanden sind.

Ausgabe formatieren

Sie können Parameter verwenden, die für alle AWS CLI Befehle verfügbar sind, um die `list-packages` Antwort kompakter und lesbarer zu gestalten. Verwenden Sie den `--query` Parameter, um das Format jeder zurückgegebenen Paketversion anzugeben. Verwenden Sie den `--output` Parameter, um die Antwort als Klartext zu formatieren.

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

Beispielausgabe:

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

Weitere Informationen finden Sie unter Steuern der Befehlsausgabe über die AWS CLI im AWS Command Line Interface -Benutzerhandbuch.

Standardwerte und andere Optionen

Standardmäßig beträgt die maximale Anzahl von Ergebnissen, die von zurückgegeben werden`list-packages`, 100. Sie können dieses Ergebnislimit ändern, indem Sie die `--max-results` Option verwenden.

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

Der zulässige Höchstwert von `--max-results` ist 1.000. Um das Auflisten von Paketen in Repositorys mit mehr als 1.000 Paketen zu ermöglichen, wird `list-packages` die Paginierung mithilfe des `nextToken` Felds in der Antwort unterstützt. Wenn die Anzahl der Pakete im Repository den Wert von übersteigt`--max-results`, können Sie den Wert von an einen anderen Aufruf von übergeben, `nextToken` um die nächste Ergebnisseite `list-packages` zu erhalten.

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

Paketversionen auflisten

Verwenden Sie den `list-package-versions` Befehl in AWS CodeArtifact , um eine Liste aller Versionen eines Paketnamens in einem Repository abzurufen.

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

Beispielausgabe:

```
{
  "defaultDisplayVersion": "1.0.1",
  "format": "npm",
  "package": "kind-of",
  "versions": [
      {
          "version": "1.0.1",
```

```
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "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"
              }
        }
    ]
}
```

Sie können dem `list-package-versions` Aufruf den `--status` Parameter hinzufügen, um die Ergebnisse nach dem Status der Paketversion zu filtern. Weitere Hinweise zum Status der Paketversion finden Sie unterStatus der Paketversion.

Sie können die Antwort `list-package-versions` mithilfe der `--next-token` Parameter `--max-results` und paginieren. Geben Sie für `--max-results` eine Ganzzahl zwischen 1 und 1000 an, um die Anzahl der auf einer einzelnen Seite zurückgegebenen Ergebnisse anzugeben. Die Standardeinstellung ist 50. Um nachfolgende Seiten zurückzugeben, führen Sie den `list-package-versions` Vorgang erneut aus und übergeben Sie den in der vorherigen Befehlsausgabe empfangenen `nextToken` Wert an`--next-token`. Wenn die `--next-token` Option nicht verwendet wird, wird immer die erste Ergebnisseite zurückgegeben.

Der `list-package-versions` Befehl listet keine Paketversionen in Upstream-Repositorys auf. Es werden jedoch Verweise auf Paketversionen in einem Upstream-Repository aufgeführt, die während einer Paketversionsanforderung in Ihr Repository kopiert wurden. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.

NPM-Paketversionen auflisten

Um alle Paketversionen für ein npm-Paket aufzulisten, setzen Sie den Wert der `--format` Option auf. `npm`

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

Verwenden Sie die Option, um npm-Paketversionen in einem bestimmten Namespace (npm-Bereich) aufzulisten. `--namespace` Der Wert für die `--namespace` Option sollte den Anfang nicht enthalten. `@` Um nach dem Namespace zu suchen`@types`, setzen Sie den Wert auf`types`.

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

Maven-Paketversionen auflisten

Um alle Paketversionen für ein Maven-Paket aufzulisten, setzen Sie den Wert der `--format` Option auf. `maven` Sie müssen in der Option auch die Maven-Gruppen-ID angeben. `--namespace`

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

Versionen sortieren

`list-package-versions`kann Versionen ausgeben, die nach dem Zeitpunkt der Veröffentlichung in absteigender Reihenfolge sortiert sind (die zuletzt veröffentlichten Versionen werden zuerst aufgeführt). Verwenden Sie den `--sort-by` Parameter mit einem Wert von `PUBLISHED_TIME` wie folgt.

```
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
```

Beispielausgabe:

```
{

  "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-3-654S65A5C5E1FC",
        "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...."
}
```

Standardanzeigeversion

Der Rückgabewert für `defaultDisplayVersion` hängt vom Paketformat ab:

- Für generische, Maven- und PyPI-Pakete ist dies die zuletzt veröffentlichte Paketversion.
- Bei npm-Paketen ist dies die Version, auf die das Tag verweist. `latest` Wenn das `latest` Tag nicht gesetzt ist, handelt es sich um die zuletzt veröffentlichte Paketversion.

Ausgabe formatieren

Sie können Parameter verwenden, die für alle AWS CLI Befehle verfügbar sind, um die `list-package-versions` Antwort kompakter und lesbarer zu gestalten. Verwenden Sie den `--query` Parameter, um das Format jeder zurückgegebenen Paketversion anzugeben. Verwenden Sie den `--output` Parameter, um die Antwort als Klartext zu formatieren.

```
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]'
```

Beispielausgabe:

```
0.1.1
0.1.2
0.1.0
3.0.0
```

Weitere Informationen finden Sie AWS CLI im AWS Command Line Interface Benutzerhandbuch unter Steuern der Befehlsausgabe.

Listet die Ressourcen der Paketversion auf

Ein Asset ist eine einzelne Datei (z. B. eine `.tgz` NPM-Datei oder eine Maven-POM- oder JAR-Datei) CodeArtifact , die in einer Paketversion gespeichert ist. Sie können den `list-package-version-assets` Befehl verwenden, um die Assets in jeder Paketversion aufzulisten.

Führen Sie den `list-package-version-assets` Befehl aus, um die folgenden Informationen zu jedem Asset in Ihrem AWS Konto und Ihrer aktuellen AWS Region zurückzugeben:

- Sein Name.
- Seine Größe in Byte.
- Eine Reihe von Hashwerten, die für die Prüfsummenvalidierung verwendet werden.

Verwenden Sie beispielsweise den folgenden Befehl, um die Ressourcen des Python-Pakets`flatten-json`, Version, aufzulisten`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
```

Nachfolgend sehen Sie die Ausgabe.

```
{
    "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": "43f24850b7b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
                "SHA-512":
 "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fced7bd1e80a0dca9ce320d95f086
SHA-512"
            }
        },
        {
            "name": "flatten_json-0.1.7.tar.gz",
            "size": 2865,
            "hashes": {
                "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",
                "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",
                "SHA-256": "43f24850b7b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
                "SHA-512":
 "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fced7bd1e80a0dca9ce320d95f086
SHA-512"
            }
        }
    ]
}
```

Listet die Assets eines NPM-Pakets auf

Ein npm-Paket hat immer ein einzelnes Asset mit dem Namen. `package.tgz` Um die Ressourcen eines NPM-Pakets mit Gültigkeitsbereich aufzulisten, schließen Sie den Bereich in die Option ein. `--namespace`

```
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
```

Listet die Ressourcen eines Maven-Pakets auf

Um die Ressourcen eines Maven-Pakets aufzulisten, fügen Sie den Paket-Namespace in die Option ein. `--namespace` Um die Ressourcen des Maven-Pakets aufzulisten: `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
```

Laden Sie die Ressourcen der Paketversion herunter

Ein Asset ist eine einzelne Datei (z. B. eine `.tgz` NPM-Datei oder eine Maven-POM- oder JAR-Datei) CodeArtifact , die in einer Paketversion gespeichert ist. Sie können Paket-Assets mit dem herunterladen. `get-package-version-assets command` Auf diese Weise können Sie Assets abrufen, ohne einen Paketmanager-Client wie `npm` oder zu verwenden`pip`. Um ein Asset herunterzuladen, müssen Sie den Namen des Assets angeben, den Sie mit dem `list-package-version-assets` Befehl abrufen können. Weitere Informationen finden Sie unterListet die Ressourcen der Paketversion auf. Das Asset wird mit einem von Ihnen angegebenen Dateinamen in den lokalen Speicher heruntergeladen.

Im folgenden Beispiel wird das *`guava-27.1-jre.jar`* Asset aus dem Maven-Paket *`com.google.guava:guava`* mit Version *`27.1-jre`* heruntergeladen.

```
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
```

In diesem Beispiel wurde der Dateiname wie *`guava-27.1-jre.jar`* durch das letzte Argument im vorherigen Befehl angegeben, sodass das heruntergeladene Asset benannt *`guava-27.1-jre.jar`* wird.

Die Ausgabe des Befehls wird wie folgt aussehen:

```
{
```

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

> **Note**
>
> Um Assets aus einem NPM-Paket mit Gültigkeitsbereich herunterzuladen, schließen Sie den Bereich in die Option ein. `--namespace` Das `@` Symbol muss bei der Verwendung weggelassen werden. `--namespace` Wenn der Gültigkeitsbereich beispielsweise lautet`@types`, verwenden Sie`--namespace types`.

Für das Herunterladen von Inhalten mithilfe von `get-package-version-asset` ist eine `codeartifact:GetPackageVersionAsset` Genehmigung für die Paketressource erforderlich. Weitere Informationen zu ressourcenbasierten Berechtigungsrichtlinien finden Sie im Benutzerhandbuch unter Ressourcenbasierte Richtlinien.AWS Identity and Access Management

Pakete zwischen Repositorys kopieren

Sie können Paketversionen von einem Repository in ein anderes kopieren CodeArtifact. Dies kann für Szenarien wie Workflows zur Förderung von Paketen oder die gemeinsame Nutzung von Paketversionen zwischen Teams oder Projekten hilfreich sein. Das Quell- und das Ziel-Repository müssen sich in derselben Domäne befinden, um Paketversionen kopieren zu können.

Erforderliche IAM-Berechtigungen zum Kopieren von Paketen

Zum Kopieren von Paketversionen muss der aufrufende Benutzer über die erforderlichen IAM-Berechtigungen verfügen CodeArtifact, und die ressourcenbasierte Richtlinie, die mit den Quell- und Ziel-Repositorys verknüpft ist, muss über die erforderlichen Berechtigungen verfügen. Weitere Informationen zu ressourcenbasierten Berechtigungsrichtlinien und Repositorys finden Sie unter. CodeArtifact Repository-Richtlinien

Der Benutzer, der anruft, `copy-package-versions` muss über die `ReadFromRepository` Berechtigungen für das Quell-Repository und über die `CopyPackageVersions` Berechtigung für das Ziel-Repository verfügen.

Das Quell-Repository muss über die `ReadFromRepository` entsprechende Berechtigung verfügen und das Ziel-Repository muss über die `CopyPackageVersions` Berechtigung verfügen, die dem

IAM-Konto oder dem Benutzer zugewiesen wurde, der Pakete kopiert. Die folgenden Richtlinien sind Beispiele für Repository-Richtlinien, die mit dem `put-repository-permissions-policy` Befehl zum Quell-Repository oder Ziel-Repository hinzugefügt werden können. Ersetze es *111122223333* durch die ID des Accounts, der anruft`copy-package-versions`.

> **Note**
>
> Durch den Aufruf `put-repository-permissions-policy` wird die aktuelle Repository-Richtlinie ersetzt, falls eine existiert. Sie können den `get-repository-permissions-policy` Befehl verwenden, um zu überprüfen, ob eine Richtlinie existiert. Weitere Informationen finden Sie unterLesen Sie eine Richtlinie. Wenn eine Richtlinie existiert, sollten Sie ihr diese Berechtigungen hinzufügen, anstatt sie zu ersetzen.

Beispiel für eine Berechtigungsrichtlinie für das Quell-Repository

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

Beispiel für eine Berechtigungsrichtlinie für ein Ziel-Repository

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

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

Kopieren Sie die Paketversionen

Verwenden Sie den `copy-package-versions` Befehl in CodeArtifact , um eine oder mehrere Paketversionen von einem Quell-Repository in ein Ziel-Repository in derselben Domäne zu kopieren. Im folgenden Beispiel werden die Versionen 6.0.2 und 4.0.0 eines npm-Pakets mit dem Namen `my-package` aus dem Repository in das `my_repo` Repository kopiert. `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
```

Sie können mehrere Versionen desselben Paketnamens in einem einzigen Vorgang kopieren. Um Versionen verschiedener Paketnamen zu kopieren, müssen Sie `copy-package-versions` jede Version aufrufen.

Der vorherige Befehl erzeugt die folgende Ausgabe, vorausgesetzt, beide Versionen konnten erfolgreich kopiert werden.

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

```
}
```

Kopiert ein Paket aus Upstream-Repositorys

Normalerweise sucht es `copy-package-versions` nur in dem mit der `--source-repository` Option angegebenen Repository nach zu kopierenden Versionen. Sie können jedoch Versionen sowohl aus dem Quell-Repository als auch aus seinen Upstream-Repositorys kopieren, indem Sie die `--include-from-upstream` Option verwenden. Wenn Sie das CodeArtifact SDK verwenden, rufen Sie die `CopyPackageVersions` API auf, wobei der `includeFromUpstream` Parameter auf true gesetzt ist. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.

Kopieren Sie ein NPM-Paket mit Gültigkeitsbereich

Um eine NPM-Paketversion in einen Bereich zu kopieren, verwenden Sie die `--namespace` Option, um den Bereich anzugeben. Um das Paket beispielsweise zu kopieren`@types/react`, verwenden Sie`--namespace types`. Das `@` Symbol muss bei der Verwendung weggelassen werden`--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
```

Kopieren Sie die Maven-Paketversionen

Um Maven-Paketversionen zwischen Repositorys zu kopieren, geben Sie das zu kopierende Paket an, indem Sie die Maven-Gruppen-ID mit der `--namespace` Option und die Maven-ArtifactID mit der Option übergeben. `--name` Um beispielsweise eine einzelne Version von zu kopieren: `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
```

Wenn die Paketversion erfolgreich kopiert wurde, sieht die Ausgabe wie folgt aus.

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

Versionen, die im Quell-Repository nicht existieren

Wenn Sie eine Version angeben, die im Quell-Repository nicht existiert, schlägt der Kopiervorgang fehl. Wenn einige Versionen im Quell-Repository existieren und andere nicht, schlägt das Kopieren aller Versionen fehl. Im folgenden Beispiel ist Version 0.2.0 des `array-unique` npm-Pakets im Quell-Repository vorhanden, Version 5.6.7 jedoch nicht:

```
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
```

Die Ausgabe in diesem Szenario wird der folgenden ähneln.

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

Der `SKIPPED` Fehlercode wird verwendet, um anzuzeigen, dass die Version nicht in das Ziel-Repository kopiert wurde, weil eine andere Version nicht kopiert werden konnte.

Versionen, die bereits im Ziel-Repository vorhanden sind

Wenn eine Paketversion in ein Repository kopiert wird, in dem sie bereits existiert, werden ihre Paketressourcen und Metadaten auf Paketversionsebene in den beiden Repositorys CodeArtifact verglichen.

Wenn die Ressourcen und Metadaten der Paketversion in den Quell- und Ziel-Repositorys identisch sind, wird kein Kopiervorgang durchgeführt, der Vorgang wird jedoch als erfolgreich angesehen. Das bedeutet, dass es `copy-package-versions` idempotent ist. In diesem Fall wird die Version, die bereits im Quell- und im Ziel-Repository vorhanden war, in der Ausgabe von nicht aufgeführt. `copy-package-versions`

Im folgenden Beispiel `array-unique` sind zwei Versionen des npm-Pakets im Quell-Repository vorhanden. `repo-1` Version 0.2.1 ist auch im Ziel-Repository vorhanden `dest-repo` und Version 0.2.0 nicht.

```
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
```

Die Ausgabe in diesem Szenario wird der folgenden ähneln.

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

Version 0.2.0 ist in aufgeführt`successfulVersions`, weil sie erfolgreich vom Quell- in das Ziel-Repository kopiert wurde. Version 0.2.1 wird in der Ausgabe nicht angezeigt, da sie bereits im Ziel-Repository vorhanden war.

Wenn sich die Inhalte oder Metadaten der Paketversion in den Quell- und Ziel-Repositorys unterscheiden, schlägt der Kopiervorgang fehl. Sie können den `--allow-overwrite` Parameter verwenden, um ein Überschreiben zu erzwingen.

Wenn einige Versionen im Ziel-Repository existieren und andere nicht, können alle Versionen nicht kopiert werden. Im folgenden Beispiel ist Version 0.3.2 des `array-unique` npm-Pakets sowohl im Quell- als auch im Ziel-Repository vorhanden, der Inhalt der Paketversion ist jedoch unterschiedlich. Version 0.2.1 ist im Quell-Repository vorhanden, aber nicht im Ziel-Repository.

```
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
```

Die Ausgabe in diesem Szenario wird der folgenden ähneln.

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

Version 0.2.1 ist als gekennzeichnet`SKIPPED`, weil sie nicht in das Ziel-Repository kopiert wurde. Sie wurde nicht kopiert, weil die Kopie von Version 0.3.2 fehlschlug, weil sie bereits im Ziel-Repository vorhanden war, aber im Quell- und Ziel-Repository nicht identisch war.

Angabe einer Paketversionsrevision

Eine Paketversionsrevision ist eine Zeichenfolge, die einen bestimmten Satz von Elementen und Metadaten für eine Paketversion angibt. Sie können eine Paketversionsrevision angeben, um Paketversionen zu kopieren, die sich in einem bestimmten Status befinden. Um eine Paketversionsrevision anzugeben, verwenden Sie den `--version-revisions` Parameter, um eine oder mehrere durch Kommas getrennte Paketversionen und die Revisionspaare der Paketversion an den `copy-package-versions` Befehl zu übergeben.

> **Note**
>
> Sie müssen den `--versions` oder den `--version-revisions` Parameter mit angeben. `copy-package-versions` Sie können nicht beides angeben.

Im folgenden Beispiel wird Version 0.3.2 des Pakets nur kopiert, `my-package` wenn es im Quell-Repository mit der Paketversionsrevision `REVISION-1-SAMPLE-6C81EFF7DA55CC` vorhanden ist.

```
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
```

Das folgende Beispiel kopiert zwei Versionen des Paketsmy-package, 0.3.2 und 0.3.13. Das Kopieren ist nur erfolgreich, wenn im Quell-Repository Version 0.3.2 von Revision `REVISION-1-SAMPLE-6C81EFF7DA55CC` und Version `my-package` 0.3.13 Revision hat. `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
```

Um die Revisionen einer Paketversion zu finden, verwenden Sie den Befehl oder den `describe-package-version` Befehl. `list-package-versions`

Weitere Informationen finden Sie unter Revision der Paketversion und CopyPackageVersion in der CodeArtifact -API-Referenz.

Kopiere npm-Pakete

Weitere Informationen zum `copy-package-versions` Verhalten mit npm-Paketen finden Sie unter npm-Tags und in der API. CopyPackageVersions

Löschen Sie ein Paket oder eine Paketversion

Mit dem `delete-package-versions` Befehl können Sie eine oder mehrere Paketversionen gleichzeitig löschen. Um ein Paket vollständig aus einem Repository zu entfernen, einschließlich aller zugehörigen Versionen und Konfigurationen, verwenden Sie den `delete-package` Befehl. Ein Paket kann in einem Repository ohne Paketversionen existieren. Dies kann passieren, wenn alle Versionen mit dem `delete-package-versions` Befehl gelöscht wurden oder wenn das Paket mithilfe der `put-package-origin-configuration` API-Operation ohne Versionen erstellt wurde (sieheDie Einstellungen zur Herkunft des Pakets werden bearbeitet).

Themen

- Löschen eines Pakets (AWS CLI)
- Löschen eines Pakets (Konsole)
- Löschen einer Paketversion (AWS CLI)
- Löschen einer Paketversion (Konsole)
- Löschen eines NPM-Pakets oder einer Paketversion
- Löschen eines Maven-Pakets oder einer Paketversion
- Bewährte Methoden zum Löschen von Paketen oder Paketversionen

Löschen eines Pakets (AWS CLI)

Mit dem `delete-package` Befehl können Sie ein Paket einschließlich aller Paketversionen und Konfigurationen löschen. Das folgende Beispiel löscht das PyPI-Paket, das im Repo `my-package` `my_repo` in der Domäne benannt ist: `my_domain`

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

Beispielausgabe:

```
{
   "deletedPackage": {
      "format": "pypi",
      "originConfiguration": {
         "restrictions": {
            "publish": "ALLOW",
```

```
            "upstream": "BLOCK"
         }
      },
      "package": "my-package"
   }
}
```

Sie können bestätigen, dass das Paket gelöscht wurde, indem Sie es `describe-package` für denselben Paketnamen ausführen:

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

Löschen eines Pakets (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im linken Navigationsbereich Repositorys aus.
3. Wählen Sie das Repository aus, aus dem Sie ein Paket löschen möchten.
4. Wählen Sie das Package aus, das Sie löschen möchten.
5. Wählen Sie Package löschen.

Löschen einer Paketversion (AWS CLI)

Mit dem `delete-package-versions` Befehl können Sie eine oder mehrere Paketversionen gleichzeitig löschen. Das folgende Beispiel löscht die Versionen `4.0.0` `4.0.1`, und `5.0.0` des PyPI-Pakets, das `my-package` `my_repo` in der Domäne benannt ist: `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
```

Beispielausgabe:

```
{
   "successfulVersions": {
```

```
      "4.0.0": {
         "revision": "oxwwYC9dDeuBoCt6+PDSwL6OMZ7rXeiXy44BM32Iawo=",
            "status": "Deleted"
      },
      "4.0.1": {
         "revision": "byaaQR748wrsdBaT+PDSwL6OMZ7rXeiBKM0551aqWmo=",
            "status": "Deleted"
      },
      "5.0.0": {
         "revision": "yubm34QWeST345ts+ASeioPI354rXeiSWr734PotwRw=",
            "status": "Deleted"
      }
   },
   "failedVersions": {}
}
```

Sie können überprüfen, ob die Versionen gelöscht wurden, indem Sie die Ausführung `list-package-versions` für denselben Paketnamen ausführen:

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

Löschen einer Paketversion (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im linken Navigationsbereich Repositorys aus.
3. Wählen Sie das Repository aus, aus dem Sie Paketversionen löschen möchten.
4. Wählen Sie das Package aus, aus dem Sie Versionen löschen möchten.
5. Wählen Sie die Paketversion aus, die Sie löschen möchten.
6. Wählen Sie Löschen.

> **Note**
>
> In der Konsole können Sie jeweils nur eine Paketversion löschen. Verwenden Sie die CLI, um mehrere gleichzeitig zu löschen.

Löschen eines NPM-Pakets oder einer Paketversion

Um ein npm-Paket oder einzelne Paketversionen zu löschen, setzen Sie die `--format` Option auf. `npm` Um eine Paketversion in einem NPM-Paket mit Gültigkeitsbereich zu löschen, verwenden Sie die `--namespace` Option, um den Bereich anzugeben. Um das Paket `@types/react` beispielsweise zu löschen, verwenden Sie. `--namespace types` Lassen Sie das `@` Symbol weg, wenn Sie verwenden`--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
```

Um das Paket `@types/react` mit all seinen Versionen zu löschen:

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

Löschen eines Maven-Pakets oder einer Paketversion

Um ein Maven-Paket oder einzelne Paketversionen zu löschen, setzen Sie die `--format` Option auf `maven` und geben Sie das zu löschende Paket an, indem Sie die Maven-Gruppen-ID mit der `--namespace` Option und die Maven-ArtifactID mit der Option übergeben. `--name` Im Folgenden wird beispielsweise gezeigt, wie eine einzelne Version von gelöscht wird: `com.google.guava: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
```

Das folgende Beispiel zeigt, wie das Paket `com.google.guava:guava` einschließlich aller seiner Versionen gelöscht wird:

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

Bewährte Methoden zum Löschen von Paketen oder Paketversionen

Wenn Sie eine Paketversion löschen müssen, empfiehlt es sich als bewährte Methode, ein Repository zu erstellen, um eine Sicherungskopie der Paketversion zu speichern, die Sie löschen möchten. Sie können dies tun, indem Sie zuerst `copy-package-versions` das Backup-Repository aufrufen:

```
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
```

Sobald Sie die Paketversion kopiert haben, können Sie dann das Paket oder die Paketversion aufrufen`delete-package-versions`, die Sie löschen möchten.

```
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
```

Details und Abhängigkeiten der Paketversion anzeigen und aktualisieren

Informationen zu einer Paketversion, einschließlich Abhängigkeiten, finden Sie unter CodeArtifact. Sie können auch den Status einer Paketversion aktualisieren. Weitere Informationen zum Status der Paketversion finden Sie unterStatus der Paketversion.

Details zur Paketversion anzeigen

Verwenden Sie den `describe-package-version` Befehl, um Details zu Paketversionen anzuzeigen. Paketversionsdetails werden aus einem Package extrahiert, wenn es veröffentlicht wird CodeArtifact. Die Details in den verschiedenen Paketen variieren und hängen von ihren Formaten und davon ab, wie viele Informationen die Autoren ihnen hinzugefügt haben.

Die meisten Informationen in der Ausgabe des `describe-package-version` Befehls hängen vom Paketformat ab. `describe-package-version`Extrahiert beispielsweise die Informationen eines NPM-Pakets aus seiner `package.json` Datei. Die Revision wurde von CodeArtifact erstellt. Weitere Informationen finden Sie unter Angabe einer Paketversionsrevision.

Zwei Paketversionen mit demselben Namen können sich im selben Repository befinden, wenn sie sich jeweils in unterschiedlichen Namespaces befinden. Verwenden Sie den optionalen `--namespace` Parameter, um einen Namespace anzugeben. Weitere Informationen finden Sie unter Versionsdetails des npm-Pakets anzeigen oder Versionsdetails des Maven-Pakets anzeigen.

Das folgende Beispiel gibt Details zur Version `1.9.0` eines Python-Pakets mit dem Namen zurück`pyhamcrest`, das sich im `my_repo` Repository befindet.

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

Die Ausgabe könnte wie folgt aussehen.

```
{
  "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 ruft Paketversionsdetails wie die Paket-Homepage oder Paketlizenzinformationen aus den vom Paketautor bereitgestellten Metadaten ab. Wenn eine dieser Informationen 400 KB überschreitet, was der Größenbeschränkung für DynamoDB-Elemente entspricht, können solche Daten CodeArtifact nicht verarbeitet werden, und Sie sehen diese Informationen möglicherweise nicht auf der Konsole oder in der Antwort von. `describe-package-version` Zum Beispiel ein Python-Paket wie https://py pi. org/project/

rapyd-sdk/ hat ein sehr großes Lizenzfeld, sodass diese Informationen nicht von verarbeitet werden würden. CodeArtifact

Versionsdetails des npm-Pakets anzeigen

Um Details zu einer NPM-Paketversion anzuzeigen, setzen Sie den Wert der `--format` Option auf. **`npm`** Fügen Sie optional den Namespace der Paketversion (NPM-Bereich) in die Option ein. `--namespace` Der Wert für die `--namespace` Option sollte den Anfang nicht enthalten. `@` Um nach dem Namespace zu suchen`@types`, setzen Sie den Wert auf*`types`*.

Im Folgenden werden Details zur Version `4.41.5` eines `webpack` im Bereich genannten npm-Pakets zurückgegeben. `@types`

```
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
```

Die Ausgabe könnte wie folgt aussehen.

```
{
  "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"
    }
}
```

Versionsdetails des Maven-Pakets anzeigen

Um Details zu einer Maven-Paketversion anzuzeigen, setzen Sie den Wert der `--format` Option auf `maven` und schließen Sie den Namespace der Paketversion in die Option ein. `--namespace`

Das folgende Beispiel gibt Details zur Version `1.2` eines Maven-Pakets mit dem Namen zurück`commons-rng-client-api`, das sich im `org.apache.commons` Namespace und im Repository befindet. `my_repo`

```
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
```

Die Ausgabe könnte wie folgt aussehen.

```
{
  "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 extrahiert keine Paketversionsdetailinformationen aus den übergeordneten POM-Dateien. Die Metadaten für eine bestimmte Paketversion enthalten nur Informationen im POM für genau diese Paketversion, nicht für das übergeordnete POM oder ein anderes POM, auf das mithilfe des `parent` POM-Tags transitiv verwiesen wird. Das bedeutet, dass bei der

Ausgabe von Metadaten (wie Lizenzinformationen) für Maven-Paketversionen weggelassen `describe-package-version` werden, die auf eine `parent` Referenz angewiesen sind, um diese Metadaten zu enthalten.

Abhängigkeiten von Paketversionen anzeigen

Verwenden Sie den `list-package-version-dependencies` Befehl, um eine Liste der Abhängigkeiten einer Paketversion abzurufen. Der folgende Befehl listet die Abhängigkeiten eines npm-Pakets mit dem Namen `my-package` Version `4.41.5` im `my_repo` Repository in der `my_domain` Domäne auf.

```
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
```

Die Ausgabe könnte wie folgt aussehen.

```
{
  "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"
}
```

Den Bereich der unterstützten Werte für das DependencyType-Feld finden Sie unter dem PackageDependencyDatentyp in der API. CodeArtifact

Readme-Datei zur Paketversion anzeigen

Einige Paketformate, wie z. B. npm, enthalten eine `README` Datei. Verwenden Sie die`get-package-version-readme`, um die `README` Datei einer Paketversion abzurufen. Der folgende Befehl gibt die `README` Datei eines NPM-Pakets mit dem Namen `my-package` Version `4.41.5` im `my_repo` Repository in der `my_domain` Domäne zurück.

> **Note**
>
> CodeArtifact unterstützt nicht die Anzeige von Readme-Dateien aus generischen Paketen oder Maven-Paketen.

```
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
```

Die Ausgabe könnte wie folgt aussehen.

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

Aktualisiere den Status der Paketversion

Jede Paketversion in CodeArtifact hat einen Status, der den aktuellen Status und die Verfügbarkeit der Paketversion beschreibt. Sie können den Status der Paketversion sowohl mit der AWS CLI als auch mit der Konsole ändern.

> **Note**
>
> Weitere Informationen zum Status der Paketversion, einschließlich einer Liste der verfügbaren Status, finden Sie unterStatus der Paketversion.

Der Status der Paketversion wird aktualisiert

Durch das Festlegen des Status einer Paketversion können Sie steuern, wie eine Paketversion verwendet werden kann, ohne sie vollständig aus dem Repository zu löschen. Wenn eine Paketversion beispielsweise den Status hat`Unlisted`, kann sie weiterhin wie gewohnt heruntergeladen werden, erscheint aber nicht in den Paketversionslisten, die auf Befehle wie zurückgegeben werden`npm view`. Die UpdatePackageVersionsStatus API ermöglicht das Festlegen des Paketversionsstatus mehrerer Versionen desselben Pakets in einem einzigen API-Aufruf. Eine Beschreibung der verschiedenen Status finden Sie unterÜberblick über Pakete.

Verwenden Sie den `update-package-versions-status` Befehl, um den Status einer Paketversion in `Published``Unlisted`, oder `Archived` zu ändern. Informationen zu den für die Verwendung des Befehls erforderlichen IAM-Berechtigungen finden Sie unterErforderliche IAM-Berechtigungen, um den Status einer Paketversion zu aktualisieren. Im folgenden Beispiel wird der Status von Version 4.1.0 des npm-Pakets auf gesetzt. `chalk` `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
```

Beispielausgabe:

```
{
    "successfulVersions": {
        "4.1.0": {
            "revision": "+Oz8skWbwY3k8M6SrNIqNj6bVH/ax+CxvkJx+No5j8I=",
            "status": "Archived"
        }
    },
    "failedVersions": {}
}
```

In diesem Beispiel wird ein npm-Paket verwendet, aber der Befehl funktioniert auch für andere Formate. Mehrere Versionen können mit einem einzigen Befehl in denselben Zielstatus versetzt werden, siehe das folgende Beispiel.

```
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
```

Beispielausgabe:

```
{
    "successfulVersions": {
        "4.1.0": {
            "revision": "25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ4=",
            "status": "Archived"
        },
        "4.1.1": {
            "revision": "+Oz8skWbwY3k8M6SrNIqNj6bVH/ax+CxvkJx+No5j8I=",
            "status": "Archived"
        }
    },
    "failedVersions": {}
}
```

Beachten Sie, dass eine einmal veröffentlichte Paketversion nicht wieder in den `Unfinished` Status zurückversetzt werden kann. Daher ist dieser Status als Wert für den `--target-status` Parameter nicht zulässig. Um die Paketversion in den `Disposed` Status zu verschieben, verwenden Sie stattdessen den `dispose-package-versions` Befehl wie unten beschrieben.

Erforderliche IAM-Berechtigungen, um den Status einer Paketversion zu aktualisieren

Um ein Paket `update-package-versions-status` anfordern zu können, müssen Sie über die `codeartifact:UpdatePackageVersionsStatus` entsprechende Berechtigung für die Paketressource verfügen. Das bedeutet, dass Sie die Berechtigung `update-package-versions-status` zum Aufrufen pro Paket erteilen können. Eine IAM-Richtlinie, die die Erlaubnis zum Aufrufen `update-package-versions-status` des npm-Pakets erteilt, *`chalk`* würde beispielsweise eine Aussage wie die folgende enthalten.

```
{
```

```
  "Action": [
    "codeartifact:UpdatePackageVersionsStatus"
  ],
  "Effect": "Allow",
  "Resource": "arn:aws:codeartifact:us-east-1:111122223333:package/my_domain/my_repo/
npm//chalk"
}
```

Der Status eines NPM-Pakets mit Gültigkeitsbereich wird aktualisiert

Verwenden Sie den Parameter, um den Paketversionsstatus einer npm-Paketversion mit einem Bereich zu aktualisieren. `--namespace` Verwenden Sie beispielsweise den folgenden Befehl, um die Liste der Version 8.0.0 von `@nestjs/core` aufzuheben.

```
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
```

Status für ein Maven-Paket wird aktualisiert

Maven-Pakete haben immer eine Gruppen-ID, die in als Namespace bezeichnet wird. CodeArtifact Verwenden Sie den `--namespace` Parameter, um die Maven-Gruppen-ID beim Aufrufen anzugeben. `update-package-versions-status` Verwenden Sie beispielsweise den folgenden Befehl, um Version 2.13.1 des Maven-Pakets `org.apache.logging.log4j:log4j` zu archivieren.

```
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
```

Angabe einer Paketversionsrevision

Eine Paketversionsrevision ist eine Zeichenfolge, die einen bestimmten Satz von Elementen und Metadaten für eine Paketversion angibt. Sie können eine Paketversionsrevision angeben, um den Status von Paketversionen zu aktualisieren, die sich in einem bestimmten Status befinden. Um eine Paketversionsrevision anzugeben, verwenden Sie den `--version-revisions` Parameter, um eine oder mehrere durch Kommas getrennte Paketversionen und die Revisionspaare der Paketversionen zu übergeben. Der Status einer Paketversion wird nur aktualisiert, wenn die aktuelle Version der Paketversion dem angegebenen Wert entspricht.

> **Note**
>
> Der `--versions` Parameter muss auch definiert werden, wenn der `--version-revisions` Parameter verwendet wird.

```
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
```

Um mehrere Versionen mit einem einzigen Befehl zu aktualisieren, übergeben Sie den Optionen eine kommagetrennte Liste von Versions- und Versionsrevisionspaaren. `--version-revisions` Der folgende Beispielbefehl definiert zwei verschiedene Revisionspaare für Paketversion und Paketversion.

```
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/
R80Rc9gL1P8vbzVMJ4=,4.0.0=E3lhBp0RObRTut4pkjV5c1AQGkgSA7Oxtil6hMMzelc="
 --versions 4.1.0 4.0.0 --target-status Published
```

Beispielausgabe:

```
{
    "successfulVersions": {
        "4.0.0": {
            "revision": "E3lhBp0RObRTut4pkjV5c1AQGkgSA7Oxtil6hMMzelc=",
            "status": "Published"
        },
        "4.1.0": {
            "revision": "25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ4=",
            "status": "Published"
        }
    },
    "failedVersions": {}
}
```

Beim Aktualisieren mehrerer Paketversionen `--version-revisions` müssen die übergebenen Versionen mit den übergebenen Versionen identisch sein`--versions`. Wenn eine Revision falsch angegeben wurde, wird der Status dieser Version nicht aktualisiert.

Unter Verwendung des Parameters „Erwarteter Status“

Der `update-package-versions-status` Befehl stellt den `--expected-status` Parameter bereit, der die Angabe des erwarteten aktuellen Status einer Paketversion unterstützt. Wenn der aktuelle Status nicht mit dem übergebenen Wert übereinstimmt`--expected-status`, wird der Status dieser Paketversion nicht aktualisiert.

Beispielsweise haben die Versionen 4.0.0 und 4.1.0 des npm-Pakets `chalk` derzeit den Status. *`my_repo`* `Published` Bei einem Aufrufupdate`-package-versions-status`, der den erwarteten Status von angibt, können beide Paketversionen nicht aktualisiert `Unlisted` werden, da der Status nicht übereinstimmt.

```
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
```

Beispielausgabe:

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

Fehler bei einzelnen Paketversionen

Es gibt mehrere Gründe, warum der Status einer Paketversion beim Aufrufen nicht aktualisiert wirdupdate`-package-versions-status`. Beispielsweise wurde die Revision der Paketversion

möglicherweise falsch angegeben oder der erwartete Status entspricht nicht dem aktuellen Status. In diesen Fällen wird die Version in der `failedVersions` Map in der API-Antwort enthalten sein. Wenn eine Version fehlschlägt, werden andere Versionen, die im selben Aufruf von angegeben wurden, `update-package-versions-status` möglicherweise übersprungen und ihr Status wird nicht aktualisiert. Solche Versionen werden auch mit einem „`errorCode`von`SKIPPED`“ in die `failedVersions` Map aufgenommen.

In der aktuellen Implementierung von werden alle anderen Versionen übersprungen`update-package-versions-status`, wenn der Status einer oder mehrerer Versionen nicht geändert werden kann. Das heißt, entweder wurden alle Versionen erfolgreich aktualisiert oder es wurden keine Versionen aktualisiert. Dieses Verhalten ist im API-Vertrag nicht garantiert. In future könnten einige Versionen erfolgreich sein, während andere Versionen bei einem einzigen Aufruf von fehlschlagen`update-package-versions-status`.

Der folgende Beispielbefehl beinhaltet einen Fehler bei der Aktualisierung des Versionsstatus, der auf eine nicht übereinstimmende Paketversion zurückzuführen ist. Dieser Aktualisierungsfehler führt dazu, dass ein weiterer Aufruf zur Aktualisierung des Versionsstatus übersprungen wird.

```
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=E3lhBp0RObRTut4pkjV5c1AQGkgSA7Oxtil6hMMzelc="
  --versions 4.1.0 4.0.0 --target-status Archived
```

Beispielausgabe:

```
{
    "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="
        }
    }
```

```
}
```

Entsorgung von Paketversionen

Der `Disposed` Paketstatus verhält sich ähnlich wie`Archived`, mit der Ausnahme, dass die Paketressourcen dauerhaft gelöscht werden, CodeArtifact sodass dem Konto des Domaininhabers keine Kosten mehr für die Speicherung der Inhalte in Rechnung gestellt werden. Weitere Informationen zum Status der einzelnen Paketversionen finden Sie unterStatus der Paketversion. Verwenden Sie den `dispose-package-versions` Befehl`Disposed`, um den Status einer Paketversion auf zu zu ändern. Diese Funktion ist unabhängig davon`update-package-versions-status`, dass das Löschen einer Paketversion nicht rückgängig gemacht werden kann. Da die Paketressourcen gelöscht werden, kann der Status der Version nicht wieder auf `Archived``Unlisted`, oder `Published` zurückgesetzt werden. Die einzige Aktion, die für eine entsorgte Paketversion ausgeführt werden kann, besteht darin, sie mithilfe des `delete-package-versions` Befehls zu löschen.

Für einen `dispose-package-versions` erfolgreichen Aufruf muss der aufrufende IAM-Principal über die `codeartifact:DisposePackageVersions` entsprechende Berechtigung für die Paketressource verfügen.

Das Verhalten des `dispose-package-versions` Befehls ist ähnlich wie`update-package-versions-status`, einschließlich des Verhaltens der `--expected-status` Optionen `--version-revisions` und, die in den Abschnitten Versionsrevision und Erwarteter Status beschrieben sind. Der folgende Befehl versucht beispielsweise, eine Paketversion zu löschen, schlägt jedoch fehl, weil der erwartete Status nicht übereinstimmt.

```
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
```

Beispielausgabe:

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

```
        }
    }
}
```

Wenn derselbe Befehl erneut mit einem `--expected-status` of ausgeführt wird`Published`, ist die Löschung erfolgreich.

```
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
```

Beispielausgabe:

```
{
    "successfulVersions": {
        "4.0.0": {
            "revision": "E3lhBp0RObRTut4pkjV5c1AQGkgSA7Oxtil6hMMzelc=",
            "status": "Disposed"
        }
    },
    "failedVersions": {}
}
```

Die Einstellungen zur Herkunft des Pakets werden bearbeitet

In können Paketversionen einem Repository hinzugefügt werden AWS CodeArtifact, indem sie direkt veröffentlicht, aus einem Upstream-Repository heruntergeladen oder aus einem externen, öffentlichen Repository aufgenommen werden. Wenn Sie zulassen, dass Paketversionen eines Pakets sowohl durch direkte Veröffentlichung als auch durch Aufnahme aus öffentlichen Repositorys hinzugefügt werden, sind Sie anfällig für Angriffe, die Abhängigkeiten ersetzen. Weitere Informationen finden Sie unter Angriffe zur Substitution von Abhängigkeiten. Um sich vor einem Angriff durch die Substitution von Abhängigkeiten zu schützen, können Sie die Kontrolle des Paketursprungs für ein Paket in einem Repository konfigurieren, um einzuschränken, wie Versionen dieses Pakets dem Repository hinzugefügt werden können.

Die Konfiguration von Kontrollen zur Paketherkunft sollte von jedem Team in Betracht gezogen werden, das zulassen möchte, dass neue Versionen verschiedener Pakete sowohl aus internen Quellen wie Direktveröffentlichungen als auch aus externen Quellen wie öffentlichen Repositorien stammen. Standardmäßig werden die Kontrollen für den Paketursprung darauf konfiguriert, wie die erste Version eines Pakets zum Repository hinzugefügt wird. Hinweise zu den Einstellungen für die Steuerung des Paketursprungs und deren Standardwerte finden Sie unterEinstellungen zur Kontrolle des Paketursprungs.

Um den Paketdatensatz nach der Verwendung des `put-package-origin-configuration` API-Vorgangs zu entfernen, verwenden Sie `delete-package` (siheLöschen Sie ein Paket oder eine Paketversion).

Allgemeine Szenarien zur Paketzugriffskontrolle

Dieser Abschnitt enthält einige allgemeine Szenarien, wenn eine Paketversion zu einem CodeArtifact Repository hinzugefügt wird. Die Einstellungen zur Kontrolle des Paketursprungs werden für neue Pakete festgelegt, je nachdem, wie die erste Paketversion hinzugefügt wird.

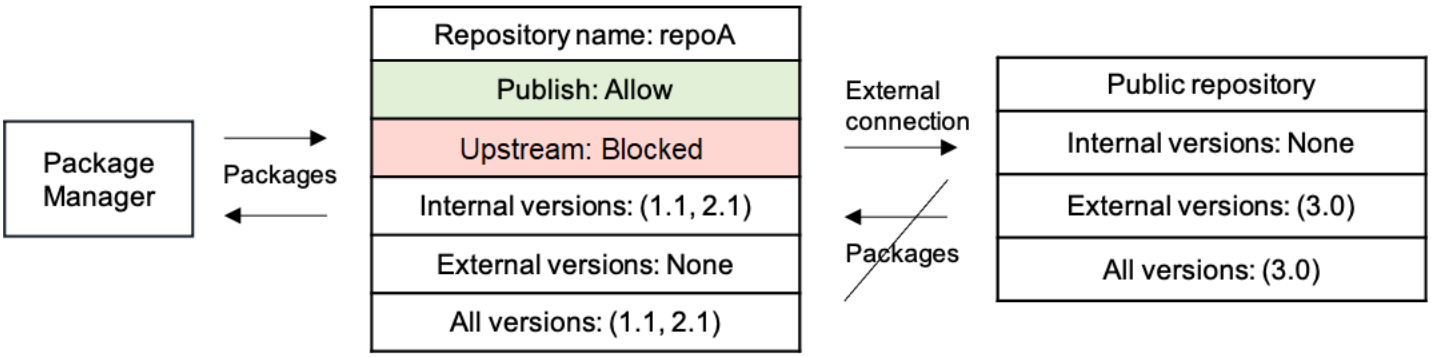
In den folgenden Szenarien ist ein internes Paket ein Paket, das direkt von einem Paketmanager in Ihrem Repository veröffentlicht wird, z. B. ein Paket, das Sie oder Ihr Team erstellen und verwalten. Ein externes Paket ist ein Paket, das in einem öffentlichen Repository vorhanden ist und über eine externe Verbindung in Ihr Repository aufgenommen werden kann.

Eine externe Paketversion wird für ein vorhandenes internes Paket veröffentlicht

Stellen Sie sich in diesem Szenario ein internes Paket, PackageA, vor. Ihr Team veröffentlicht die erste Paketversion für PackageA in einem Repository. CodeArtifact Da dies die erste Paketversion

für dieses Paket ist, werden die Einstellungen für die Kontrolle des Paketursprungs automatisch auf Veröffentlichen: Zulassen und Upstream: Blockieren gesetzt. Sobald das Paket in Ihrem Repository vorhanden ist, wird ein Paket mit demselben Namen in einem öffentlichen Repository veröffentlicht, das mit Ihrem CodeArtifact Repository verbunden ist. Dies könnte ein versuchter Angriff zur Substitution von Abhängigkeiten auf das interne Paket sein, oder es könnte auch einfach ein Zufall sein. Unabhängig davon sind die Kontrollen zur Paketherkunft so konfiguriert, dass sie die Aufnahme der neuen externen Version blockieren, um sich vor einem möglichen Angriff zu schützen.

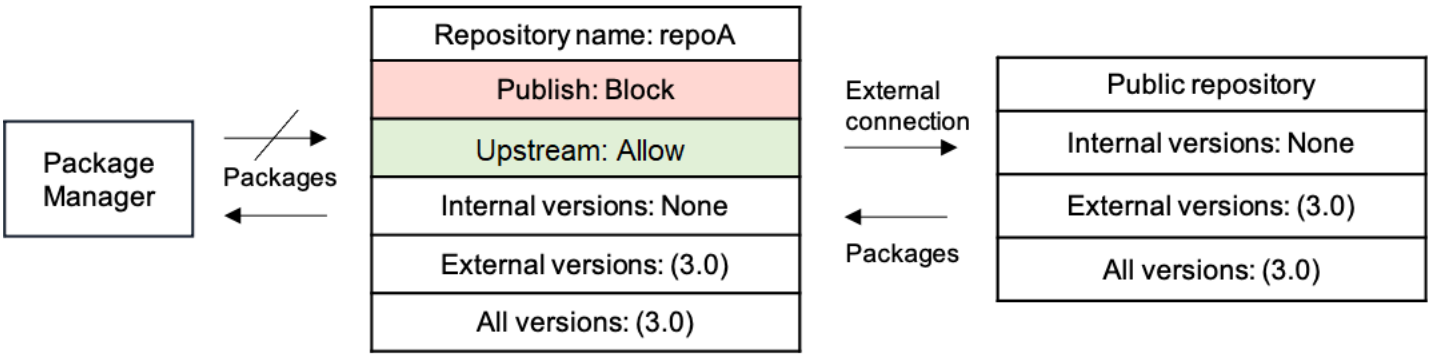
In der folgenden Abbildung ist RepoA Ihr CodeArtifact Repository mit einer externen Verbindung zu einem öffentlichen Repository. Ihr Repository enthält die Versionen 1.1 und 2.1 von PackageA, aber Version 3.0 ist im öffentlichen Repository veröffentlicht. Normalerweise würde RepoA Version 3.0 aufnehmen, nachdem das Paket von einem Paketmanager angefordert wurde. Da die Paketaufnahme auf Blockieren gesetzt ist, wird Version 3.0 nicht in Ihr CodeArtifact Repository aufgenommen und steht den damit verbundenen Paketmanagern nicht zur Verfügung.



Eine interne Paketversion wird für ein vorhandenes externes Paket veröffentlicht

In diesem Szenario existiert ein Paket, PackageB, extern in einem öffentlichen Repository, das Sie mit Ihrem Repository verbunden haben. Wenn ein mit Ihrem Repository verbundener Paketmanager PackageB anfordert, wird die Paketversion aus dem öffentlichen Repository in Ihr Repository aufgenommen. Da dies die erste Paketversion von PackageB ist, die zu Ihrem Repository hinzugefügt wurde, sind die Einstellungen für den Paketursprung auf Publish: BLOCK und Upstream: ALLOW konfiguriert. Später versuchen Sie, eine Version mit demselben Paketnamen im Repository zu veröffentlichen. Entweder kennen Sie das öffentliche Paket nicht und versuchen, ein Paket, das nichts damit zu tun hat, unter demselben Namen zu veröffentlichen, oder Sie versuchen, eine gepatchte Version zu veröffentlichen, oder Sie versuchen, genau die Paketversion, die bereits extern existiert, direkt zu veröffentlichen. CodeArtifact lehnt die Version ab, die Sie veröffentlichen möchten, ermöglicht es Ihnen jedoch, die Ablehnung explizit zu überschreiben und die Version bei Bedarf zu veröffentlichen.

In der folgenden Abbildung ist RepoA Ihr CodeArtifact Repository mit einer externen Verbindung zu einem öffentlichen Repository. Ihr Repository enthält Version 3.0, die es aus dem öffentlichen Repository aufgenommen hat. Sie möchten Version 1.1 in Ihrem Repository veröffentlichen. Normalerweise könnten Sie Version 1.2 in RepoA veröffentlichen, aber da die Veröffentlichung auf Blockieren eingestellt ist, kann Version 1.2 nicht veröffentlicht werden.



Veröffentlichung einer gepatchten Paketversion eines vorhandenen externen Pakets

In diesem Szenario existiert ein Paket, PackageB, extern in einem öffentlichen Repository, das Sie mit Ihrem Repository verbunden haben. Wenn ein mit Ihrem Repository verbundener Paketmanager PackageB anfordert, wird die Paketversion aus dem öffentlichen Repository in Ihr Repository aufgenommen. Da dies die erste Paketversion von PackageB ist, die zu Ihrem Repository hinzugefügt wurde, sind die Einstellungen für den Paketursprung auf Publish: BLOCK und Upstream: ALLOW konfiguriert. Ihr Team entscheidet, dass es gepatchte Paketversionen dieses Pakets im Repository veröffentlichen muss. Um Paketversionen direkt veröffentlichen zu können, ändert Ihr Team die Einstellungen zur Kontrolle des Paketursprungs in Publish: ALLOW und Upstream: BLOCK. Versionen dieses Pakets können jetzt direkt in Ihrem Repository veröffentlicht und aus öffentlichen Repositorys aufgenommen werden. Nachdem Ihr Team die gepatchten Paketversionen veröffentlicht hat, setzt Ihr Team die Einstellungen für den Paketursprung auf Publish: BLOCK und Upstream: ALLOW zurück.

Einstellungen zur Kontrolle des Paketursprungs

Mit den Steuerelementen für den Paketursprung können Sie konfigurieren, wie Paketversionen zu einem Repository hinzugefügt werden können. Die folgenden Listen enthalten die verfügbaren Einstellungen und Werte für die Steuerung des Paketursprungs.

> **Note**
>
> Die verfügbaren Einstellungen und Werte unterscheiden sich bei der Konfiguration der Ursprungskontrollen für Paketgruppen. Weitere Informationen finden Sie unter Herkunftskontrollen für Paketgruppen.

Veröffentlichen

Diese Einstellung konfiguriert, ob Paketversionen mithilfe von Paketmanagern oder ähnlichen Tools direkt im Repository veröffentlicht werden können.

- ZULASSEN: Paketversionen können direkt veröffentlicht werden.
- BLOCK: Paketversionen können nicht direkt veröffentlicht werden.

Upstream

Diese Einstellung konfiguriert, ob Paketversionen aus externen, öffentlichen Repositorys aufgenommen oder von Upstream-Repositorys beibehalten werden können, wenn dies von einem Paketmanager angefordert wird.

- ALLOW: Jede Paketversion kann aus anderen CodeArtifact Repositorys beibehalten werden, die als Upstream-Repositorys konfiguriert sind, oder von einer öffentlichen Quelle mit einer externen Verbindung aufgenommen werden.
- BLOCKIEREN: Paketversionen können nicht aus anderen CodeArtifact Repositorys aufbewahrt werden, die als Upstream-Repositorys konfiguriert sind, oder von einer öffentlichen Quelle mit einer externen Verbindung aufgenommen werden.

Standardeinstellungen für die Kontrolle des Paketursprungs

Die Standardeinstellungen für die Kontrolle des Paketursprungs werden auf der Grundlage der Einstellungen für die Ursprungskontrolle der Paketgruppe konfiguriert, die dem Paket zugeordnet ist. Weitere Informationen zu Paketgruppen und zur Kontrolle der Herkunft von Paketgruppen finden Sie unter Arbeiten mit Paketgruppen in CodeArtifact undHerkunftskontrollen für Paketgruppen.

Wenn ein Paket einer Paketgruppe zugeordnet ist, deren Einschränkungseinstellungen `ALLOW` für jeden Einschränkungstyp aktiviert sind, basieren die Standardsteuerungen für den Paketursprung für ein Paket darauf, wie die erste Version dieses Pakets dem Repository hinzugefügt wurde.

- Wenn die erste Paketversion direkt von einem Paketmanager veröffentlicht wird, lauten die Einstellungen Publish: ALLOW und Upstream: BLOCK.
- Wenn die erste Paketversion aus einer öffentlichen Quelle aufgenommen wurde, lauten die Einstellungen Publish: BLOCK und Upstream: ALLOW.

> **Note**
>
> Für Pakete, die vor etwa Mai 2022 in CodeArtifact Repositorys existierten, gelten standardmäßig die Einstellungen Publish: ALLOW und Upstream: ALLOW. Die Kontrolle des Paketursprungs muss für solche Pakete manuell festgelegt werden. Die aktuellen Standardwerte wurden seitdem für neue Pakete festgelegt und ab dem Start der Funktion am 14. Juli 2022 durchgesetzt. Weitere Informationen zur Einstellung von Kontrollen zur Herkunft von Paketen finden Sie unterDie Einstellungen für den Paketursprung werden bearbeitet.

Andernfalls, wenn ein Paket einer Paketgruppe zugeordnet ist, die mindestens eine Einschränkungseinstellung von `BLOCK` oder hat`ALLOW_SPECIFIC_REPOSITORIES`, dann werden die Standardeinstellungen für die Ursprungskontrolle für dieses Paket auf Publish: ALLOW und Upstream: ALLOW gesetzt.

Wie die Kontrollen zur Herkunft von Paketen mit den Ursprungskontrollen für Paketgruppen interagieren

Da Pakete über Einstellungen für die Ursprungskontrolle und die zugehörigen Paketgruppen über Einstellungen für die Ursprungskontrolle verfügen, ist es wichtig zu verstehen, wie diese beiden unterschiedlichen Einstellungen miteinander interagieren.

Die Wechselwirkung zwischen den beiden Einstellungen besteht darin, dass eine Einstellung von `BLOCK` immer Vorrang vor einer Einstellung von hat`ALLOW`. In der folgenden Tabelle sind einige Beispielkonfigurationen und ihre effektiven Origin Control-Einstellungen aufgeführt.

Einstellung zur Kontrolle des Paketursprungs	Einstellung zur Kontrolle des Ursprungs von Paketgruppen	Effektive Einstellung für die Ursprungskontrolle
VERÖFFENTLICHEN: ZULASSEN	VERÖFFENTLICHEN: ZULASSEN	VERÖFFENTLICHEN: ZULASSEN

Einstellung zur Kontrolle des Paketursprungs	Einstellung zur Kontrolle des Ursprungs von Paketgruppen	Effektive Einstellung für die Ursprungskontrolle
STROMAUFWÄRTS: ZULASSEN	STROMAUFWÄRTS: ZULASSEN	STROMAUFWÄRTS: ZULASSEN
VERÖFFENTLICHEN: BLOCKIEREN STROMAUFWÄRTS: ZULASSEN	VERÖFFENTLICHEN: ZULASSEN STROMAUFWÄRTS: ZULASSEN	VERÖFFENTLICHEN: BLOCKIEREN STROMAUFWÄRTS: ZULASSEN
VERÖFFENTLICHEN: ZULASSEN	VERÖFFENTLICHEN: ZULASSEN	VERÖFFENTLICHEN: ZULASSEN
STROMAUFWÄRTS: ZULASSEN	STROMAUFWÄRTS: BLOCKIEREN	STROMAUFWÄRTS: BLOCKIEREN

Das bedeutet, dass ein Paket mit den Ursprungseinstellungen Publish: ALLOW und Upstream: ALLOW quasi auf die Origin-Control-Einstellungen der zugehörigen Paketgruppe zurückstellt.

Die Einstellungen für den Paketursprung werden bearbeitet

Die Kontrollen zur Paketherkunft werden automatisch konfiguriert, je nachdem, wie die erste Paketversion eines Pakets zum Repository hinzugefügt wurde. Weitere Informationen finden Sie unterStandardeinstellungen für die Kontrolle des Paketursprungs. Gehen Sie wie folgt vor, um Steuerungen für den Paketursprung für ein Paket in einem CodeArtifact Repository hinzuzufügen oder zu bearbeiten.

Um Steuerelemente für den Paketursprung hinzuzufügen oder zu bearbeiten (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Repositories und dann das Repository aus, das das Paket enthält, das Sie bearbeiten möchten.
3. Suchen Sie in der Tabelle Pakete nach dem Paket, das Sie bearbeiten möchten, und wählen Sie es aus.
4. Wähle auf der Seite mit der Paketübersicht in Origin Controls die Option Bearbeiten aus.

5. Wähle unter Ursprungskontrollen bearbeiten die Kontrollen für den Paketursprung aus, die du für dieses Paket einrichten möchtest. Beide Einstellungen für die Steuerung des Paketursprungs, Publish und Upstream, müssen gleichzeitig festgelegt werden.

   - Um die direkte Veröffentlichung von Paketversionen zuzulassen, wählen Sie unter Veröffentlichen die Option Zulassen aus. Um die Veröffentlichung von Paketversionen zu blockieren, wählen Sie Blockieren aus.
   - Um die Aufnahme von Paketen aus externen Repositorys und das Abrufen von Paketen aus Upstream-Repositorys zuzulassen, wählen Sie unter Upstream-Quellen die Option Zulassen aus. Um die gesamte Aufnahme und das Abrufen von Paketversionen aus externen und Upstream-Repositorys zu blockieren, wählen Sie Blockieren.

Um Kontrollen zur Paketherkunft hinzuzufügen oder zu bearbeiten ()AWS CLI

1. Falls nicht, konfigurieren Sie das, AWS CLI indem Sie die Schritte unter befolgenEinrichtung mit AWS CodeArtifact.
2. Verwenden Sie den `put-package-origin-configuration` Befehl, um Steuerelemente für den Paketursprung hinzuzufügen oder zu bearbeiten. Ersetzen Sie die folgenden Felder:

   - *`my_domain`*Ersetzen Sie es durch die CodeArtifact Domain, die das Paket enthält, das Sie aktualisieren möchten.
   - *`my_repo`*Ersetzen Sie es durch das CodeArtifact Repository, das das Paket enthält, das Sie aktualisieren möchten.
   - *`npm`*Ersetzen Sie es durch das Paketformat des Pakets, das Sie aktualisieren möchten.
   - *`my_package`*Ersetzen Sie es durch den Namen des Pakets, das Sie aktualisieren möchten.
   - Ersetzen Sie *`ALLOW`* und *`BLOCK`* durch Ihre gewünschten Einstellungen für die Kontrolle des Paketursprungs.

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

Veröffentlichungs- und Upstream-Repositorys

CodeArtifact erlaubt nicht das Veröffentlichen von Paketversionen, die in erreichbaren Upstream-Repositorys oder öffentlichen Repositorys vorhanden sind. Nehmen wir zum Beispiel an, Sie möchten ein Maven-Paket `com.mycompany.mypackage:1.0` in einem Repository `myrepo` veröffentlichen und `myrepo` haben ein Upstream-Repository mit einer externen Verbindung zu Maven Central. Stellen Sie sich die folgenden Szenarien vor.

1. Die Einstellungen für die Kontrolle des Paketursprungs `com.mycompany.mypackage` lauten Publish: ALLOW und Upstream: ALLOW. Wenn im Upstream-Repository oder in Maven Central vorhanden `com.mycompany.mypackage:1.0` ist, wird jeder Versuch, darin zu veröffentlichen, `myrepo` mit einem 409-Konfliktfehler CodeArtifact zurückgewiesen. Sie könnten immer noch eine andere Version veröffentlichen, z. `com.mycompany.mypackage:1.1`
2. Die Einstellungen für die Kontrolle des Paketursprungs `com.mycompany.mypackage` lauten Publish: ALLOW und Upstream: BLOCK. Sie können jede Version von `com.mycompany.mypackage` in Ihrem Repository veröffentlichen, die noch nicht existiert, da Paketversionen nicht erreichbar sind.
3. Die Einstellungen zur Kontrolle des Paketursprungs `com.mycompany.mypackage` lauten Publish: BLOCK und Upstream: ALLOW. Sie können keine Paketversionen direkt in Ihrem Repository veröffentlichen.

Arbeiten mit Paketgruppen in CodeArtifact

Paketgruppen können verwendet werden, um die Konfiguration auf mehrere Pakete anzuwenden, die einem definierten Muster entsprechen, wobei das Paketformat, der Paketnamespace und der Paketname verwendet werden. Sie können Paketgruppen verwenden, um die Steuerung des Paketursprungs für mehrere Pakete bequemer zu konfigurieren. Kontrollen zur Paketherkunft werden verwendet, um die Aufnahme oder Veröffentlichung neuer Paketversionen zu blockieren oder zuzulassen. Dadurch werden Benutzer vor böswilligen Aktionen geschützt, die als Abhängigkeitsersetzungsangriffe bezeichnet werden.

Jede Domäne enthält CodeArtifact automatisch eine Stammpaketgruppe. Diese Stammpaketgruppe/*,, enthält alle Pakete und ermöglicht es Paketversionen standardmäßig, Repositorys aller Herkunft in der Domäne einzugeben. Die Stammpaketgruppe kann geändert, aber nicht gelöscht werden.

Die Funktion zur Konfiguration von Paketgruppen funktioniert letztendlich konsistent, wenn eine neue Paketgruppe erstellt oder eine bestehende Paketgruppe gelöscht wird. Das bedeutet, dass beim Erstellen oder Löschen einer Paketgruppe die Ursprungskontrollen auf die erwarteten zugehörigen Pakete angewendet werden, allerdings mit einer gewissen Verzögerung aufgrund des letztlich konsistenten Verhaltens. Wie lange es dauert, bis eine endgültige Konsistenz erreicht ist, hängt von der Anzahl der Paketgruppen in der Domäne sowie von der Anzahl der Pakete in der Domäne ab. Es kann einen kurzen Zeitraum geben, in dem sich die Herkunftskontrollen nach dem Erstellen oder Löschen einer Paketgruppe nicht sofort auf die zugehörigen Pakete auswirken.

Darüber hinaus sind Aktualisierungen der Ursprungskontrollen für Paketgruppen fast sofort wirksam. Im Gegensatz zum Erstellen oder Löschen von Paketgruppen werden Änderungen an den Ursprungskontrollen einer vorhandenen Paketgruppe ohne die gleiche Verzögerung in den zugehörigen Paketen übernommen.

Diese Themen enthalten Informationen zu Paketgruppen in AWS CodeArtifact.

Themen

- Erstellen Sie eine Paketgruppe
- Eine Paketgruppe anzeigen oder bearbeiten
- Löschen Sie eine Paketgruppe
- Herkunftskontrollen für Paketgruppen
- Syntax und Abgleichverhalten der Paketgruppendefinition

- Kennzeichnen Sie eine Paketgruppe in CodeArtifact

Erstellen Sie eine Paketgruppe

Sie können eine Paketgruppe mit der CodeArtifact Konsole, der AWS Command Line Interface (AWS CLI) oder erstellen AWS CloudFormation. Weitere Informationen zur Verwaltung von CodeArtifact Paketgruppen mit CloudFormation finden Sie unter CodeArtifact Ressourcen erstellen mit AWS CloudFormation.

Erstellen Sie eine Paketgruppe (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann die Domain aus, in der Sie eine Paketgruppe erstellen möchten.
3. Wählen Sie Paketgruppen und anschließend Paketgruppe erstellen aus.
4. Geben Sie Package Paketgruppendefinition die Paketgruppendefinition für Ihre Paketgruppe ein. Die Paketgruppendefinition bestimmt, welche Pakete der Gruppe zugeordnet sind. Sie können die Paketgruppendefinition manuell mit Text eingeben, oder Sie können den visuellen Modus verwenden, um eine Auswahl zu treffen, und die Paketgruppendefinition wird automatisch erstellt.
5. So verwenden Sie den visuellen Modus, um die Paketgruppendefinition zu erstellen:

   a. Wählen Sie Visual, um in den visuellen Modus zu wechseln.

   b. Wählen Sie Package Paketformat das Format der Pakete aus, die dieser Gruppe zugeordnet werden sollen.

   c. Wählen Sie unter Namespace (Scope) die Namespace-Kriterien aus, nach denen gesucht werden soll.

      - Entspricht: Entspricht exakt dem angegebenen Namespace. Falls ausgewählt, geben Sie den Namespace ein, für den der Abgleich vorgenommen werden soll.
      - Leer: Findet Pakete ohne Namespace.
      - Beginnt mit einem Wort: Findet Namespaces, die mit einem bestimmten Wort beginnen. Falls ausgewählt, geben Sie das Präfixwort ein, nach dem gesucht werden soll. Weitere Informationen zu Wörtern und Wortgrenzen finden Sie unterWörter, Wortgrenzen und Präfixabgleich.

- Alle: Entspricht Paketen in allen Namespaces.

d. Wenn Gleich, Leer oder Beginnt mit Wort ausgewählt ist, wählen Sie im Feld Paketname die Kriterien für den Paketnamen aus, nach denen gesucht werden soll.

- Entspricht genau: Entspricht exakt dem angegebenen Paketnamen. Falls ausgewählt, geben Sie den Paketnamen ein, für den ein Abgleich erforderlich ist.
- Beginnt mit Präfix: Findet Pakete, die mit dem angegebenen Präfix beginnen.
- Beginnt mit einem Wort: Findet Pakete, die mit einem bestimmten Wort beginnen. Falls ausgewählt, geben Sie das Präfixwort ein, nach dem gesucht werden soll. Weitere Informationen zu Wörtern und Wortgrenzen finden Sie unterWörter, Wortgrenzen und Präfixabgleich.
- Alle: Entspricht allen Paketen.

e. Wählen Sie Weiter, um die Definition zu überprüfen.

6. Um die Paketgruppendefinition mit Text einzugeben:

a. Wählen Sie Text, um in den Textmodus zu wechseln.

b. Geben Sie im Feld Paketgruppendefinition die Paketgruppendefinition ein. Weitere Informationen zur Syntax der Paketgruppendefinition finden Sie unterSyntax und Abgleichverhalten der Paketgruppendefinition.

c. Wählen Sie Weiter, um die Definition zu überprüfen.

7. Überprüfen Sie unter Definition überprüfen die Pakete, die auf der Grundlage der zuvor bereitgestellten Definition in die neue Paketgruppe aufgenommen werden. Wählen Sie nach der Überprüfung Weiter aus.

8. Fügen Sie Package Paketgruppeninformationen optional eine Beschreibung und eine Kontakt-E-Mail für die Paketgruppe hinzu. Wählen Sie Weiter.

9. Konfigurieren Sie unter Kontrollen zur Paketherkunft die Ursprungskontrollen, die auf die Pakete in der Gruppe angewendet werden sollen. Weitere Informationen zu den Ursprungskontrollen für Paketgruppen finden Sie unterHerkunftskontrollen für Paketgruppen.

10. Wählen Sie „Paketgruppe erstellen“.

Erstellen Sie eine Paketgruppe (AWS CLI)

Verwenden Sie den `create-package-group` Befehl, um eine Paketgruppe in Ihrer Domain zu erstellen. Geben Sie für die `--package-group` Option die Paketgruppendefinition ein,

die bestimmt, welche Pakete der Gruppe zugeordnet sind. Weitere Hinweise zur Syntax der Paketgruppendefinitionen finden Sie unterSyntax und Abgleichverhalten der Paketgruppendefinition.

Falls nicht, konfigurieren Sie das, AWS CLI indem Sie die Schritte unter befolgenEinrichtung mit 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
```

Eine Paketgruppe anzeigen oder bearbeiten

Sie können eine Liste aller Paketgruppen anzeigen, Details zu einer bestimmten Paketgruppe anzeigen oder die Details oder die Konfiguration einer Paketgruppe mit der CodeArtifact Konsole oder der AWS Command Line Interface (AWS CLI) bearbeiten.

Eine Paketgruppe anzeigen oder bearbeiten (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann die Domain aus, die die Paketgruppe enthält, die Sie anzeigen oder bearbeiten möchten.
3. Wählen Sie Paketgruppen und wählen Sie die Paketgruppe aus, die Sie anzeigen oder bearbeiten möchten.
4. Sehen Sie sich unter Details Informationen zur Paketgruppe an, einschließlich ihrer übergeordneten Gruppe, Beschreibung, ARN, Kontakt-E-Mail und Kontrollen zur Paketherkunft.
5. Sehen Sie sich unter Untergruppen eine Liste der Paketgruppen an, denen diese Gruppe als übergeordnete Gruppe zugeordnet ist. Die Paketgruppen in dieser Liste können Einstellungen von dieser Paketgruppe erben. Weitere Informationen finden Sie unter Paketgruppenhierarchie und Musterspezifität.
6. Sehen Sie sich unter Pakete die Pakete an, die zu dieser Paketgruppe gehören, basierend auf der Paketgruppendefinition. In der Spalte Stärke können Sie die Stärke der Paketzuordnung sehen. Weitere Informationen finden Sie unter Paketgruppenhierarchie und Musterspezifität.

7. Um die Paketgruppeninformationen zu bearbeiten, wählen Sie Paketgruppe bearbeiten.

    a. Aktualisieren Sie unter Informationen die Beschreibung oder Kontaktinformationen der Paketgruppe. Sie können die Definition einer Paketgruppe nicht bearbeiten.

    b. Aktualisieren Sie unter Herkunftskontrollen für Paketgruppen die Einstellungen für die Ursprungssteuerung der Paketgruppe, die festlegen, wie zugehörige Pakete in Repositorys in der Domain aufgenommen werden können. Weitere Informationen finden Sie unter Herkunftskontrollen für Paketgruppen.

Eine Paketgruppe anzeigen oder bearbeiten ()AWS CLI

Verwenden Sie die folgenden Befehle, um Paketgruppen mit dem anzuzeigen oder zu bearbeiten AWS CLI. Falls nicht, konfigurieren Sie das, AWS CLI indem Sie die Schritte unter befolgenEinrichtung mit AWS CodeArtifact.

Verwenden Sie den `list-package-groups` Befehl, um alle Paketgruppen in einer Domäne anzuzeigen.

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

Verwenden Sie den `describe-package-group` Befehl, um Details zu einer Paketgruppe anzuzeigen. Weitere Informationen zu Paketgruppendefinitionen finden Sie unterSyntax und Beispiele für Paketgruppendefinitionen.

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

Verwenden Sie den `list-sub-package-groups` Befehl, um die untergeordneten Paketgruppen einer Paketgruppe anzuzeigen.

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

Verwenden Sie den `get-associated-package-group` Befehl, um die Paketgruppe anzuzeigen, die einem Paket zugeordnet ist. Sie müssen den normalisierten Paketnamen und Namespace für die NuGet Paketformate Python und Swift verwenden. Weitere Informationen darüber, wie die Paketnamen und Namespaces normalisiert werden, finden Sie in der NuGetDokumentation zur Namensnormalisierung von Python und Swift.

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

Verwenden Sie den Befehl, um eine Paketgruppe zu bearbeiten. `update-package-group` Dieser Befehl wird verwendet, um die Kontaktinformationen oder die Beschreibung einer Paketgruppe zu aktualisieren. Informationen zu den Einstellungen für die Ursprungssteuerung von Paketgruppen und zum Hinzufügen oder Bearbeiten dieser Einstellungen finden Sie unterHerkunftskontrollen für Paketgruppen. Weitere Informationen zu Paketgruppendefinitionen finden Sie unter Syntax und Beispiele für Paketgruppendefinitionen

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

Löschen Sie eine Paketgruppe

Sie können eine Paketgruppe mit der CodeArtifact Konsole oder der AWS Command Line Interface (AWS CLI) löschen.

Beachten Sie beim Löschen von Paketgruppen das folgende Verhalten:

- Die Stammpaketgruppe,`/*`, kann nicht gelöscht werden.
- Die Pakete und Paketversionen, die dieser Paketgruppe zugeordnet sind, werden nicht gelöscht.
- Wenn eine Paketgruppe gelöscht wird, werden die direkt untergeordneten Paketgruppen zu untergeordneten Paketgruppen der direkt übergeordneten Paketgruppe der Paketgruppe. Wenn

also eine der untergeordneten Gruppen Einstellungen von der übergeordneten Gruppe erbt, können sich diese Einstellungen ändern.

Löschen Sie eine Paketgruppe (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann die Domain aus, die die Paketgruppe enthält, die Sie anzeigen oder bearbeiten möchten.
3. Wählen Sie Paketgruppen.
4. Wählen Sie die Paketgruppe aus, die Sie löschen möchten, und wählen Sie Löschen.
5. Geben Sie Löschen in das Feld ein und wählen Sie Löschen.

Löschen Sie eine Paketgruppe (AWS CLI)

Verwenden Sie den `delete-package-group` Befehl, um eine Paketgruppe zu löschen.

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

Herkunftskontrollen für Paketgruppen

Steuerungen zur Paketherkunft werden verwendet, um zu konfigurieren, wie Paketversionen in eine Domäne gelangen können. Sie können Ursprungskontrollen für eine Paketgruppe einrichten, um zu konfigurieren, wie Versionen aller Pakete, die der Paketgruppe zugeordnet sind, in bestimmte Repositorys in der Domäne gelangen können.

Die Einstellungen für die Ursprungssteuerung von Paketgruppen bestehen aus den folgenden Einstellungen:

- Einstellungen für Einschränkungen: Diese Einstellungen definieren, ob Pakete CodeArtifact aus veröffentlichten, internen Upstreams oder externen, öffentlichen Repositorys in ein Repository gelangen können.

- Listen zulässiger Repositorys: Jede Einschränkungseinstellung kann so eingestellt werden, dass bestimmte Repositorien zugelassen werden. Wenn eine Einschränkungseinstellung so eingestellt ist, dass bestimmte Repositorys zugelassen werden, wird für diese Einschränkung eine entsprechende Liste zulässiger Repositorys angezeigt.

> **Note**
>
> Die Einstellungen für die Ursprungskontrolle für Paketgruppen unterscheiden sich geringfügig von den Einstellungen für die Ursprungskontrolle für einzelne Pakete. Weitere Informationen zu den Origin-Control-Einstellungen für Pakete finden Sie unterEinstellungen zur Kontrolle des Paketursprungs.

Einstellungen für Einschränkungen

Die Einschränkungseinstellungen der Ursprungssteuerungseinstellungen einer Paketgruppe bestimmen, wie die mit dieser Gruppe verknüpften Pakete in Repositorys in der Domäne aufgenommen werden können.

PUBLISH

Die `PUBLISH` Einstellung konfiguriert, ob Paketversionen mithilfe von Paketmanagern oder ähnlichen Tools direkt in einem beliebigen Repository in der Domäne veröffentlicht werden können.

- ZULASSEN: Paketversionen können direkt in allen Repositorien veröffentlicht werden.
- BLOCK: Paketversionen können nicht direkt in einem Repository veröffentlicht werden.
- ALLOW_SPECIFIC_REPOSITORIES: Paketversionen können nur direkt in Repositorys veröffentlicht werden, die in der Liste der erlaubten Repositorys für die Veröffentlichung angegeben sind.
- INHERIT: Die `PUBLISH` Einstellung wird von der ersten übergeordneten Paketgruppe mit einer Einstellung übernommen, bei der dies nicht der Fall ist. `INHERIT`

EXTERNAL_UPSTREAM

Die `EXTERNAL_UPSTREAM` Einstellung konfiguriert, ob Paketversionen aus externen, öffentlichen Repositorys aufgenommen werden können, wenn sie von einem Paketmanager angefordert werden.

Eine Liste der unterstützten externen Repositorys finden Sie unter. Unterstützte Repositorys für externe Verbindungen

- ZULASSEN: Jede Paketversion kann von einer öffentlichen Quelle mit einer externen Verbindung in alle Repositorys aufgenommen werden.
- BLOCK: Paketversionen können nicht von einer öffentlichen Quelle mit einer externen Verbindung in ein Repository aufgenommen werden.
- ALLOW_SPECIFIC_REPOSITORIES: Paketversionen können nur aus einer öffentlichen Quelle in Repositorys aufgenommen werden, die in der Liste der zulässigen Repositorys für externe Upstreams angegeben sind.
- INHERIT: Die `EXTERNAL_UPSTREAM` Einstellung wird von der ersten übergeordneten Paketgruppe übernommen, mit einer Einstellung, die dies nicht ist. `INHERIT`

INTERNAL_UPSTREAM

Die `INTERNAL_UPSTREAM` Einstellung konfiguriert, ob Paketversionen von internen Upstream-Repositorys in derselben CodeArtifact Domäne beibehalten werden können, wenn sie von einem Paketmanager angefordert werden.

- ALLOW: Jede Paketversion kann aus anderen CodeArtifact Repositorys, die als Upstream-Repositorys konfiguriert sind, beibehalten werden.
- BLOCK: Paketversionen können nicht aus anderen CodeArtifact Repositorys aufbewahrt werden, die als Upstream-Repositorys konfiguriert sind.
- CodeArtifact ALLOW_SPECIFIC_REPOSITORIES: Paketversionen können nur von anderen Repositorien, die als Upstream-Repositorys konfiguriert sind, in Repositorys aufbewahrt werden, die in der Liste der zulässigen Repositorys für interne Upstreams angegeben sind.
- INHERIT: Die **`INTERNAL_UPSTREAM`** Einstellung wird von der ersten übergeordneten Paketgruppe übernommen, mit einer Einstellung, die dies nicht ist. `INHERIT`

Listen zulässiger Repositorys

Wenn eine Einschränkungseinstellung als konfiguriert ist`ALLOW_SPECIFIC_REPOSITORIES`, enthält die Paketgruppe eine zugehörige Liste zulässiger Repositorys, die eine Liste der Repositorys enthält, die für diese Einschränkungseinstellung zulässig sind. Daher enthält eine Paketgruppe zwischen 0 und 3 Listen zulässiger Repositorys, eine für jede Einstellung, die als konfiguriert ist. `ALLOW_SPECIFIC_REPOSITORIES`

Wenn Sie ein Repository zur Liste der zulässigen Repositorys einer Paketgruppe hinzufügen, müssen Sie angeben, zu welcher Liste zulässiger Repositorys es hinzugefügt werden soll.

Die möglichen Listen zulässiger Repositorys lauten wie folgt:

- `EXTERNAL_UPSTREAM`: Erlaubt oder blockiert die Aufnahme von Paketversionen aus externen Repositorys im hinzugefügten Repository.
- `INTERNAL_UPSTREAM`: Erlaubt oder blockiert das Abrufen von Paketversionen aus einem anderen CodeArtifact Repository im hinzugefügten Repository.
- `PUBLISH`: Erlaubt oder blockiert die direkte Veröffentlichung von Paketversionen von Paketmanagern im hinzugefügten Repository.

Bearbeiten der Origin Control-Einstellungen für Paketgruppen

Gehen Sie wie folgt vor, um Ursprungskontrollen für eine Paketgruppe hinzuzufügen oder zu bearbeiten. Informationen zu den Einstellungen für die Ursprungssteuerung von Paketgruppen finden Sie unter Einstellungen für Einschränkungen undListen zulässiger Repositorys.

So fügen Sie Origin Controls (CLI) für Paketgruppen hinzu oder bearbeiten sie

1. Falls nicht, konfigurieren Sie das, AWS CLI indem Sie die Schritte unter befolgenEinrichtung mit AWS CodeArtifact.
2. Verwenden Sie den `update-package-group-origin-configuration` Befehl, um Steuerelemente für den Paketursprung hinzuzufügen oder zu bearbeiten.

   - Geben Sie für `--domain` die CodeArtifact Domäne ein, die die Paketgruppe enthält, die Sie aktualisieren möchten.
   - Geben Sie für `--domain-owner` die Kontonummer des Domaininhabers ein.
   - Geben Sie für `--package-group` die Paketgruppe ein, die Sie aktualisieren möchten.
   - Geben Sie für `--restrictions` Schlüssel-Wert-Paare ein, die die Einschränkungen der Ursprungskontrolle darstellen.
   - Geben Sie für ein JSON-Objekt ein`--add-allowed-repositories`, das den Einschränkungstyp und den Repository-Namen enthält, um es der entsprechenden Liste der zulässigen Repositorys für die Einschränkung hinzuzufügen.

- Geben Sie für ein JSON-Objekt ein`--remove-allowed-repositories`, das den Einschränkungstyp und den Repository-Namen enthält, die aus der entsprechenden Liste der zulässigen Repositorys für die Einschränkung entfernt werden sollen.

```
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
```

Im folgenden Beispiel werden mehrere Einschränkungen und mehrere Repositorys in einem Befehl hinzugefügt.

```
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=
 \
   --add-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2 \
   --remove-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2
```

Konfigurationsbeispiele für die Ursprungssteuerung von Paketgruppen

Die folgenden Beispiele zeigen Konfigurationen der Paketursprungssteuerung für gängige Paketverwaltungsszenarien.

Zulassen, dass Pakete mit privaten Namen veröffentlicht, aber nicht aufgenommen werden

Dieses Szenario ist wahrscheinlich ein übliches Szenario in der Paketverwaltung:

- Erlauben Sie, dass Pakete mit privaten Namen von Paketmanagern in Repositorys in Ihrer Domain veröffentlicht werden, und verhindern Sie, dass sie von externen, öffentlichen Repositorys in Repositorys in Ihrer Domain aufgenommen werden.
- Erlauben Sie, dass alle anderen Pakete aus externen, öffentlichen Repositorys in Repositorys in Ihrer Domain aufgenommen werden, und verhindern Sie, dass sie von Paketmanagern in Repositorys in Ihrer Domain veröffentlicht werden.

Um dies zu erreichen, sollten Sie eine Paketgruppe mit einem Muster konfigurieren, das die privaten Namen und die Ursprungseinstellungen PUBLISH: ALLOW, EXTERNAL_UPSTREAM: BLOCK und INTERNAL_UPSTREAM: ALLOW enthält. Dadurch wird sichergestellt, dass Pakete mit privaten Namen direkt veröffentlicht, aber nicht aus externen Repositorys aufgenommen werden können.

Die folgenden AWS CLI Befehle erstellen und konfigurieren eine Paketgruppe mit Einstellungen für die Quellenbeschränkung, die dem gewünschten Verhalten entsprechen:

Um die Paketgruppe zu erstellen:

```
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"
```

Um die Ursprungskonfiguration der Paketgruppe zu aktualisieren:

```
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
```

Erlaubt die Aufnahme aus externen Repositorys über ein Repository

In diesem Szenario hat Ihre Domain mehrere Repositorys. Von diesen Repositorys `repoA` hat eine Upstream-Verbindung zu dem öffentlichen Repository`repoB`, das eine externe Verbindung zum öffentlichen Repository hat`npmjs.com`, wie hier gezeigt:

```
repoA --> repoB --> npmjs.com
```

Sie möchten die Aufnahme von Paketen aus einer bestimmten Paketgruppe zulassen, zwar von dort `/npm/space/anycompany~` aus `npmjs.com`repoA, aber nur durch. `repoB` Sie möchten auch die Aufnahme von Paketen, die der Paketgruppe zugeordnet sind, in andere Repositorys in Ihrer Domain blockieren und die direkte Veröffentlichung von Paketen mit Paketmanagern blockieren. Um dies zu erreichen, erstellen und konfigurieren Sie die Paketgruppe wie folgt:

Ursprungseinschränkungseinstellungen für PUBLISH: BLOCK und EXTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES und INTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES.

`repoA`und zur Liste der entsprechenden zulässigen Repositorys hinzugefügt: `repoB`

- `repoA`sollte der `INTERNAL_UPSTREAM` Liste hinzugefügt werden, da es Pakete von seinem internen Upstream erhält,`repoB`.
- `repoB`sollte der `EXTERNAL_UPSTREAM` Liste hinzugefügt werden, da es Pakete aus dem externen Repository bezieht,`npmjs.com`.

Die folgenden AWS CLI Befehle erstellen und konfigurieren eine Paketgruppe mit Ursprungseinschränkungseinstellungen, die dem gewünschten Verhalten entsprechen:

Um die Paketgruppe zu erstellen:

```
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"
```

Um die Ursprungskonfiguration der Paketgruppe zu aktualisieren:

```
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=ALLO
 \
```

```
  --add-allowed-repositories
originRestrictionType=INTERNAL_UPSTREAM,repositoryName=repoA
originRestrictionType=EXTERNAL_UPSTREAM,repositoryName=repoB
```

Wie die Einstellungen für die Ursprungssteuerung von Paketgruppen mit den Einstellungen für die Paketursprungskontrolle interagieren

Da Pakete über Einstellungen für die Ursprungskontrolle und die zugehörigen Paketgruppen über Einstellungen für die Ursprungskontrolle verfügen, ist es wichtig zu verstehen, wie diese beiden unterschiedlichen Einstellungen miteinander interagieren. Informationen zur Interaktion zwischen den Einstellungen finden Sie unterWie die Kontrollen zur Herkunft von Paketen mit den Ursprungskontrollen für Paketgruppen interagieren.

Syntax und Abgleichverhalten der Paketgruppendefinition

Dieses Thema enthält Informationen zur Definition von Paketgruppen, zum Verhalten beim Mustervergleich, zur Stärke der Paketzuordnungen und zur Paketgruppenhierarchie.

Inhalt

- Syntax und Beispiele für Paketgruppendefinitionen
  - Definition und Normalisierung von Paketgruppen
  - Namespaces in Paketgruppendefinitionen
- Paketgruppenhierarchie und Musterspezifität
- Wörter, Wortgrenzen und Präfixabgleich
- Groß-/Kleinschreibung
- Starkes und schwaches Spiel
- Zusätzliche Varianten

Syntax und Beispiele für Paketgruppendefinitionen

Die Mustersyntax für die Definition von Paketgruppen folgt weitgehend der Formatierung von Paketpfaden. Ein Paketpfad wird aus den Koordinatenkomponenten eines Pakets (Format, Namespace und Name) erstellt, indem am Anfang ein Schrägstrich hinzugefügt und die einzelnen Komponenten durch einen Schrägstrich getrennt werden. Der Paketpfad für das im Namespace-

Space benannte npm-Paket lautet beispielsweise anycompany-ui-components/-ui-components. npm/space/anycompany

Ein Paketgruppenmuster folgt derselben Struktur wie ein Paketpfad, außer dass Komponenten, die nicht als Teil der Gruppendefinition angegeben sind, weggelassen werden und das Muster mit einem Suffix abgeschlossen wird. Das enthaltene Suffix bestimmt das Übereinstimmungsverhalten des Musters wie folgt:

- Ein $ Suffix entspricht der vollständigen Paketkoordinate.
- Ein ~ Suffix entspricht einem Präfix.
- Ein * Suffix entspricht allen Werten der zuvor definierten Komponente.

Hier sind Beispielmuster für jede der zulässigen Kombinationen:

1. Alle Paketformate: `/*`
2. Ein bestimmtes Paketformat: `/npm/*`
3. Paketformat und Namespace-Präfix: `/maven/com.anycompany~`
4. Paketformat und Namespace: `/npm/space/*`
5. Paketformat, Namespace und Namenspräfix: `/npm/space/anycompany-ui~`
6. Paketformat, Namespace und Name: `/maven/org.apache.logging.log4j/log4j-core$`

Wie in den obigen Beispielen gezeigt, wird das ~ Suffix am Ende eines Namespaces oder Namens hinzugefügt, um eine Präfixübereinstimmung darzustellen. Es * steht hinter einem Schrägstrich, wenn es verwendet wird, um alle Werte für die nächste Komponente im Pfad abzugleichen (entweder alle Formate, alle Namespaces oder alle Namen).

Definition und Normalisierung von Paketgruppen

CodeArtifact normalisiert Python NuGet - und Swift-Paketnamen und normalisiert Swift-Paketnamespaces, bevor sie gespeichert werden. CodeArtifact verwendet diese normalisierten Namen, wenn Pakete mit Paketgruppendefinitionen abgeglichen werden. Daher müssen Paketgruppen, die einen Namespace oder Namen in diesen Formaten enthalten, den normalisierten Namespace und Namen verwenden. Weitere Informationen darüber, wie die Paketnamen und Namespaces normalisiert werden, finden Sie in der NuGetDokumentation zur Namensnormalisierung von Python und Swift.

Namespaces in Paketgruppendefinitionen

Bei Paketen oder Paketformaten ohne Namespace (Python und NuGet) dürfen Paketgruppen keinen Namespace enthalten. Die Paketgruppendefinition für diese Paketgruppen enthält einen leeren Namespace-Abschnitt. Der Pfad für das Python-Paket mit dem Namen requests lautet beispielsweise /python//requests.

Bei Paketen oder Paketformaten mit einem Namespace (Maven, Generic und Swift) muss der Namespace enthalten sein, wenn der Paketname enthalten ist. Für das Swift-Paketformat wird der normalisierte Paket-Namespace verwendet. Weitere Hinweise zur Normalisierung von Swift-Paketnamespaces finden Sie unter. Schnelle Normalisierung von Paketnamen und Namespace

Paketgruppenhierarchie und Musterspezifität

Die Pakete, die sich in einer Paketgruppe befinden oder ihr „zugeordnet“ sind, sind Pakete mit einem Paketpfad, der dem Muster der Gruppe entspricht, aber nicht dem Muster einer spezifischeren Gruppe entspricht. Wenn beispielsweise die Paketgruppen `/npm/*` und angegeben sind`/npm/space/*`, ist der Paketpfad /npm//react der ersten Gruppe (`/npm/*`) zugeordnet, während/npm/space/aui.components und/npm/space/amplify-ui-core der zweiten Gruppe () zugeordnet sind. `/npm/space/*` Obwohl ein Paket mehreren Gruppen entsprechen kann, ist jedes Paket nur einer einzigen Gruppe zugeordnet, wobei die spezifischste Gruppe zutrifft, und nur die Konfiguration dieser einen Gruppe gilt für das Paket.

Wenn ein Paketpfad mehreren Mustern entspricht, kann das „spezifischere“ Muster als das am längsten übereinstimmende Muster angesehen werden. Alternativ ist das spezifischere Muster dasjenige, das einer richtigen Teilmenge der Pakete entspricht, die dem weniger spezifischen Muster entsprechen. Aus unserem vorherigen Beispiel geht hervor, dass jedes Paket, das übereinstimmt`/npm/*`, `/npm/space/*` auch übereinstimmt, aber das Gegenteil ist nicht der Fall, weshalb `/npm/space/*` das Muster spezifischer ist, weil es eine richtige Teilmenge von ist. `/npm/*` Da es sich bei einer Gruppe um eine Teilmenge einer anderen Gruppe handelt, entsteht eine Hierarchie, in der `/npm/space/*` sich eine Untergruppe der übergeordneten Gruppe befindet. `/npm/*`

Obwohl nur die Konfiguration der spezifischsten Paketgruppe für ein Paket gilt, kann diese Gruppe so konfiguriert werden, dass sie von der Konfiguration der übergeordneten Gruppe erbt.

Wörter, Wortgrenzen und Präfixabgleich

Bevor wir uns mit dem Präfixabgleich befassen, wollen wir einige Schlüsselbegriffe definieren:

- Ein Wort, ein Buchstabe oder eine Zahl, gefolgt von null oder mehr Buchstaben, Zahlen oder Markierungszeichen (wie Akzente, Umlaute usw.).
- Eine Wortgrenze befindet sich am Ende eines Wortes, wenn ein Zeichen erreicht wird, das kein Wort ist. Bei Sonderzeichen handelt es sich um Satzzeichen wie `.`, und. `-` `_`

Insbesondere lautet das Regex-Muster für ein Wort`[\p{L}\p{N}][\p{L}\p{N}\p{M}]*`, das wie folgt aufgeteilt werden kann:

- `\p{L}`steht für einen beliebigen Buchstaben.
- `\p{N}`steht für eine beliebige Zahl.
- `\p{M}`steht für ein beliebiges Markierungszeichen wie Akzente, Umlaute usw.

Steht daher `[\p{L}\p{N}]` für eine Zahl oder einen Buchstaben und `[\p{L}\p{N}\p{M}]*` steht für null oder mehr Buchstaben, Zahlen oder Markierungszeichen. Am Ende jeder Übereinstimmung dieses regulären Musters befindet sich eine Wortgrenze.

> **Note**
>
> Der Abgleich von Wortgrenzen basiert auf dieser Definition eines „Wortes“. Es basiert nicht auf Wörtern, die in einem Wörterbuch definiert sind, oder CameCase. Zum Beispiel gibt es in `oneword` oder keine Wortgrenze`OneWord`.

Nachdem das Wort und die Wortgrenze nun definiert sind, können wir sie verwenden, um den Präfixabgleich in zu beschreiben CodeArtifact. Um anzugeben, dass ein Präfix an einer Wortgrenze übereinstimmt, wird nach einem Wortzeichen ein Übereinstimmungszeichen (~) verwendet. Das Muster `/npm/space/foo~` entspricht beispielsweise den Paketpfaden `/npm/space/foo` und`/npm/space/foo-bar`, aber nicht `/npm/space/food` oder`/npm/space/foot`.

Ein Platzhalter (`*`) muss anstelle von verwendet werden, ~ wenn einem Zeichen gefolgt wird, das kein Wort ist, wie im Muster. `/npm/*`

Groß-/Kleinschreibung

Bei Paketgruppendefinitionen wird zwischen Groß- und Kleinschreibung unterschieden, was bedeutet, dass Muster, die sich nur durch die Groß- und Kleinschreibung unterscheiden, als

separate Paketgruppen existieren können. Beispielsweise kann ein Benutzer separate Paketgruppen mit den Mustern und `/npm//asyncstorage$` für die drei separaten Pakete erstellen `/npm//AsyncStorage$`/`npm//asyncStorage$`, die in der öffentlichen Registrierung von npm vorhanden sind:, asyncStorage AsyncStorage, asyncstorage, die sich nur durch die Groß- und Kleinschreibung unterscheiden.

Auch wenn die Groß- und Kleinschreibung wichtig ist, werden Pakete CodeArtifact trotzdem einer Paketgruppe zugeordnet, wenn das Paket eine Variation des Musters aufweist, die sich von Fall zu Fall unterscheidet. Wenn ein Benutzer die `/npm//AsyncStorage$` Paketgruppe erstellt, ohne die anderen beiden oben aufgeführten Gruppen zu erstellen, werden alle Varianten des Namens zwischen Groß- und Kleinschreibung AsyncStorage, einschließlich AsyncStorage und Asyncstorage, der Paketgruppe zugeordnet. Wie im nächsten Abschnitt beschrieben, Starkes und schwaches Spiel werden diese Variationen jedoch anders behandelt als das AsyncStorage, was genau dem Muster entspricht.

Starkes und schwaches Spiel

Die Informationen im vorherigen Abschnitt,, besagenGroß-/Kleinschreibung, dass bei Paketgruppen zwischen Groß- und Kleinschreibung unterschieden wird. Anschließend wird erklärt, dass sie nicht zwischen Groß- und Kleinschreibung unterscheiden. Das liegt daran, dass in CodeArtifact Paketgruppendefinitionen ein Konzept von starker Übereinstimmung (oder exakter Übereinstimmung) und schwacher Übereinstimmung (oder Variantenübereinstimmung) verwendet wird. Eine starke Übereinstimmung liegt vor, wenn das Paket dem Muster exakt und ohne jegliche Variation entspricht. Eine schwache Übereinstimmung liegt vor, wenn das Paket einer Variante des Musters entspricht, z. B. einer anderen Groß- und Kleinschreibung. Das Verhalten bei schwacher Übereinstimmung verhindert, dass Pakete, bei denen es sich um Varianten des Musters einer Paketgruppe handelt, zu einer allgemeineren Paketgruppe zusammengefasst werden. Wenn es sich bei einem Paket um eine Variante (schwache Übereinstimmung) des Musters handelt, das am genauesten zu der Gruppe passt, wird das Paket zwar der Gruppe zugeordnet, aber das Paket wird blockiert, anstatt die Konfiguration der Ursprungsverwaltung der Gruppe anzuwenden, wodurch verhindert wird, dass neue Versionen des Pakets aus Upstreams abgerufen oder veröffentlicht werden. Dieses Verhalten reduziert das Risiko von Angriffen auf die Lieferkette, die aus einer Verwechselung der Abhängigkeiten von Paketen mit nahezu identischen Namen resultieren.

Um das Verhalten bei schwachen Übereinstimmungen zu veranschaulichen, nehmen wir an, dass die Paketgruppe die Aufnahme `/npm/*` zulässt und die Veröffentlichung blockiert. Eine spezifischere Paketgruppe,, ist so konfiguriert`/npm//anycompany-spicy-client$`, dass sie die Aufnahme blockiert und die Veröffentlichung ermöglicht. Das angegebene anycompany-spicy-clientPaket

entspricht stark der Paketgruppe, die die Veröffentlichung von Paketversionen ermöglicht und die Aufnahme von Paketversionen blockiert. Die einzige Groß-/Kleinschreibung des Paketnamens, der veröffentlicht werden darf anycompany-spicy-client, ist, weil er dem Muster der Paketdefinition sehr gut entspricht. Bei einer anderen Variante von Groß- und Kleinschreibung, z. B. AnyCompany-spicy-client, ist die Veröffentlichung gesperrt, da es sich um eine schwache Übereinstimmung handelt. Noch wichtiger ist, dass die Paketgruppe die Aufnahme aller Varianten von Groß- und Kleinschreibung blockiert, nicht nur des im Muster verwendeten Kleinbuchstabennamens, wodurch das Risiko einer Konfusion von Abhängigkeiten verringert wird.

Zusätzliche Varianten

Neben Unterschieden zwischen Groß- und Kleinschreibung werden beim schwachen Abgleich auch Unterschiede in der Reihenfolge von Bindestrichen`-`, Punkten `.`_, Unterstrichen und verwechselbaren Zeichen (z. B. ähnlich aussehende Zeichen aus unterschiedlichen Alphabeten) ignoriert. Bei der Normalisierung, die für schwache Übereinstimmungen verwendet wird, wird Groß- und Kleinschreibung (ähnlich der Konvertierung in Kleinbuchstaben) CodeArtifact durchgeführt, Abfolgen von Bindestrich-, Punkt- und Unterstrichzeichen durch einen einzelnen Punkt ersetzt und verwirrende Zeichen normalisiert.

Bei schwachem Abgleich werden Bindestriche, Punkte und Unterstriche als gleichwertig behandelt, aber nicht vollständig ignoriert. Das bedeutet, dass foo-bar, foo.bar, foo.. bar und foo_bar allesamt schwache Match-Entsprechungen sind, foobar jedoch nicht. Obwohl mehrere öffentliche Repositorien Maßnahmen ergreifen, um diese Art von Variationen zu verhindern, macht der Schutz, den öffentliche Repositorien bieten, diese Funktion von Paketgruppen nicht unnötig. Beispielsweise verhindern öffentliche Repositorien wie die Registrierung für die öffentliche Registrierung von npm nur dann neue Varianten des Pakets mit dem Namen my-package, wenn my-package dort bereits veröffentlicht ist. Wenn my-package ein internes Paket ist und eine Paketgruppe erstellt */npm//my-package $* wird, die die Veröffentlichung ermöglicht und die Aufnahme blockiert, möchten Sie my-package wahrscheinlich nicht in der öffentlichen Registrierung von npm veröffentlichen, um zu verhindern, dass eine Variante wie my.package zugelassen wird.

Während einige Paketformate wie Maven diese Zeichen unterschiedlich behandeln (Maven behandelt sie als Trennzeichen für die Namespace-Hierarchie, aber nicht `-` oder_), könnte etwas `.` wie com.act-on dennoch mit com.act.on verwechselt werden.

> **Note**
>
> Beachten Sie, dass immer dann, wenn mehrere Varianten einer Paketgruppe zugeordnet sind, ein Administrator eine neue Paketgruppe für eine bestimmte Variante erstellen kann, um ein anderes Verhalten für diese Variante zu konfigurieren.

Kennzeichnen Sie eine Paketgruppe in CodeArtifact

Tags sind mit AWS-Ressourcen verknüpfte Schlüssel-Wert-Paare. Sie können Tags auf Ihre Paketgruppen in anwenden CodeArtifact. Informationen zur Kennzeichnung von CodeArtifact Ressourcen, zu Anwendungsfällen, Einschränkungen für Tagschlüssel und -werte sowie zu unterstützten Ressourcentypen finden Sie unterTaggen von -Ressourcen.

Sie können die CLI verwenden, um Tags anzugeben, wenn Sie eine Paketgruppe erstellen oder den Wert von Tags einer vorhandenen Paketgruppe hinzufügen, entfernen oder aktualisieren.

Tag-Paketgruppen (CLI)

Sie können die CLI verwenden, um Paketgruppen-Tags zu verwalten.

Falls nicht, konfigurieren Sie das, AWS CLI indem Sie die Schritte unter befolgenEinrichtung mit AWS CodeArtifact.

> **Tip**
>
> Um Tags hinzuzufügen, müssen Sie den Amazon-Ressourcennamen (ARN) der Paketgruppe angeben. Um den ARN der Paketgruppe abzurufen, führen Sie den `describe-package-group` folgenden Befehl aus:
>
> ```
> aws codeartifact describe-package-group \
>    --domain my_domain \
>    --package-group /npm/scope/anycompany~ \
>    --query packageGroup.arn
> ```

Themen

- Hinzufügen von Tags zu einer Paketgruppe (CLI)

- Tags für eine Paketgruppe anzeigen (CLI)
- Tags für eine Paketgruppe bearbeiten (CLI)
- Tags aus einer Paketgruppe entfernen (CLI)

Hinzufügen von Tags zu einer Paketgruppe (CLI)

Sie können Tags zu Paketgruppen hinzufügen, wenn sie erstellt werden, oder zu einer vorhandenen Paketgruppe. Informationen zum Hinzufügen von Tags zu einer Paketgruppe bei deren Erstellung finden Sie unterErstellen Sie eine Paketgruppe.

Um einer vorhandenen Paketgruppe ein Tag hinzuzufügen AWS CLI, führen Sie im Terminal oder in der Befehlszeile den tag-resource Befehl aus und geben Sie den Amazon-Ressourcennamen (ARN) der Paketgruppe an, der Sie Tags hinzufügen möchten, sowie den Schlüssel und den Wert des Tags, das Sie hinzufügen möchten. Informationen zur Paketgruppe finden ARNs Sie unterPaketgruppe ARNs.

Sie können einer Paketgruppe mehr als ein Tag hinzufügen. Um beispielsweise eine Paketgruppe */npm/scope/anycompany~* mit zwei Tags zu taggen, einem Tag-Schlüssel *key1* mit dem Tag-Wert von *value1* und einem Tag-Schlüssel *key2* mit dem Tag-Wert von*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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags für eine Paketgruppe anzeigen (CLI)

Gehen Sie wie folgt vor AWS CLI , um die AWS Tags für eine Paketgruppe anzuzeigen. Wenn keine Tags hinzugefügt wurden, ist die zurückgegebene Liste leer.

Führen Sie im Terminal oder in der Befehlszeile den list-tags-for-resource Befehl mit dem Amazon-Ressourcennamen (ARN) der Paketgruppe aus. Informationen zur Paketgruppe finden ARNs Sie unterPaketgruppe ARNs.

Um beispielsweise eine Liste von Tag-Schlüsseln und Tag-Werten für eine Paketgruppe anzuzeigen, die mit einem ARN-Wert von */npm/scope/anycompany~* benannt ist

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~
```

Bei erfolgreicher Ausführung gibt dieser Befehl etwa wie folgt aussehende Informationen zurück:

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

Tags für eine Paketgruppe bearbeiten (CLI)

Gehen Sie wie folgt vor AWS CLI , um ein Tag für eine Paketgruppe zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen. Sie können Tags auch aus einer Paketgruppe entfernen, wie im nächsten Abschnitt gezeigt.

Führen Sie im Terminal oder in der Befehlszeile den tag-resource Befehl aus und geben Sie den ARN der Paketgruppe an, in der Sie ein Tag aktualisieren möchten, und geben Sie den Tag-Schlüssel und den Tag-Wert an. Hinweise zur Paketgruppe finden ARNs Sie unterPaketgruppe 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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags aus einer Paketgruppe entfernen (CLI)

Gehen Sie wie folgt vor, AWS CLI um mit dem ein Tag aus einer Paketgruppe zu entfernen.

> **Note**
>
> Wenn Sie eine Paketgruppe löschen, werden alle Tag-Zuordnungen aus der gelöschten Paketgruppe entfernt. Sie müssen Tags nicht entfernen, bevor Sie eine Paketgruppe löschen.

Führen Sie den Befehl im Terminal oder in der untag-resource Befehlszeile aus und geben Sie den ARN der Paketgruppe an, aus der Sie Tags entfernen möchten, und den Tagschlüssel des Tags, das Sie entfernen möchten. Hinweise zur Paketgruppe finden ARNs Sie unterPaketgruppe ARNs.

Um beispielsweise mehrere Tags aus einer Paketgruppe zu entfernen*/npm/scope/anycompany~*, mit den Tag-Schlüsseln *key1* und*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
```

Bei Erfolg hat dieser Befehl keine Ausgabe. Nach dem Entfernen von Tags können Sie mit dem `list-tags-for-resource` Befehl die verbleibenden Tags in der Paketgruppe anzeigen.

Arbeiten Sie mit Domänen in CodeArtifact

CodeArtifact Domänen erleichtern die Verwaltung mehrerer Repositorys in einer Organisation. Sie können eine Domäne verwenden, um Berechtigungen auf viele Repositorys anzuwenden, die verschiedenen AWS-Konten gehören. Ein Asset wird nur einmal in einer Domain gespeichert, auch wenn es in mehreren Repositorys verfügbar ist.

Sie können zwar mehrere Domains haben, wir empfehlen jedoch eine einzige Produktionsdomain, die alle veröffentlichten Artefakte enthält, damit Ihre Entwicklungsteams Pakete finden und gemeinsam nutzen können. Sie können eine zweite Vorproduktionsdomäne verwenden, um Änderungen an der Konfiguration der Produktionsdomäne zu testen.

In diesen Themen wird beschrieben, wie Sie die CodeArtifact Konsole AWS CLI, die und AWS CloudFormation zum Erstellen oder Konfigurieren von CodeArtifact Domänen verwenden.

Themen

- Überblick über die Domäne
- Domain erstellen
- Domäne löschen
- Domain-Richtlinien
- Kennzeichnen Sie eine Domain in CodeArtifact

Überblick über die Domäne

Wenn Sie mit arbeiten CodeArtifact, sind Domains für Folgendes nützlich:

- Deduplizierter Speicher: Ein Asset muss nur einmal in einer Domain gespeichert werden, auch wenn es in 1 oder 1.000 Repositorys verfügbar ist. Das heißt, Sie zahlen nur einmal für Speicherplatz.
- Schnelles Kopieren: Wenn Sie Pakete aus einem CodeArtifact Upstream-Repository in ein Downstream-Repository abrufen oder die CopyPackageVersions API verwenden, müssen nur Metadatensätze aktualisiert werden. Es werden keine Assets kopiert. Dadurch lässt sich schnell ein neues Repository für Staging oder Tests einrichten. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.
- Einfache gemeinsame Nutzung zwischen Repositorys und Teams: Alle Ressourcen und Metadaten in einer Domain werden mit einem einzigen Schlüssel AWS KMS key (KMS-Schlüssel)

verschlüsselt. Sie müssen nicht für jedes Repository einen Schlüssel verwalten oder mehreren Konten Zugriff auf einen einzigen Schlüssel gewähren.

- Richtlinie auf mehrere Repositorys anwenden: Der Domainadministrator kann Richtlinien auf die gesamte Domain anwenden. Dazu gehört die Einschränkung, welche Konten Zugriff auf Repositorys in der Domain haben und wer Verbindungen zu öffentlichen Repositorys konfigurieren kann, um sie als Paketquellen zu verwenden. Weitere Informationen finden Sie unter Domänenrichtlinien.
- Eindeutige Repository-Namen: Die Domain stellt einen Namespace für Repositorys bereit. Repository-Namen müssen nur innerhalb der Domain eindeutig sein. Sie sollten aussagekräftige Namen verwenden, die leicht zu verstehen sind.

Domainnamen müssen innerhalb eines Kontos eindeutig sein.

Sie können kein Repository ohne Domain erstellen. Wenn Sie die CreateRepositoryAPI verwenden, um ein Repository zu erstellen, müssen Sie einen Domainnamen angeben. Sie können ein Repository nicht von einer Domain in eine andere verschieben.

Ein Repository kann demselben AWS Konto gehören, dem die Domain gehört, oder einem anderen Konto. Wenn es sich bei den Eigentümerkonten um unterschiedliche Konten handelt, muss dem Konto, das das Repository besitzt, die entsprechenden Berechtigungen für die `CreateRepository` Domänenressource erteilt werden. Sie können dies tun, indem Sie der Domäne mithilfe des Befehls eine Ressourcenrichtlinie hinzufügen. PutDomainPermissionsPolicy

Eine Organisation kann zwar mehrere Domänen haben, es wird jedoch empfohlen, eine einzige Produktionsdomäne zu verwenden, die alle veröffentlichten Artefakte enthält, sodass Entwicklungsteams Pakete innerhalb ihrer Organisation finden und gemeinsam nutzen können. Eine zweite Vorproduktionsdomäne kann nützlich sein, um Änderungen an der Konfiguration der Produktionsdomäne zu testen.

Kontenübergreifende Domänen

Domainnamen müssen nur innerhalb eines Kontos eindeutig sein, was bedeutet, dass es innerhalb einer Region mehrere Domains geben kann, die denselben Namen haben. Wenn Sie also auf eine Domain zugreifen möchten, die einem Konto gehört, für das Sie nicht authentifiziert sind, müssen Sie die Domaininhaber-ID zusammen mit dem Domainnamen sowohl in der CLI als auch in der Konsole angeben. Sehen Sie sich die folgenden CLI-Beispiele an.

Greifen Sie auf eine Domain zu, die einem Konto gehört, für das Sie authentifiziert sind:

Wenn Sie innerhalb des Kontos, für das Sie authentifiziert sind, auf eine Domain zugreifen, müssen Sie nur den Domainnamen angeben. Das folgende Beispiel listet Pakete im *my_repo* Repository in der *my_domain* Domain auf, die Ihrem Konto gehört.

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

Greifen Sie auf eine Domain zu, die einem Konto gehört, für das Sie nicht authentifiziert sind:

Wenn Sie auf eine Domain zugreifen, die einem Konto gehört, für das Sie nicht authentifiziert sind, müssen Sie den Domaininhaber sowie den Domainnamen angeben. Das folgende Beispiel listet Pakete im *other-repo* Repository in der *other-domain* Domain auf, die einem Konto gehört, für das Sie nicht authentifiziert sind. Beachten Sie die Hinzufügung des `--domain-owner` Parameters.

```
aws codeartifact list-packages --domain other-domain --domain-owner 111122223333 --
repository other-repo
```

Arten von AWS KMS Schlüsseln, die unterstützt werden in CodeArtifact

CodeArtifact unterstützt nur symmetrische KMS-Schlüssel. Sie können keinen asymmetrischen KMS-Schlüssel verwenden, um Ihre Domänen zu verschlüsseln. CodeArtifact Weitere Informationen finden Sie unter Identifizieren symmetrischer und asymmetrischer KMS-Schlüssel. Informationen zum Erstellen eines neuen, vom Kunden verwalteten Schlüssels finden Sie unter Erstellen von KMS-Schlüsseln mit symmetrischer Verschlüsselung im AWS Key Management Service Entwicklerhandbuch.

CodeArtifact unterstützt AWS KMS externe Schlüsselspeicher (XKS). Sie sind verantwortlich für die Verfügbarkeit, Beständigkeit und Latenz wichtiger Operationen mit XKS-Schlüsseln, was sich auf die Verfügbarkeit, Haltbarkeit und Latenz von auswirken kann. CodeArtifact Einige Beispiele für Auswirkungen der Verwendung von XKS-Schlüsseln mit: CodeArtifact

- Da jedes Asset eines angeforderten Pakets und all seine Abhängigkeiten einer Entschlüsselungslatenz unterliegen, kann die Build-Latenz mit einer Erhöhung der XKS-Operationslatenz erheblich erhöht werden.
- Da alle Ressourcen verschlüsselt sind CodeArtifact, führt ein Verlust von XKS-Schlüsselmaterialien zum Verlust aller Ressourcen, die mit der Domäne verknüpft sind, die den XKS-Schlüssel verwendet.

Weitere Informationen zu XKS-Schlüsseln finden Sie unter Externe Schlüsselspeicher im AWS Key Management Service Entwicklerhandbuch.

Domain erstellen

Sie können eine Domäne mithilfe der CodeArtifact Konsole, der Taste AWS Command Line Interface (AWS CLI) oder AWS CloudFormation erstellen. Wenn Sie eine Domain erstellen, enthält sie keine Repositorys. Weitere Informationen finden Sie unter Erstellen eines -Repositorys. Weitere Informationen zur Verwaltung von CodeArtifact Domains mit finden Sie CloudFormation unter CodeArtifact Ressourcen erstellen mit AWS CloudFormation.

Themen

- Erstellen Sie eine Domäne (Konsole)
- Erstellen Sie eine Domäne (AWS CLI)
- Beispiel für eine AWS KMS Schlüsselrichtlinie

Erstellen Sie eine Domäne (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann Create domain aus.
3. Geben Sie im Feld Name einen Namen für Ihre Domain ein.
4. Erweitern Sie Additional configuration (Zusätzliche Konfiguration).
5. Verwenden Sie einen AWS KMS key (KMS-Schlüssel), um alle Ressourcen in Ihrer Domain zu verschlüsseln. Sie können einen AWS verwalteten KMS-Schlüssel oder einen KMS-Schlüssel verwenden, den Sie verwalten. Weitere Informationen zu den unterstützten Typen von KMS-Schlüsseln finden Sie unterArten von AWS KMS Schlüsseln, die unterstützt werden in CodeArtifact. CodeArtifact

   - Wählen Sie AWS-verwalteten Schlüssel, wenn Sie den Standard verwenden möchten Von AWS verwalteter Schlüssel.
   - Wählen Sie Vom Kunden verwalteter Schlüssel, wenn Sie einen von Ihnen verwalteten KMS-Schlüssel verwenden möchten. Um einen KMS-Schlüssel zu verwenden, den Sie verwalten, suchen Sie unter ARN für vom Kunden verwalteten Schlüssel nach dem KMS-Schlüssel und wählen Sie ihn aus.

Weitere Informationen finden Sie unter Von AWS verwalteter Schlüsselund vom Kunden verwalteter Schlüssel im AWS Key Management Service Entwicklerhandbuch.

6. Wählen Sie Domain erstellen aus.

Erstellen Sie eine Domäne (AWS CLI)

Um eine Domain mit dem zu erstellen AWS CLI, verwenden Sie den `create-domain` Befehl. Sie müssen einen AWS KMS key (KMS-Schlüssel) verwenden, um alle Ressourcen in Ihrer Domain zu verschlüsseln. Sie können einen AWS verwalteten KMS-Schlüssel oder einen von Ihnen verwalteten KMS-Schlüssel verwenden. Wenn Sie einen AWS verwalteten KMS-Schlüssel verwenden, verwenden Sie den `--encryption-key` Parameter nicht.

Weitere Informationen zu den unterstützten Typen von KMS-Schlüsseln finden Sie unterArten von AWS KMS Schlüsseln, die unterstützt werden in CodeArtifact. CodeArtifact Weitere Informationen zu KMS-Schlüsseln finden Sie unter Von AWS verwalteter Schlüsselund vom Kunden verwaltete Schlüssel im AWS Key Management Service Entwicklerhandbuch.

```
aws codeartifact create-domain --domain my_domain
```

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrer neuen Domain angezeigt.

```
{
    "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"
    }
}
```

Wenn Sie einen KMS-Schlüssel verwenden, den Sie verwalten, geben Sie dessen Amazon-Ressourcennamen (ARN) in den `--encryption-key` Parameter ein.

```
aws codeartifact create-domain --domain my_domain --encryption-key arn:aws:kms:us-
west-2:111122223333:key/your-kms-key
```

In der Ausgabe werden Daten im JSON-Format mit Details zu Ihrer neuen Domain angezeigt.

```
{
    "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"
    }
}
```

Erstellen einer Domäne mit Tags

Um eine Domain mit Tags zu erstellen, fügen Sie den `--tags` Parameter zu Ihrem Befehl hinzu. `create-domain`

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

Beispiel für eine AWS KMS Schlüsselrichtlinie

Wenn Sie eine Domäne in erstellen CodeArtifact, verwenden Sie einen KMS-Schlüssel, um alle Ressourcen in der Domäne zu verschlüsseln. Sie können einen AWS verwalteten KMS-Schlüssel oder einen vom Kunden verwalteten Schlüssel wählen, den Sie verwalten. Weitere Informationen zu KMS-Schlüsseln finden Sie im AWS Key Management Service Entwicklerhandbuch.

Um einen vom Kunden verwalteten Schlüssel verwenden zu können, muss Ihr KMS-Schlüssel über eine Schlüsselrichtlinie verfügen, die Zugriff darauf gewährt CodeArtifact. Eine Schlüsselrichtlinie ist eine Ressourcenrichtlinie für einen AWS KMS Schlüssel und stellt die wichtigste Methode zur Steuerung des Zugriffs auf KMS-Schlüssel dar. Jeder KMS-Schlüssel muss genau eine Schlüsselrichtlinie haben. Die Anweisungen im Schlüsselrichtliniendokument legen fest, wer über eine Berechtigung zur Verwendung des KMS-Schlüssels verfügt, und wie diese Verwendung erfolgen kann.

Das folgende Beispiel für eine wichtige Richtlinienerklärung ermöglicht es AWS CodeArtifact , Zuschüsse zu erstellen und wichtige Details im Namen autorisierter Benutzer einzusehen. Diese Grundsatzerklärung beschränkt die Erlaubnis darauf, im Namen der angegebenen Konto-ID zu CodeArtifact handeln, indem die Bedingungsschlüssel `kms:ViaService` und die `kms:CallerAccount` Bedingungsschlüssel verwendet werden. Außerdem werden dem IAM-Root-Benutzer alle AWS KMS Berechtigungen erteilt, sodass der Schlüssel nach seiner Erstellung verwaltet werden kann.

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

Domäne löschen

Sie können eine Domain mit der CodeArtifact Konsole oder der AWS Command Line Interface (AWS CLI) löschen.

Themen

- Einschränkungen beim Löschen von Domains
- Eine Domain löschen (Konsole)
- Löschen Sie eine Domäne (AWS CLI)

Einschränkungen beim Löschen von Domains

Normalerweise können Sie eine Domain, die Repositorys enthält, nicht löschen. Bevor Sie die Domain löschen, müssen Sie zuerst ihre Repositorys löschen. Weitere Informationen finden Sie unter Löschen Sie ein Repository.

Wenn Sie jedoch CodeArtifact keinen Zugriff mehr auf den KMS-Schlüssel der Domäne haben, können Sie die Domäne löschen, auch wenn sie noch Repositorys enthält. Diese Situation tritt ein, wenn Sie den KMS-Schlüssel der Domäne löschen oder die KMS-Genehmigung widerrufen, die für den Zugriff auf den Schlüssel CodeArtifact verwendet wird. In diesem Zustand können Sie nicht auf die Repositorys in der Domäne oder die darin gespeicherten Pakete zugreifen. Das Auflisten und Löschen von Repositorys ist auch nicht möglich, wenn CodeArtifact kein Zugriff auf den KMS-Schlüssel der Domäne möglich ist. Aus diesem Grund wird beim Löschen von Domänen nicht geprüft, ob die Domain Repositorys enthält, wenn auf den KMS-Schlüssel der Domain nicht zugegriffen werden kann.

> **Note**
>
> Wenn eine Domain gelöscht wird, die noch Repositorys enthält, CodeArtifact werden die Repositorys innerhalb von 15 Minuten asynchron gelöscht. Nach dem Löschen der Domain sind die Repositorys weiterhin in der CodeArtifact Konsole und in der `list-repositories` Befehlsausgabe sichtbar, bis die automatische Repository-Bereinigung erfolgt.

Eine Domain löschen (Konsole)

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Domains und dann die Domain aus, die Sie löschen möchten.
3. Wählen Sie Löschen.

Löschen Sie eine Domäne (AWS CLI)

Verwenden Sie den delete-domain Befehl, um eine Domain zu löschen.

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

In der Ausgabe werden Daten im JSON-Format mit Details zur gelöschten Domain angezeigt.

```
{
    "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"
    }
}
```

Domain-Richtlinien

CodeArtifact unterstützt die Verwendung ressourcenbasierter Berechtigungen zur Zugriffskontrolle. Mit ressourcenbasierten Berechtigungen können Sie angeben, wer Zugriff auf eine Ressource hat und welche Aktionen sie mit ihr ausführen können. Standardmäßig kann nur das AWS-Konto, dem die Domain gehört, Repositorys in der Domain erstellen und darauf zugreifen. Sie können ein Richtliniendokument auf eine Domain anwenden, um anderen IAM-Prinzipalen den Zugriff darauf zu ermöglichen.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen und Identitätsbasierte Richtlinien und Ressourcenbasierte Richtlinien.

Themen

- Aktivieren Sie den kontoübergreifenden Zugriff auf eine Domain
- Beispiel für eine Domänenrichtlinie
- Beispiel für eine Domänenrichtlinie mit AWS Organizations
- Legen Sie eine Domain-Richtlinie fest
- Lesen Sie eine Domänenrichtlinie
- Löschen Sie eine Domänenrichtlinie

Aktivieren Sie den kontoübergreifenden Zugriff auf eine Domain

Eine Ressourcenrichtlinie ist eine Textdatei im JSON-Format. Die Datei muss einen Prinzipal (Akteur), eine oder mehrere Aktionen und einen Effekt (`AllowoderDeny`) angeben. Um ein Repository in einer Domäne zu erstellen, die einem anderen Konto gehört, muss dem Prinzipal die entsprechenden `CreateRepository` Berechtigungen für die Domänenressource erteilt werden.

Die folgende Ressourcenrichtlinie gewährt dem Konto beispielsweise die `123456789012` Berechtigung, ein Repository in der Domäne zu erstellen.

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

Um das Erstellen von Repositorys mit Tags zu ermöglichen, müssen Sie die `codeartifact:TagResource` entsprechende Berechtigung angeben. Dadurch erhält das Konto auch Zugriff auf das Hinzufügen von Tags zur Domain und allen darin enthaltenen Repositorys.

Die Domänenrichtlinie wird für alle Operationen mit der Domain und allen Ressourcen innerhalb der Domain bewertet. Das bedeutet, dass die Domänenrichtlinie verwendet werden kann, um Berechtigungen auf Repositorys und Pakete in der Domäne anzuwenden. Wenn das `Resource` Element auf gesetzt ist*, gilt die Anweisung für alle Ressourcen in der Domäne. Wenn die obige Richtlinie beispielsweise auch `codeartifact:DescribeRepository` in der Liste der zulässigen IAM-Aktionen enthalten wäre, würde die Richtlinie den Aufruf `DescribeRepository` jedes Repositorys in der Domäne zulassen. Eine Domänenrichtlinie kann verwendet werden, um Berechtigungen auf bestimmte Ressourcen in der Domäne anzuwenden, indem bestimmte Ressourcen ARNs in dem `Resource` Element verwendet werden.

> **Note**
>
> Sowohl Domänen- als auch Repository-Richtlinien können zur Konfiguration von Berechtigungen verwendet werden. Wenn beide Richtlinien vorhanden sind, werden beide Richtlinien bewertet und eine Aktion ist zulässig, sofern eine der Richtlinien dies zulässt. Weitere Informationen finden Sie unter Interaktion zwischen Repository- und Domain-Richtlinien.

Um auf Pakete in einer Domäne zuzugreifen, die einem anderen Konto gehört, muss einem Prinzipal die entsprechenden `GetAuthorizationToken` Berechtigungen für die Domänenressource erteilt werden. Auf diese Weise kann der Domaininhaber kontrollieren, welche Konten die Inhalte von Repositorys in der Domain lesen können.

Die folgende Ressourcenrichtlinie gewährt dem Konto beispielsweise die `123456789012` Erlaubnis, ein Authentifizierungstoken für jedes Repository in der Domain abzurufen.

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

```
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Einem Principal, der Pakete von einem Repository-Endpunkt abrufen möchte, muss zusätzlich zu der `ReadFromRepository` Berechtigung für die Domäne auch die Berechtigung für die Repository-Ressource `GetAuthorizationToken` erteilt werden. In ähnlicher Weise muss einem Prinzipal, der Pakete auf einem Repository-Endpunkt veröffentlichen möchte, zusätzlich zu `GetAuthorizationToken` die `PublishPackageVersion` entsprechende Berechtigung erteilt werden.
> Weitere Informationen zu den `PublishPackageVersion` Berechtigungen `ReadFromRepository` und finden Sie unter Repository-Richtlinien.

Beispiel für eine Domänenrichtlinie

Wenn mehrere Konten eine Domäne verwenden, sollten den Konten grundlegende Berechtigungen erteilt werden, damit sie die Domäne vollständig nutzen können. Die folgende Ressourcenrichtlinie listet eine Reihe von Berechtigungen auf, die die vollständige Nutzung der Domäne ermöglichen.

```
{
    "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**
>
> Sie müssen keine Domänenrichtlinie erstellen, wenn eine Domäne und alle zugehörigen Repositorys einem einzigen Konto gehören und nur von diesem Konto aus verwendet werden müssen.

Beispiel für eine Domänenrichtlinie mit AWS Organizations

Sie können den `aws:PrincipalOrgID` Bedingungsschlüssel verwenden, um von allen Konten in Ihrer Organisation aus Zugriff auf eine CodeArtifact Domain zu gewähren. Gehen Sie dazu wie folgt vor.

```
{
    "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-xxxxxxxxxxx"]}
        }
    }
}
```

Weitere Informationen zur Verwendung des `aws:PrincipalOrgID` Bedingungsschlüssels finden Sie unter AWS Global Condition Context Keys im IAM-Benutzerhandbuch.

Legen Sie eine Domain-Richtlinie fest

Sie können den `put-domain-permissions-policy` Befehl verwenden, um eine Richtlinie an eine Domain anzuhängen.

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

Wenn Sie aufrufen`put-domains-permissions-policy`, wird die Ressourcenrichtlinie für die Domain bei der Auswertung von Berechtigungen ignoriert. Dadurch wird sichergestellt, dass sich der Besitzer einer Domain nicht selbst aus der Domain aussperren kann, was ihn daran hindern würde, die Ressourcenrichtlinie zu aktualisieren.

> **Note**
>
> Sie können einem anderen AWS Konto keine Berechtigungen zum Aktualisieren der Ressourcenrichtlinie für eine Domäne mithilfe einer Ressourcenrichtlinie gewähren, da die Ressourcenrichtlinie beim Aufrufen ignoriert wird put-domain-permissions-policy.

Beispielausgabe:

```
{
    "policy": {
        "resourceArn": "arn:aws:codeartifact:region-id:111122223333:domain/my_domain",
        "document": "{ ...policy document content...}",
        "revision": "MQlyyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxx="
    }
}
```

Die Ausgabe des Befehls enthält den Amazon-Ressourcennamen (ARN) der Domain-Ressource, den vollständigen Inhalt des Richtliniendokuments und eine Revisions-ID. Die Revisions-ID kann `put-domain-permissions-policy` mithilfe der `--policy-revision` Option an übergeben werden. Dadurch wird sichergestellt, dass eine bekannte Version des Dokuments überschrieben wird und nicht eine neuere Version, die von einem anderen Autor festgelegt wurde.

Lesen Sie eine Domänenrichtlinie

Verwenden Sie den `get-domain-permissions-policy` Befehl, um eine vorhandene Version eines Richtliniendokuments zu lesen. Um die Ausgabe aus Gründen der Lesbarkeit zu formatieren, verwenden Sie das `--output` und `--query policy.document` zusammen mit dem `json.tool` Python-Modul wie folgt.

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

Beispielausgabe:

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

Löschen Sie eine Domänenrichtlinie

Verwenden Sie den `delete-domain-permissions-policy` Befehl, um eine Richtlinie aus einer Domain zu löschen.

```
aws codeartifact delete-domain-permissions-policy --domain my_domain --domain-
owner 111122223333
```

Das Format der Ausgabe entspricht dem der `delete-domain-permissions-policy` Befehle `get-domain-permissions-policy` und.

Kennzeichnen Sie eine Domain in CodeArtifact

Tags sind mit AWS-Ressourcen verknüpfte Schlüssel-Wert-Paare. In können Sie Tags auf Ihre Domains anwenden CodeArtifact. Informationen zu CodeArtifact Ressourcen-Tagging, Anwendungsfällen, Einschränkungen für Tagschlüssel und -werte sowie zu unterstützten Ressourcentypen finden Sie unterTaggen von -Ressourcen.

Sie können die CLI verwenden, um Tags anzugeben, wenn Sie eine Domain erstellen. Sie können die Konsole oder CLI verwenden, um Tags hinzuzufügen oder zu entfernen und die Werte von Tags in einer Domain zu aktualisieren. Sie können jeder Domain bis zu 50 Tags hinzufügen.

Themen

- Tag-Domains (CLI)
- Domains taggen (Konsole)

Tag-Domains (CLI)

Sie können die CLI verwenden, um Domain-Tags zu verwalten.

Themen

- Hinzufügen von Tags zu einer Domain (CLI)
- Tags für eine Domain anzeigen (CLI)
- Tags für eine Domain bearbeiten (CLI)
- Tags aus einer Domain entfernen (CLI)

Hinzufügen von Tags zu einer Domain (CLI)

Sie können die Konsole oder die verwenden, um Domains AWS CLI zu taggen.

Informationen zum Hinzufügen eines Tags zu einer Domain, wenn Sie sie erstellen, finden Sie unterErstellen eines -Repositorys.

Bei diesen Schritten wird davon ausgegangen, dass Sie bereits eine aktuelle Version der AWS CLI installiert oder eine Aktualisierung auf die aktuelle Version vorgenommen haben. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface.

Führen Sie den Befehl am Terminal oder in der tag-resource Befehlszeile aus und geben Sie den Amazon-Ressourcennamen (ARN) der Domain an, der Sie Tags hinzufügen möchten, sowie den Schlüssel und den Wert des Tags, das Sie hinzufügen möchten.

> **Note**
>
> Führen Sie den `describe-domain` folgenden Befehl aus, um den ARN der Domain abzurufen:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Sie können einer Domain mehr als ein Tag hinzufügen. Um beispielsweise eine Domain *my_domain* mit zwei Tags zu taggen, einem Tag-Schlüssel *key1* mit dem Tag-Wert von *value1* und einem Tag-Schlüssel *key2* mit dem Tag-Wert von*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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags für eine Domain anzeigen (CLI)

Gehen Sie wie folgt vor AWS CLI , um die AWS Tags für eine Domain anzuzeigen. Wenn keine Tags hinzugefügt wurden, ist die zurückgegebene Liste leer.

Führen Sie im Terminal oder in der Befehlszeile den list-tags-for-resource Befehl mit dem Amazon-Ressourcennamen (ARN) der Domain aus.

> **Note**
>
> Führen Sie den `describe-domain` folgenden Befehl aus, um den ARN der Domain abzurufen:

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

Um beispielsweise eine Liste von Tag-Schlüsseln und Tag-Werten für eine Domain anzuzeigen, die *my_domain* mit dem `arn:aws:codeartifact:`*us-west-2*`:`*123456789012*`:domain/`*my_domain* ARN-Wert benannt ist:

```
aws codeartifact list-tags-for-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain
```

Bei erfolgreicher Ausführung gibt dieser Befehl etwa wie folgt aussehende Informationen zurück:

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

Tags für eine Domain bearbeiten (CLI)

Gehen Sie wie folgt vor AWS CLI , um ein Tag für eine Domain zu bearbeiten. Sie können den Wert für einen vorhandenen Schlüssel ändern oder einen anderen Schlüssel hinzufügen. Sie können Tags auch aus einer Domain entfernen, wie im nächsten Abschnitt gezeigt.

Führen Sie im Terminal oder in der Befehlszeile den tag-resource Befehl aus und geben Sie den ARN der Domain an, in der Sie ein Tag aktualisieren möchten, und geben Sie den Tag-Schlüssel und den Tag-Wert an:

> **Note**
>
> Führen Sie den `describe-domain` folgenden Befehl aus, um den ARN der Domain abzurufen:
>
> ```
> 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
```

Bei Erfolg hat dieser Befehl keine Ausgabe.

Tags aus einer Domain entfernen (CLI)

Gehen Sie wie folgt vor AWS CLI , um ein Tag aus einer Domain zu entfernen.

> **Note**
>
> Wenn Sie eine Domain löschen, werden alle Tag-Verknüpfungen aus der gelöschten Domain entfernt. Sie müssen keine Tags entfernen, bevor Sie eine Domain löschen.

Führen Sie im Terminal oder in der Befehlszeile den untag-resource Befehl aus und geben Sie den ARN der Domain an, aus der Sie Tags entfernen möchten, und den Tag-Schlüssel des Tags, das Sie entfernen möchten.

> **Note**
>
> Führen Sie den `describe-domain` folgenden Befehl aus, um den ARN der Domain abzurufen:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Um beispielsweise mehrere Tags auf einer Domain zu entfernen, die *mydomain* mit den Tag-Schlüsseln benannt ist *key1* und*key2*:

```
aws codeartifact untag-resource --resource-arn arn:aws:codeartifact:us-
west-2:123456789012:domain/my_domain --tag-keys key1 key2
```

Wenn dieser Befehl erfolgreich ist, hat er keine Ausgabe. Nachdem Sie die Tags entfernt haben, können Sie mit dem `list-tags-for-resource` Befehl die verbleibenden Tags auf der Domain anzeigen.

Domains taggen (Konsole)

Sie können Ressourcen über die Konsole oder CLI mit Tags markieren.

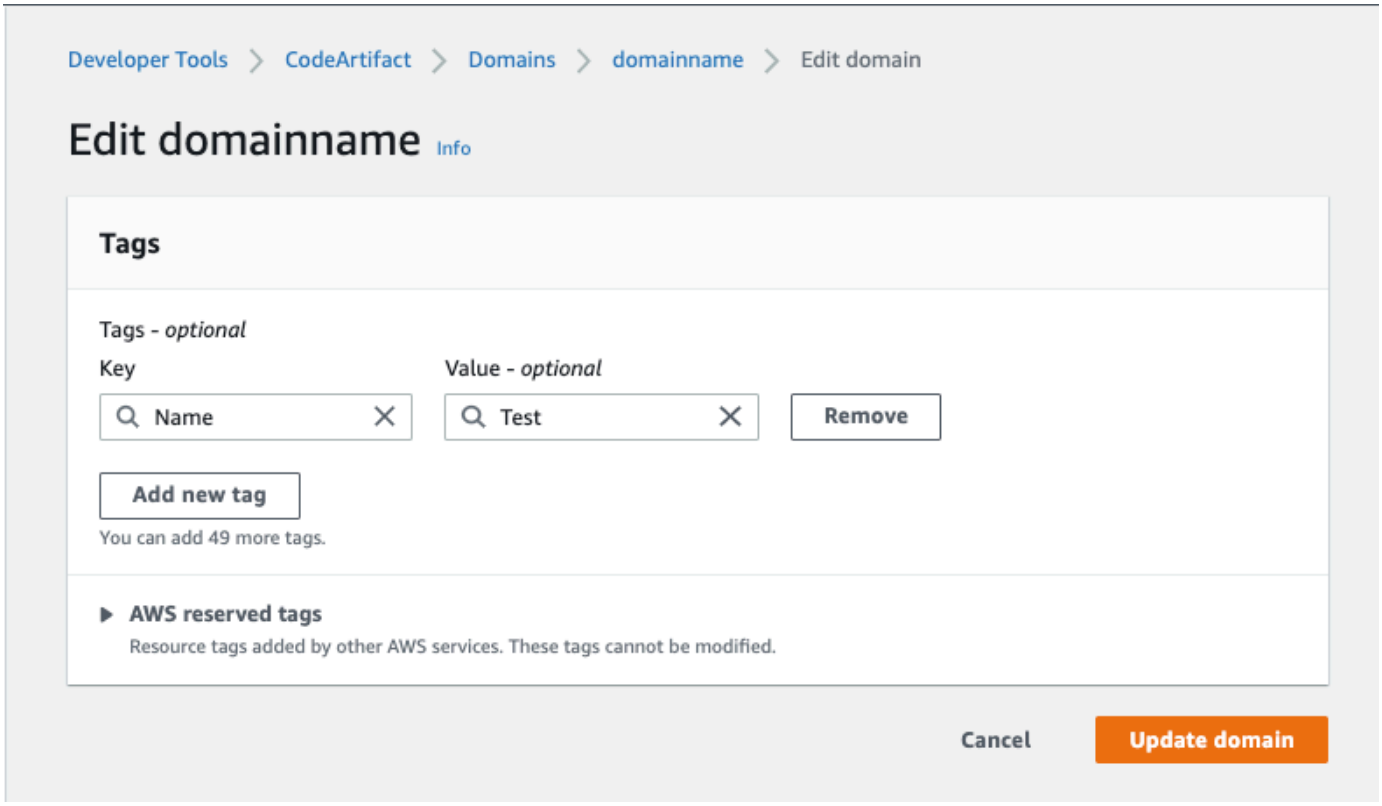
Themen

- Fügen Sie einer Domain Tags hinzu (Konsole)
- Tags für eine Domain anzeigen (Konsole)
- Bearbeiten Sie die Tags für eine Domain (Konsole)
- Tags aus einer Domain entfernen (Konsole)

Fügen Sie einer Domain Tags hinzu (Konsole)

Sie können die Konsole verwenden, um einer vorhandenen Domain Tags hinzuzufügen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Domains die Domain aus, der Sie Tags hinzufügen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Domain-Tags die Option Domain-Tags hinzufügen, wenn die Domain keine Tags enthält, oder wählen Sie Domain-Tags anzeigen und bearbeiten, falls vorhanden.
5. Wählen Sie Neues Tag hinzufügen aus.
6. Geben Sie in den Feldern Schlüssel und Wert den Text für jedes Tag ein, das Sie hinzufügen möchten. (Das Feld Value (Wert) ist optional.) Geben Sie beispielsweise für Key (Schlüssel) **`Name`** ein. Geben Sie unter Value (Wert) **`Test`** ein.



7. (Optional) Wählen Sie Add tag (Tag hinzufügen) aus, um weitere Zeilen hinzuzufügen und weitere Tags einzugeben.
8. Wählen Sie Domain aktualisieren.

Tags für eine Domain anzeigen (Konsole)

Sie können die Konsole verwenden, um Tags für bestehende Domains aufzulisten.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Domains die Domain aus, für die Sie Tags anzeigen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Domain-Tags die Option Domain-Tags anzeigen und bearbeiten aus.

> **Note**
>
> Wenn zu dieser Domain keine Tags hinzugefügt wurden, wird in der Konsole Domain-Tags hinzufügen angezeigt.

Bearbeiten Sie die Tags für eine Domain (Konsole)

Sie können die Konsole verwenden, um Tags zu bearbeiten, die der Domain hinzugefügt wurden.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Domains die Domain aus, für die Sie Tags aktualisieren möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Domain-Tags die Option Domain-Tags anzeigen und bearbeiten aus.

   > **Note**
   >
   > Wenn zu dieser Domain keine Tags hinzugefügt wurden, wird in der Konsole Domain-Tags hinzufügen angezeigt.

5. Aktualisieren Sie in den Feldern Key (Schlüssel) und Value (Wert) die Werte nach Bedarf. Ändern Sie beispielsweise für den Schlüssel **`Name`** unter Value (Wert) die Angabe **`Test`** in **`Prod`**.
6. Wählen Sie „Domain aktualisieren“.

Tags aus einer Domain entfernen (Konsole)

Sie können die Konsole verwenden, um Tags aus Domains zu löschen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie auf der Seite Domains die Domain aus, für die Sie Tags entfernen möchten.
3. Erweitern Sie den Abschnitt Details.
4. Wählen Sie unter Domain-Tags die Option Domain-Tags anzeigen und bearbeiten aus.

   > **Note**
   >
   > Wenn zu dieser Domain keine Tags hinzugefügt wurden, wird in der Konsole Domain-Tags hinzufügen angezeigt.

5. Wählen Sie neben dem Schlüssel und dem Wert für jedes Tag, das Sie löschen möchten, die Option Entfernen aus.
6. Wählen Sie „Domain aktualisieren“.

Verwendung CodeArtifact mit Cargo

Diese Themen beschreiben, wie Sie Cargo, den Rust-Paketmanager, mit verwenden CodeArtifact.

> **Note**
>
> CodeArtifact unterstützt nur Cargo 1.74.0 und höher. Cargo 1.74.0 ist die früheste Version, die die Authentifizierung in einem CodeArtifact Repository unterstützt.

Themen

- Konfigurieren und verwenden Sie Cargo mit CodeArtifact
- Unterstützung von Cargo Command

Konfigurieren und verwenden Sie Cargo mit CodeArtifact

Sie können Cargo verwenden, um Kisten aus CodeArtifact Repositorys zu veröffentlichen und herunterzuladen oder um Kisten von crates.io, der Crate-Registry der Rust-Community, abzurufen. In diesem Thema wird beschrieben, wie Cargo so konfiguriert wird, dass es sich bei einem Repository authentifiziert und es verwendet. CodeArtifact

Konfigurieren Sie Cargo mit CodeArtifact

Um Cargo zum Installieren und Veröffentlichen von Crates zu verwenden AWS CodeArtifact, müssen Sie sie zunächst mit Ihren CodeArtifact Repository-Informationen konfigurieren. Folgen Sie den Schritten in einem der folgenden Verfahren, um Cargo mit Ihren CodeArtifact Repository-Endpunktinformationen und Anmeldeinformationen zu konfigurieren.

Konfigurieren Sie Cargo mithilfe der Konsolenanweisungen

Sie können die Konfigurationsanweisungen in der Konsole verwenden, um Cargo mit Ihrem CodeArtifact Repository zu verbinden. Die Konsolenanweisungen bieten eine auf Ihr CodeArtifact Repository zugeschnittene Cargo-Konfiguration. Sie können diese benutzerdefinierte Konfiguration verwenden, um Cargo einzurichten, ohne Ihre CodeArtifact Informationen suchen und eingeben zu müssen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Repositories und dann ein Repository aus, um eine Verbindung zu Cargo herzustellen.
3. Wählen Sie Verbindungsanweisungen anzeigen.
4. Wählen Sie Ihr Betriebssystem.
5. Wählen Sie Cargo.
6. Folgen Sie den generierten Anweisungen, um Cargo mit Ihrem CodeArtifact Repository zu verbinden.

Konfigurieren Sie Cargo manuell

Wenn Sie die Konfigurationsanweisungen von der Konsole aus nicht verwenden können oder wollen, können Sie die folgenden Anweisungen verwenden, um Cargo manuell mit Ihrem CodeArtifact Repository zu verbinden.

macOS and Linux

Um Cargo mit zu konfigurieren CodeArtifact, müssen Sie Ihr CodeArtifact Repository in der Cargo-Konfiguration als Registry definieren und Anmeldeinformationen angeben.

- *my_registry*Ersetzen Sie es durch Ihren Registrierungsnamen.
- Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
- *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
- Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

Kopieren Sie die Konfiguration, um Cargo-Pakete zu veröffentlichen und in Ihr Repository herunterzuladen, und speichern Sie sie in der `~/.cargo/config.toml` Datei für eine Konfiguration auf Systemebene oder `.cargo/config.toml` für eine Konfiguration auf Projektebene:

```
[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

Um Cargo mit zu konfigurieren CodeArtifact, müssen Sie Ihr CodeArtifact Repository in der Cargo-Konfiguration als Registrierung definieren und Anmeldeinformationen angeben.

- *my_registry*Ersetzen Sie es durch Ihren Registrierungsnamen.
- Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
- *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
- Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

Kopieren Sie die Konfiguration, um nur Cargo-Pakete aus Ihrem Repository herunterzuladen, und speichern Sie sie in der `%USERPROFILE%\.cargo\config.toml` Datei für eine Konfiguration auf Systemebene oder `.cargo\config.toml` für eine Konfiguration auf Projektebene:

```
[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. Um Cargo mit zu konfigurieren CodeArtifact, müssen Sie Ihr CodeArtifact Repository in der Cargo-Konfiguration als Registrierung definieren und Anmeldeinformationen angeben.

   - *my_registry*Ersetzen Sie es durch Ihren Registrierungsnamen.
   - Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
   - *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
   - Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

   Kopieren Sie die Konfiguration, um Cargo-Pakete zu veröffentlichen und in Ihr Repository herunterzuladen, und speichern Sie sie in der `%USERPROFILE%\.cargo\config.toml` Datei für eine Konfiguration auf Systemebene oder `.cargo\config.toml` für eine Konfiguration auf Projektebene.

   Es wird empfohlen, den Anmeldeinformationsanbieter zu verwenden`cargo:token`, der die in Ihrer Datei gespeicherten Anmeldeinformationen verwendet. `~/.cargo/credentials.toml` `cargo publish`Bei der Verwendung kann ein Fehler auftreten, `cargo:token-from-stdout` da der Cargo-Client das Autorisierungstoken `cargo publish` währenddessen nicht richtig kürzt.

   ```
   [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. Um Cargo-Pakete mit Windows in Ihrem Repository zu veröffentlichen, müssen Sie den CodeArtifact `get-authorization-token` Befehl und den `login` Cargo-Befehl verwenden, um ein Autorisierungstoken und Ihre Anmeldeinformationen abzurufen.

- *my_registry*Ersetzen Sie es durch Ihren Registrierungsnamen, wie unter definiert. `[registries.my_registry]`
- *my_domain*Ersetzen Sie es durch Ihren CodeArtifact Domainnamen.
- *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

```
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**
>
> Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

Der `[registries.my_registry]` Abschnitt im vorherigen Beispiel definiert eine Registrierung mit *my_registry* `index` und stellt `credential-provider` Informationen bereit.

- `index`gibt die URL des Indexes für Ihre Registrierung an. Dabei handelt es sich um den CodeArtifact Repository-Endpunkt, der mit a endet`/`. Das `sparse+` Präfix ist für Registries erforderlich, die keine Git-Repositorys sind.

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den Endpunkt. `codeartifact.region.on.aws`

- `credential-provider`gibt den Anmeldeinformationsanbieter für die angegebene Registrierung an. Wenn `credential-provider` nicht gesetzt, `registry.global-credential-providers` werden die angegebenen Anbieter verwendet. Wenn Sie `credential-provider` auf setzen`cargo:token-from-stdout`, ruft der Cargo-Client beim Veröffentlichen oder Herunterladen aus Ihrem CodeArtifact Repository automatisch ein neues Autorisierungstoken ab, sodass Sie das Autorisierungstoken nicht alle 12 Stunden manuell aktualisieren müssen.

Der `[registry]` Abschnitt definiert die verwendete Standardregistrierung.

- `default`gibt den Namen der Registrierung an`[registries.my_registry]`, der in definiert ist und standardmäßig beim Veröffentlichen oder Herunterladen aus Ihrem CodeArtifact Repository verwendet werden soll.

Der `[source.crates-io]` Abschnitt definiert die Standardregistrierung, die verwendet wird, wenn keine angegeben ist.

- `replace-with = "my_registry"`ersetzt die öffentliche Registrierung crates.io durch Ihr unter definiertes CodeArtifact Repository. `[registries.my_registry]` Diese Konfiguration wird empfohlen, wenn Sie Pakete über die externe Verbindung anfordern müssen, z. B. crates.io.

  Um alle Vorteile nutzen zu können CodeArtifact, wie z. B. die Kontrolle des Paketursprungs, die Angriffe zur Konfusion von Abhängigkeiten verhindert, wird empfohlen, den Quellcode zu ersetzen. Beim Austausch der Quelle werden alle Anfragen CodeArtifact per Proxy an die externe Verbindung weitergeleitet und das Paket von der externen Verbindung in Ihr Repository kopiert. Ohne den Ersatz der Quelle ruft der Cargo-Client das Paket direkt auf der Grundlage der Konfiguration in Ihrer `Cargo.toml` Datei in Ihrem Projekt ab. Wenn eine Abhängigkeit nicht mit markiert ist`registry=my_registry`, ruft der Cargo-Client sie direkt von crates.io ab, ohne mit Ihrem Repository zu kommunizieren. CodeArtifact

  > **Note**
  >
  > Wenn Sie anfangen, den Quellenersatz zu verwenden und dann Ihre Konfigurationsdatei so aktualisieren, dass der Quellenersatz nicht verwendet wird, können Fehler auftreten. Das gegenteilige Szenario kann ebenfalls zu Fehlern führen. Es wird daher empfohlen, die Konfiguration für Ihr Projekt nicht zu ändern.

Frachtkisten installieren

Gehen Sie wie folgt vor, um Cargo-Kisten aus einem CodeArtifact Repository oder von crates.io zu installieren.

Installiere Frachtkisten von CodeArtifact

Sie können die Cargo (`cargo`) CLI verwenden, um schnell eine bestimmte Version einer Cargo-Kiste aus Ihrem CodeArtifact Repository zu installieren.

Um Cargo-Kisten aus einem CodeArtifact Repository zu installieren mit **`cargo`**

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie die `cargo` CLI so, dass Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet wird. Konfigurieren und verwenden Sie Cargo mit CodeArtifact
2. Verwenden Sie den folgenden Befehl, um Cargo Crates von CodeArtifact zu installieren:

   ```
   cargo add my_cargo_package@1.0.0
   ```

   Weitere Informationen finden Sie unter Cargo Add in The Cargo Book.

Frachtkisten veröffentlichen auf CodeArtifact

Gehen Sie wie folgt vor, um Cargo-Kisten mithilfe der `cargo` CLI in einem CodeArtifact Repository zu veröffentlichen.

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie die `cargo` CLI so, dass Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet wird. Konfigurieren und verwenden Sie Cargo mit CodeArtifact
2. Verwenden Sie den folgenden Befehl, um Cargo-Kisten in einem CodeArtifact Repository zu veröffentlichen:

   ```
   cargo publish
   ```

   Weitere Informationen finden Sie unter Cargo Publish in The Cargo Book.

Unterstützung von Cargo Command

In den folgenden Abschnitten werden die Cargo-Befehle zusammengefasst, die von CodeArtifact Repositorys unterstützt werden, zusätzlich zu bestimmten Befehlen, die nicht unterstützt werden.

Inhalt

- Unterstützte Befehle, für die der Zugriff auf die Registrierung erforderlich ist
- Befehle werden nicht unterstützt

Unterstützte Befehle, für die der Zugriff auf die Registrierung erforderlich ist

In diesem Abschnitt werden Cargo-Befehle aufgeführt, bei denen der Cargo-Client Zugriff auf die Registry benötigt, mit der er konfiguriert wurde. Es wurde verifiziert, dass diese Befehle korrekt funktionieren, wenn sie in einem CodeArtifact Repository aufgerufen werden.

Befehl	Beschreibung
bauen	Erstellt lokale Pakete und deren Abhängigk eiten.
überprüfen	Überprüft lokale Pakete und ihre Abhängigk eiten auf Fehler.
holen	Ruft die Abhängigkeiten eines Pakets ab.
veröffentlichen	Veröffentlicht ein Paket in der Registrierung.

Befehle werden nicht unterstützt

Diese Cargo-Befehle werden von CodeArtifact Repositorys nicht unterstützt.

Befehl	Beschreibung
Besitzer	Verwaltet die Besitzer der Kiste im Register.
search	Sucht in der Registrierung nach Paketen.

Verwendung CodeArtifact mit Maven

Das Maven-Repository-Format wird von vielen verschiedenen Sprachen verwendet, darunter Java, Kotlin, Scala und Clojure. Es wird von vielen verschiedenen Build-Tools unterstützt, darunter Maven, Gradle, Scala SBT, Apache Ivy und Leiningen.

Wir haben die Kompatibilität mit den folgenden Versionen getestet und bestätigt: CodeArtifact

- Letzte Maven-Version: 3.6.3.
- Die neueste Gradle-Version: 6.4.1. 5.5.1 wurde ebenfalls getestet.
- Die neueste Clojure-Version: 1.11.1 wurde ebenfalls getestet.

Themen

- CodeArtifact Mit Gradle verwenden
- CodeArtifact Mit MVN verwenden
- CodeArtifact Mit deps.edn verwenden
- Publizieren mit curl
- Verwenden Sie Maven-Prüfsummen
- Verwenden Sie Maven-Snapshots
- Maven-Pakete von Upstreams und externen Verbindungen anfordern
- Fehlerbehebung in Maven

CodeArtifact Mit Gradle verwenden

Nachdem Sie das CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen gespeichert haben, wie unter Ein Authentifizierungstoken mithilfe einer Umgebungsvariablen übergeben beschrieben, folgen Sie diesen Anweisungen, um Maven-Pakete aus einem Repository zu konsumieren und neue Pakete in einem Repository zu veröffentlichen. CodeArtifact

Themen

- Abhängigkeiten abrufen
- Plugins abrufen
- Veröffentlichen Sie Artefakte

- Führen Sie einen Gradle-Build in IntelliJ IDEA aus

Abhängigkeiten abrufen

Verwenden Sie das folgende Verfahren, um Abhängigkeiten aus CodeArtifact einem Gradle-Build abzurufen.

Um Abhängigkeiten aus CodeArtifact einem Gradle-Build abzurufen

1. Wenn nicht, erstellen und speichern Sie ein CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, indem Sie das Verfahren unter befolgen. Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen
2. Fügen Sie dem `maven` Abschnitt in der `build.gradle` Projektdatei einen `repositories` Abschnitt hinzu.

```
maven {
         url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
         credentials {
             username "aws"
             password System.env.CODEARTIFACT_AUTH_TOKEN
         }
}
```

Das `url` Beispiel oben ist der Endpunkt Ihres CodeArtifact Repositorys. Gradle verwendet den Endpunkt, um eine Verbindung zu Ihrem Repository herzustellen. Im Beispiel `my_domain` ist dies der Name Ihrer Domain, `111122223333` die ID des Eigentümers der Domain und `my_repo` der Name Ihres Repositorys. Sie können den Endpunkt eines Repositorys mit dem `get-repository-endpoint` AWS CLI Befehl abrufen.

Bei einem Repository, das *`my_repo`* innerhalb einer Domain mit dem Namen benannt ist*`my_domain`*, lautet der Befehl beispielsweise wie folgt:

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format maven
```

Der `get-repository-endpoint` Befehl gibt den Repository-Endpunkt zurück:

```
url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
```

Das `credentials` Objekt im vorherigen Beispiel enthält das CodeArtifact Authentifizierungstoken, das Sie in Schritt 1 erstellt haben und das Gradle zur Authentifizierung verwendet. CodeArtifact

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den Endpunkt. `codeartifact.region.on.aws`

3. (Optional) — Um das CodeArtifact Repository als einzige Quelle für Ihre Projektabhängigkeiten zu verwenden, entfernen Sie alle anderen Abschnitte in `repositories` von. `build.gradle` Wenn Sie mehr als ein Repository haben, durchsucht Gradle jedes Repository nach Abhängigkeiten in der Reihenfolge, in der sie aufgelistet sind.
4. Nachdem Sie das Repository konfiguriert haben, können Sie dem `dependencies` Abschnitt Projektabhängigkeiten mit der Standard-Gradle-Syntax hinzufügen.

```
dependencies {
    implementation 'com.google.guava:guava:27.1-jre'
    implementation 'commons-cli:commons-cli:1.4'
    testImplementation 'org.testng:testng:6.14.3'
}
```

Plugins abrufen

Standardmäßig löst Gradle Plugins aus dem öffentlichen Gradle Plugin Portal auf. Gehen Sie wie folgt vor, um Plugins aus einem CodeArtifact Repository abzurufen.

Um Plugins aus einem CodeArtifact Repository abzurufen

1. Falls nicht, erstellen und speichern Sie ein CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, indem Sie das Verfahren unter befolgen. Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen

2. Fügen Sie Ihrer `settings.gradle` Datei einen `pluginManagement` Block hinzu. Der `pluginManagement` Block muss vor allen anderen Anweisungen in erscheinen`settings.gradle`, siehe den folgenden Ausschnitt:

```
pluginManagement {
    repositories {
        maven {
            name 'my_repo'
            url
 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password System.env.CODEARTIFACT_AUTH_TOKEN
            }
        }
    }
}
```

Dadurch wird sichergestellt, dass Gradle Plugins aus dem angegebenen Repository auflöst. Das Repository muss über ein Upstream-Repository mit einer externen Verbindung zum Gradle-Plugin-Portal (z. B.`gradle-plugins-store`) verfügen, damit die häufig benötigten Gradle-Plugins für den Build verfügbar sind. Weitere Informationen finden Sie in der Gradle-Dokumentation.

Veröffentlichen Sie Artefakte

In diesem Abschnitt wird beschrieben, wie Sie eine mit Gradle erstellte Java-Bibliothek in einem CodeArtifact Repository veröffentlichen.

Fügen Sie zunächst das `maven-publish` Plugin zum `plugins` Abschnitt der `build.gradle` Projektdatei hinzu.

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

Als Nächstes fügen Sie der `build.gradle` Projektdatei einen `publishing` Abschnitt hinzu.

```
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
            }
        }
    }
}
```

Das `maven-publish` Plugin generiert eine POM-Datei auf der Grundlage von`groupId`, und`artifactId`, die im `publishing` Abschnitt `version` angegeben sind.

Wenn diese Änderungen abgeschlossen `build.gradle` sind, führen Sie den folgenden Befehl aus, um das Projekt zu erstellen und in das Repository hochzuladen.

```
./gradlew publish
```

Verwenden Sie diese Option, `list-package-versions` um zu überprüfen, ob das Paket erfolgreich veröffentlicht wurde.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven\
  --namespace com.company.framework --package my-package-name
```

Beispielausgabe:

```
{
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "example",
```

```
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Weitere Informationen finden Sie in den folgenden Themen auf der Gradle-Website:

- Java-Bibliotheken erstellen
- Ein Projekt als Modul veröffentlichen

Führen Sie einen Gradle-Build in IntelliJ IDEA aus

Sie können einen Gradle-Build in IntelliJ IDEA ausführen, der Abhängigkeiten abruft. CodeArtifact Um sich zu authentifizieren CodeArtifact, müssen Sie Gradle ein Autorisierungstoken zur Verfügung stellen. CodeArtifact Es gibt drei Methoden, um ein Authentifizierungstoken bereitzustellen.

- Methode 1: Speichern des Authentifizierungstokens in. `gradle.properties` Verwenden Sie diese Methode, wenn Sie den Inhalt der Datei überschreiben oder ihn erweitern können. `gradle.properties`
- Methode 2: Speichern des Authentifizierungstokens in einer separaten Datei. Verwenden Sie diese Methode, wenn Sie Ihre `gradle.properties` Datei nicht ändern möchten.
- Methode 3: Generieren eines neuen Authentifizierungstokens für jeden Lauf, indem es `aws` als Inline-Skript in `build.gradle` ausgeführt wird. Verwenden Sie diese Methode, wenn Sie möchten, dass das Gradle-Skript bei jedem Lauf ein neues Token abruft. Das Token wird nicht im Dateisystem gespeichert.

Token stored in gradle.properties

Methode 1: Speichern des Authentifizierungstokens in **`gradle.properties`**

> **Note**
>
> Das Beispiel zeigt die `gradle.properties` Datei in`GRADLE_USER_HOME`.

1. Aktualisieren Sie Ihre `build.gradle` Datei mit dem folgenden Codeausschnitt:

```
repositories {
    maven {
            url
 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username "aws"
                password "$codeartifactToken"
            }
    }
}
```

2. Um Plugins abzurufen CodeArtifact, füge deiner Datei einen `pluginManagement` Block hinzu. `settings.gradle` Der `pluginManagement` Block muss vor allen anderen Anweisungen in `settings.gradle` erscheinen.

```
pluginManagement {
    repositories {
        maven {
            name 'my_repo'
            url
 'https://my_domain-111122223333.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password "$codeartifactToken"
            }
        }
    }
}
```

3. Rufen Sie ein CodeArtifact Authentifizierungstoken ab:

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. Schreiben Sie das Authentifizierungstoken in die Datei: `gradle.properties`

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > ~/.gradle/gradle.properties
```

Token stored in separate file

Methode 2: Speichern des Authentifizierungstokens in einer separaten Datei

1. Aktualisieren Sie Ihre `build.gradle` Datei mit dem folgenden Snippet:

```
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. Um Plugins abzurufen CodeArtifact, füge deiner Datei einen `pluginManagement` Block hinzu. `settings.gradle` Der `pluginManagement` Block muss vor allen anderen Anweisungen in `settings.gradle` erscheinen.

```
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. Rufen Sie ein CodeArtifact Authentifizierungstoken ab:

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. Schreiben Sie das Authentifizierungstoken in die Datei, die in Ihrer Datei angegeben wurde: `build.gradle`

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > file
```

Token generated for each run in build.gradle

Methode 3: Generieren eines neuen Authentifizierungstokens für jeden Lauf durch Ausführung **`aws`** als Inline-Skript in **`build.gradle`**

1. Aktualisieren Sie Ihre `build.gradle` Datei mit dem folgenden Snippet:

```
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. Um Plugins abzurufen CodeArtifact, füge deiner Datei einen `pluginManagement` Block hinzu. `settings.gradle` Der `pluginManagement` Block muss vor allen anderen Anweisungen in `settings.gradle` erscheinen.

```
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
            }
        }
    }
}
```

CodeArtifact Mit MVN verwenden

Sie verwenden den `mvn` Befehl, um Maven-Builds auszuführen. In diesem Abschnitt wird gezeigt, wie Sie die Verwendung eines CodeArtifact Repositorys konfigurieren`mvn`.

Themen

- Abhängigkeiten abrufen
- Artefakte veröffentlichen
- Veröffentlichen Sie Artefakte von Drittanbietern
- Beschränken Sie das Herunterladen von Maven-Abhängigkeiten auf ein Repository CodeArtifact
- Informationen zum Apache Maven-Projekt

Abhängigkeiten abrufen

Um das Abrufen von Abhängigkeiten aus einem CodeArtifact Repository `mvn` zu konfigurieren, müssen Sie die Maven-Konfigurationsdatei und optional das POM Ihres Projekts bearbeiten. `settings.xml`

1. Falls nicht, erstellen und speichern Sie ein CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, wie unter Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen So richten Sie die Authentifizierung für Ihr Repository ein. CodeArtifact

2. Fügen Sie unter `settings.xml` (normalerweise zu finden unter`~/.m2/settings.xml`) einen `<servers>` Abschnitt mit einem Verweis auf die `CODEARTIFACT_AUTH_TOKEN` Umgebungsvariable hinzu, sodass Maven das Token in HTTP-Anfragen weitergibt.

```
<settings>
...
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

3. Fügen Sie den URL-Endpunkt für Ihr CodeArtifact Repository in einem `<repository>` Element hinzu. Sie können dies in der POM-Datei `settings.xml` oder in der POM-Datei Ihres Projekts tun.

   Sie können den Endpunkt Ihres Repositorys mit dem `get-repository-endpoint` AWS CLI Befehl abrufen.

   Bei einem Repository, das *`my_repo`* innerhalb einer Domain mit dem Namen benannt ist*`my_domain`*, lautet der Befehl beispielsweise wie folgt:

```
aws codeartifact get-repository-endpoint --domain my_domain --repository my_repo --
format maven
```

   Der `get-repository-endpoint` Befehl gibt den Repository-Endpunkt zurück:

```
url 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_repo/'
```

   > **Note**
   >
   > Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den `codeartifact.`*`region`*`.on.aws` Endpunkt.

Fügen Sie den Repository-Endpunkt `settings.xml` wie folgt hinzu.

```
<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>
```

Sie können den `<repositories>` Abschnitt auch zu einer POM-Projekt-Datei hinzufügen, um ihn nur CodeArtifact für dieses Projekt zu verwenden.

```
<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**
>
> Sie können einen beliebigen Wert im `<id>` Element verwenden, dieser muss jedoch in den `<repository>` Elementen `<server>` und identisch sein. Dadurch können die angegebenen Anmeldeinformationen in Anfragen an aufgenommen werden CodeArtifact.

Nachdem Sie diese Konfigurationsänderungen vorgenommen haben, können Sie das Projekt erstellen.

```
mvn compile
```

Maven protokolliert die vollständige URL aller Abhängigkeiten, die es auf die Konsole herunterlädt.

```
[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)
```

Artefakte veröffentlichen

Um ein Maven-Artefakt mit in einem CodeArtifact Repository `mvn` zu veröffentlichen, müssen Sie auch das Projekt `~/.m2/settings.xml` POM bearbeiten.

1. Falls nicht, erstellen und speichern Sie ein CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, wie unter Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen So richten Sie die Authentifizierung für Ihr Repository ein. CodeArtifact
2. Fügen Sie einen `<servers>` Abschnitt `settings.xml` mit einem Verweis auf die `CODEARTIFACT_AUTH_TOKEN` Umgebungsvariable hinzu, sodass Maven das Token in HTTP-Anfragen weitergibt.

```
<settings>
...
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

3. Fügen Sie Ihrem Projekt einen `<distributionManagement>` Abschnitt hinzu. `pom.xml`

```
<project>
...
     <distributionManagement>
         <repository>
             <id>codeartifact</id>
             <name>codeartifact</name>
             <url>https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_repo/</url>
         </repository>
     </distributionManagement>
...
</project>
```

Nachdem Sie diese Konfigurationsänderungen vorgenommen haben, können Sie das Projekt erstellen und im angegebenen Repository veröffentlichen.

```
mvn deploy
```

Wird verwendet`list-package-versions`, um zu überprüfen, ob das Paket erfolgreich veröffentlicht wurde.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven \
  --namespace com.company.framework --package my-package-name
```

Beispielausgabe:

```
{
    "defaultDisplayVersion": null,
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "my-package-name",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Veröffentlichen Sie Artefakte von Drittanbietern

Sie können Maven-Artefakte von Drittanbietern in einem CodeArtifact Repository mit `mvn deploy:deploy-file` veröffentlichen. Dies kann für Benutzer hilfreich sein, die Artefakte veröffentlichen möchten und nur über JAR-Dateien verfügen und keinen Zugriff auf Paketquellcode oder POM-Dateien haben.

Der `mvn deploy:deploy-file` Befehl generiert eine POM-Datei auf der Grundlage der in der Befehlszeile übergebenen Informationen.

Veröffentlichen Sie Maven-Artefakte von Drittanbietern

1. Falls nicht, erstellen und speichern Sie ein CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, wie unter Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen So richten Sie die Authentifizierung für Ihr CodeArtifact Repository ein.
2. Erstellen Sie eine `~/.m2/settings.xml` Datei mit dem folgenden Inhalt:

```
<settings>
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
</settings>
```

3. Führen Sie den Befehl `mvn deploy:deploy-file` aus:

```
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**
>
> Das obige Beispiel veröffentlicht`commons-cli 1.4`. Ändern Sie die Argumente groupId, artifactID, version und file, um eine andere JAR zu veröffentlichen.

Diese Anweisungen basieren auf Beispielen im Leitfaden zur Bereitstellung eines JARs Drittanbieter-Repositorys in der Apache Maven-Dokumentation.

Beschränken Sie das Herunterladen von Maven-Abhängigkeiten auf ein Repository CodeArtifact

Wenn ein Paket nicht aus einem konfigurierten Repository abgerufen werden kann, ruft der `mvn` Befehl es standardmäßig von Maven Central ab. Fügen Sie das `mirrors` Element hinzu, damit `settings.xml` Sie `mvn` immer Ihr Repository verwenden können. CodeArtifact

```
<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>
```

Wenn Sie ein `mirrors` Element hinzufügen, müssen Sie auch ein `pluginRepository` Element in Ihrem `settings.xml` oder habenpom.xml. Im folgenden Beispiel werden Anwendungsabhängigkeiten und Maven-Plugins aus einem CodeArtifact Repository abgerufen.

```
<settings>
...
  <profiles>
    <profile>
      <pluginRepositories>
        <pluginRepository>
          <id>codeartifact</id>
          <name>CodeArtifact Plugins</name>
          <url>https://my_domain-111122223333.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>
```

Im folgenden Beispiel werden Anwendungsabhängigkeiten aus einem CodeArtifact Repository und Maven-Plugins aus Maven Central abgerufen.

```
<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>
```

Informationen zum Apache Maven-Projekt

Weitere Informationen zu Maven finden Sie in den folgenden Themen auf der Apache Maven Project-Website:

- Einrichtung mehrerer Repositorys
- Referenz für Einstellungen
- Vertriebsmanagement
- Profile

CodeArtifact Mit deps.edn verwenden

Sie verwenden `deps.edn` with, `clj` um Abhängigkeiten für Clojure-Projekte zu verwalten. In diesem Abschnitt wird gezeigt, wie Sie die Verwendung eines `deps.edn` CodeArtifact Repositorys konfigurieren.

Themen

- Abhängigkeiten abrufen

- Artefakte veröffentlichen

Abhängigkeiten abrufen

Um das Abrufen von Abhängigkeiten aus einem CodeArtifact Repository `Clojure` zu konfigurieren, müssen Sie die Maven-Konfigurationsdatei bearbeiten,. `settings.xml`

1. Fügen Sie `<servers>` unter einen Abschnitt mit einem Verweis auf die `CODEARTIFACT_AUTH_TOKEN` Umgebungsvariable hinzu, sodass Clojure das Token in HTTP-Anfragen weitergibt. `settings.xml`

   > **Note**
   >
   > Clojure erwartet, dass sich die Datei settings.xml unter befindet. `~/.m2/settings.xml` Falls an einem anderen Ort, erstellen Sie die Datei an diesem Ort.

   ```
   <settings>
   ...
       <servers>
           <server>
               <id>codeartifact</id>
               <username>aws</username>
               <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
           </server>
       </servers>
   ...
   </settings>
   ```

2. Wenn Sie noch keine haben, generieren Sie mit Hilfe von POM-XML für Ihr Projekt`clj -Spom`.
3. Fügen Sie in Ihrer `deps.edn` Konfigurationsdatei ein Repository hinzu, das der Server-ID von Maven `settings.xml` entspricht.

   ```
   :mvn/repos {
     "clojars" nil
     "central" nil
     "codeartifact" {:url "https://my_domain-111122223333.d.codeartifact.us-
   west-2.amazonaws.com/maven/my_repo/"}
   }
   ```

> **Note**
>
> - `tools.deps`garantiert, dass die `central` und `clojars` -Repositorien zuerst auf Maven-Bibliotheken überprüft werden. Danach werden die anderen unter aufgeführten Repositorien überprüft. `deps.edn`
> - Um das direkte Herunterladen von Clojars und Maven Central zu verhindern, `clojars` muss dies auf `central` eingestellt sein. `nil`

Stellen Sie sicher, dass Sie das CodeArtifact Auth-Token in einer Umgebungsvariablen haben (siehe). Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen Wenn Sie das Paket nach diesen Änderungen erstellen, `deps.edn` werden die Abhängigkeiten von abgerufen. CodeArtifact

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den Endpunkt. `codeartifact.`*`region`*`.on.aws`

Artefakte veröffentlichen

1. Aktualisieren Sie Ihre Maven-Einstellungen und fügen Sie `deps.edn` sie CodeArtifact als Server hinzu, der von Maven erkannt wurde (siehe). Abhängigkeiten abrufen Sie können ein Tool wie deps-deploy verwenden, um Artefakte hochzuladen. CodeArtifact
2. Fügen Sie in Ihrem eine `deploy` Aufgabe hinzu`build.clj`, um die erforderlichen Artefakte in das zuvor eingerichtete Repository hochzuladen. `codeartifact`

```
(ns build
(:require [deps-deploy.deps-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. Veröffentlichen Sie das Artefakt, indem Sie den folgenden Befehl ausführen: `clj -T:build deploy`

Weitere Informationen zum Ändern von Standard-Repositorys finden Sie unter Ändern der Standard-Repositorys in der Clojure Deps and CLI Reference Rationale.

Publizieren mit curl

In diesem Abschnitt wird gezeigt, wie Sie den HTTP-Client verwenden`curl`, um Maven-Artefakte in einem CodeArtifact Repository zu veröffentlichen. Das Veröffentlichen von Artefakten mit `curl` kann nützlich sein, wenn Sie den Maven-Client nicht in Ihren Umgebungen haben oder installieren möchten.

Veröffentlichen Sie ein Maven-Artefakt mit **`curl`**

1. Rufen Sie ein CodeArtifact Autorisierungstoken ab, indem Sie die Schritte unter befolgen Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen und zu diesen Schritten zurückkehren.
2. Verwenden Sie den folgenden `curl` Befehl, um die JAR in einem CodeArtifact Repository zu veröffentlichen:

   Ersetzen Sie in jedem der `curl` Befehle in diesem Verfahren die folgenden Platzhalter:

   - Ersetzen Sie es *`my_domain`* durch Ihren CodeArtifact Domainnamen.
   - *`111122223333`*Ersetzen Sie es durch die ID des Inhabers Ihrer CodeArtifact Domain.
   - *`us-west-2`*Ersetzen Sie durch die Region, in der sich Ihre CodeArtifact Domain befindet.
   - Ersetze es *`my_repo`* durch deinen CodeArtifact Repository-Namen.

```
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**
>
> Sie müssen dem Wert des `--data-binary` Parameters ein `@` Zeichen voranstellen. Wenn Sie den Wert in Anführungszeichen setzen, `@` muss er innerhalb der Anführungszeichen stehen.

3. Verwenden Sie den folgenden `curl` Befehl, um das POM in einem CodeArtifact Repository zu veröffentlichen:

```
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. Zu diesem Zeitpunkt befindet sich das Maven-Artefakt in Ihrem CodeArtifact Repository mit dem Status. `Unfinished` Um das Paket nutzen zu können, muss es sich im `Published` Status befinden. Sie können das Paket von `Unfinished` nach verschieben, `Published` indem Sie entweder eine `maven-metadata.xml` Datei in Ihr Paket hochladen oder die UpdatePackageVersionsStatus API aufrufen, um den Status zu ändern.

   a. Option 1: Verwenden Sie den folgenden `curl` Befehl, um Ihrem Paket eine `maven-metadata.xml` Datei hinzuzufügen:

```
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
```

   Das Folgende ist ein Beispiel für den Inhalt einer `maven-metadata.xml` Datei:

```
<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: Aktualisieren Sie den Paketstatus `Published` mit der `UpdatePackageVersionsStatus` API auf.

```
aws codeartifact update-package-versions-status \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo \
    --format maven \
    --namespace com.mycompany.app \
    --package my-app \
    --versions 1.0 \
    --target-status Published
```

Wenn Sie nur über die JAR-Datei eines Artefakts verfügen, können Sie eine verbrauchbare Paketversion in einem CodeArtifact Repository veröffentlichen, indem Sie. `mvn` Dies kann nützlich sein, wenn Sie keinen Zugriff auf den Quellcode oder POM des Artefakts haben. Details dazu finden Sie unter Veröffentlichen Sie Artefakte von Drittanbietern.

Verwenden Sie Maven-Prüfsummen

Wenn ein Maven-Artefakt in einem AWS CodeArtifact Repository veröffentlicht wird, wird die Prüfsumme, die jedem Asset oder jeder Datei im Paket zugeordnet ist, zur Validierung des Uploads verwendet. Beispiele für Assets sind JAR -, POM - und WAR-Dateien. Für jedes Asset enthält das Maven-Artefakt mehrere Prüfsummendateien, die den Asset-Namen mit einer zusätzlichen Erweiterung verwenden, z. B. oder. `md5` `sha1` Beispielsweise könnten die Prüfsummendateien für eine Datei mit dem Namen `my-maven-package.jar` und lauten. `my-maven-package.jar.md5` `my-maven-package.jar.sha1`

> **Note**
>
> Maven verwendet den Begriff. `artifact` In diesem Handbuch entspricht ein Maven-Paket einem Maven-Artefakt. Weitere Informationen finden Sie unter Paket.AWS CodeArtifact

Prüfsummenspeicher

CodeArtifact speichert Maven-Prüfsummen nicht als Vermögenswerte. Das bedeutet, dass Prüfsummen nicht als einzelne Assets in der Ausgabe der API erscheinen. ListPackageVersionAssets Stattdessen CodeArtifact sind Prüfsummen, die von berechnet wurden, für jedes Asset in allen unterstützten Prüfsummentypen verfügbar. Ein Teil der Antwort auf den Aufruf der ListPackageVersionAssets Maven-Paketversion lautet beispielsweise: `commons-lang:commons-lang 2.1`


```
{
    "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
    }
}
```

Auch wenn Prüfsummen nicht als Assets gespeichert werden, können Maven-Clients Prüfsummen dennoch an den erwarteten Speicherorten veröffentlichen und herunterladen. Wenn sie beispielsweise in einem Repository aufgerufen `commons-lang:commons-lang 2.1` `maven-repo` wurde, würde der URL-Pfad für die SHA-256-Prüfsumme der JAR-Datei wie folgt lauten:

```
/maven/maven-repo/commons-lang/commons-lang/2.1/commons-lang-2.1.jar.sha256
```

Wenn Sie bestehende Maven-Pakete (z. B. Pakete, die zuvor in Amazon S3 gespeichert wurden) auf CodeArtifact einen generischen HTTP-Client wie hochladen, ist es nicht erforderlich`curl`, die Prüfsummen hochzuladen. CodeArtifact generiert sie automatisch. Wenn Sie überprüfen möchten, ob die Assets korrekt hochgeladen wurden, können Sie den ListPackageVersionAssets API-Vorgang verwenden, um die Prüfsummen in der Antwort mit den ursprünglichen Prüfsummenwerten für jedes Asset zu vergleichen.

Bei der Veröffentlichung stimmen die Prüfsummen nicht überein

Neben Assets und Prüfsummen enthalten Maven-Artefakte auch eine Datei. `maven-metadata.xml` Die normale Veröffentlichungsreihenfolge für ein Maven-Paket besteht darin, dass alle Assets und Prüfsummen zuerst hochgeladen werden, gefolgt von. `maven-metadata.xml` Die zuvor `commons-lang 2.1` beschriebene Veröffentlichungsreihenfolge für die Maven-Paketversion, vorausgesetzt, der Client wäre für die Veröffentlichung von SHA-256-Prüfsummendateien konfiguriert, wäre beispielsweise:

```
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
```

Beim Hochladen der Prüfsummendatei für ein Asset, z. B. eine JAR-Datei, schlägt die Prüfsummen-Upload-Anfrage mit einer Antwort von 400 (Bad Request) fehl, wenn zwischen dem hochgeladenen

Prüfsummenwert und dem von berechneten Prüfsummenwert eine Diskrepanz besteht. CodeArtifact Wenn das entsprechende Asset nicht existiert, schlägt die Anfrage mit einer 404-Antwort (Not Found) fehl. Um diesen Fehler zu vermeiden, müssen Sie zuerst das Asset und dann die Prüfsumme hochladen.

Wenn hochgeladen `maven-metadata.xml` wird, ändert sich CodeArtifact normalerweise der Status der Maven-Paketversion von `Unfinished` zu. `Published` Wenn bei einem Asset eine nicht übereinstimmende Prüfsumme festgestellt wird, CodeArtifact wird als Antwort auf die Veröffentlichungsanfrage eine 400 (Bad Request) zurückgegeben. `maven-metadata.xml` Dieser Fehler kann dazu führen, dass der Client das Hochladen von Dateien für diese Paketversion beendet. Wenn dies der Fall ist und die `maven-metadata.xml` Datei nicht hochgeladen wird, können keine Inhalte der bereits hochgeladenen Paketversion heruntergeladen werden. Dies liegt daran, dass der Status der Paketversion nicht auf festgelegt ist `Published` und weiterhin gültig ist`Unfinished`.

CodeArtifact ermöglicht das Hinzufügen weiterer Assets zu einer Maven-Paketversion, auch nachdem `maven-metadata.xml` sie hochgeladen wurden und der Status der Paketversion auf `Published` gesetzt wurde. In diesem Status schlägt eine Anfrage zum Hochladen einer nicht übereinstimmenden Prüfsummendatei ebenfalls fehl und es wird eine 400-Antwort (Bad Request) angezeigt. Da der Paketversionsstatus jedoch bereits auf gesetzt wurde`Published`, können Sie jedes Asset aus dem Paket herunterladen, auch solche, für die der Upload der Prüfsummendatei fehlgeschlagen ist. Wenn Sie eine Prüfsumme für ein Asset herunterladen, bei dem der Upload der Prüfsummendatei fehlgeschlagen ist, ist der Prüfsummenwert, den der Client erhält, der Prüfsummenwert, der auf der CodeArtifact Grundlage der hochgeladenen Asset-Daten berechnet wird.

CodeArtifact Bei Prüfsummenvergleichen wird zwischen Groß- und Kleinschreibung unterschieden, und die von berechneten Prüfsummen werden in Kleinbuchstaben formatiert. CodeArtifact Wenn die Prüfsumme hochgeladen `909FA780F76DA393E992A3D2D495F468` wird, schlägt sie daher fehl und die Prüfsumme stimmt nicht überein, da sie CodeArtifact nicht als gleich behandelt wird. `909fa780f76da393e992a3d2d495f468`

Wiederherstellung nach nicht übereinstimmenden Prüfsummen

Wenn ein Prüfsummen-Upload aufgrund einer nicht übereinstimmenden Prüfsumme fehlschlägt, versuchen Sie es mit einem der folgenden Verfahren:

- Führen Sie den Befehl, der das Maven-Artefakt veröffentlicht, erneut aus. Dies könnte funktionieren, wenn ein Netzwerkproblem die Prüfsummendatei beschädigt hat. Wenn das

Netzwerkproblem dadurch behoben wird, stimmt die Prüfsumme überein und der Download ist erfolgreich.

- Löschen Sie die Paketversion und veröffentlichen Sie sie erneut. Weitere Informationen finden Sie DeletePackageVersionsin der CodeArtifact AWS-API-Referenz.

Verwenden Sie Maven-Snapshots

Ein Maven-Snapshot ist eine spezielle Version eines Maven-Pakets, die sich auf den neuesten Produktions-Branch-Code bezieht. Es ist eine Entwicklungsversion, die der endgültigen Release-Version vorausgeht. Sie können eine Snapshot-Version eines Maven-Pakets anhand des Suffix identifizieren, das an `SNAPSHOT` die Paketversion angehängt wird. Der Snapshot der Version lautet beispielsweise. `1.1` `1.1-SNAPSHOT` Weitere Informationen finden Sie unter Was ist eine SNAPSHOT-Version? auf der Website des Apache Maven Project.

AWS CodeArtifact unterstützt das Veröffentlichen und Verwenden von Maven-Snapshots. Eindeutige Snapshots, die eine zeitbasierte Versionsnummer verwenden, sind die einzigen Snapshots, die unterstützt werden. CodeArtifact unterstützt keine uneindeutigen Snapshots, die von Maven 2-Clients generiert werden. Sie können einen unterstützten Maven-Snapshot in einem beliebigen Repository veröffentlichen. CodeArtifact

Themen

- Veröffentlichung von Snapshots in CodeArtifact
- Snapshot-Versionen werden konsumiert
- Löschen von Snapshot-Versionen
- Veröffentlichung von Snapshots mit Curl
- Schnappschüsse und externe Verbindungen
- Snapshots und Upstream-Repositorys

Veröffentlichung von Snapshots in CodeArtifact

AWS CodeArtifact unterstützt die Anforderungsmuster, die Clients `mvn` beispielsweise beim Veröffentlichen von Snapshots verwenden. Aus diesem Grund können Sie der Dokumentation für Ihr Build-Tool oder Ihren Paketmanager folgen, ohne genau zu wissen, wie Maven-Snapshots veröffentlicht werden. Wenn Sie etwas komplexeres tun, wird in diesem Abschnitt detailliert beschrieben, wie mit Snapshots CodeArtifact umgegangen wird.

Wenn ein Maven-Snapshot in einem CodeArtifact Repository veröffentlicht wird, wird seine vorherige Version in einer neuen Version, einem sogenannten Build, aufbewahrt. Jedes Mal, wenn ein Maven-Snapshot veröffentlicht wird, wird eine neue Build-Version erstellt. Alle vorherigen Versionen eines Snapshots werden in seinen Build-Versionen beibehalten. Wenn ein Maven-Snapshot veröffentlicht wird, wird sein Paketversionsstatus auf gesetzt `Published` und der Status des Builds, der die vorherige Version enthält, wird auf `Unlisted` gesetzt. Dieses Verhalten gilt nur für Maven-Paketversionen, bei denen die Paketversion ein `-SNAPSHOT` Suffix hat.

Beispielsweise werden Snapshot-Versionen eines Maven-Pakets namens in ein `com.mycompany.myapp:pkg-1` CodeArtifact Repository namens hochgeladen. `my-maven-repo` Die Snapshot-Version ist`1.0-SNAPSHOT`. Bisher wurden keine Versionen von `com.mycompany.myapp:pkg-1` veröffentlicht. Zunächst werden die Assets des ersten Builds unter den folgenden Pfaden veröffentlicht:

```
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
```

Beachten Sie, dass der Zeitstempel vom Client generiert `20210728.194552-1` wird, der die Snapshot-Builds veröffentlicht.

Nachdem die .pom- und .jar-Dateien hochgeladen wurden, ist die einzige Version davon, `com.mycompany.myapp:pkg-1` die im Repository vorhanden ist. `1.0-20210728.194552-1` Dies geschieht, obwohl es sich bei der im vorherigen Pfad angegebenen Version handelt. `1.0-SNAPSHOT` Der Status der Paketversion ist zu diesem Zeitpunkt`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"
}
```

Als Nächstes lädt der Client die `maven-metadata.xml` Datei für die Paketversion hoch:

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

Wenn die Datei maven-metadata.xml erfolgreich hochgeladen wurde, wird die `1.0-SNAPSHOT` Paketversion CodeArtifact erstellt und die `1.0-20210728.194552-1` Version auf `Unlisted` gesetzt.

```
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": "Published"
        }
    ],
    "defaultDisplayVersion": "1.0-SNAPSHOT",
    "format": "maven",
    "package": "pkg-1",
    "namespace": "com.mycompany.myapp"
}
```

Zu diesem Zeitpunkt `1.0-SNAPSHOT` kann die Snapshot-Version in einem Build verwendet werden. Es gibt zwar zwei Versionen von `com.mycompany.myapp:pkg-1` im Repository`my-maven-repo`, aber beide enthalten dieselben Ressourcen.

```
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"
```

```
]
```

Die Ausführung desselben `list-package-version-assets` Befehls wie zuvor gezeigt mit `--package-version` geändertem Parameter `1.0-20210728.194552-1` führt zu einer identischen Ausgabe.

Wenn dem Repository weitere Builds von hinzugefügt `1.0-SNAPSHOT` werden, wird für jeden neuen Build eine neue `Unlisted` Paketversion erstellt. Die Ressourcen der Version `1.0-SNAPSHOT` werden jedes Mal aktualisiert, sodass sich die Version immer auf den neuesten Build für diese Version bezieht. Die Aktualisierung von `1.0-SNAPSHOT` mit den neuesten Assets wird durch das Hochladen der `maven-metadata.xml` Datei für den neuen Build initiiert.

Snapshot-Versionen werden konsumiert

Wenn Sie einen Snapshot anfordern, `Published` wird die Version mit dem Status zurückgegeben. Dies ist immer die neueste Version des Maven-Snapshots. Sie können auch einen bestimmten Build eines Snapshots anfordern, indem Sie die Build-Versionsnummer (z. B.`1.0-20210728.194552-1`) anstelle der Snapshot-Version (z. B.`1.0-SNAPSHOT`) im URL-Pfad verwenden. Um die Build-Versionen eines Maven-Snapshots zu sehen, verwenden Sie die ListPackageVersions API im CodeArtifact API-Leitfaden und setzen Sie den Status-Parameter auf`Unlisted`.

Löschen von Snapshot-Versionen

Um alle Build-Versionen eines Maven-Snapshots zu löschen, verwenden Sie die DeletePackageVersionsAPI und geben Sie die Versionen an, die Sie löschen möchten.

Veröffentlichung von Snapshots mit Curl

Wenn Sie bestehende Snapshot-Versionen in Amazon Simple Storage Service (Amazon S3) oder einem anderen Artefakt-Repository-Produkt gespeichert haben, möchten Sie diese möglicherweise erneut veröffentlichen. AWS CodeArtifact Aufgrund der CodeArtifact Unterstützung von Maven-Snapshots (siehe Veröffentlichung von Snapshots in CodeArtifact) `curl` ist das Veröffentlichen von Snapshots mit einem generischen HTTP-Client komplexer als das Veröffentlichen von Maven-Release-Versionen, wie unter beschrieben. Publizieren mit curl Beachten Sie, dass dieser Abschnitt nicht relevant ist, wenn Sie Snapshot-Versionen mit einem Maven-Client wie oder erstellen und bereitstellen. `mvn` `gradle` Sie müssen der Dokumentation für diesen Client folgen.

Das Veröffentlichen einer Snapshot-Version beinhaltet das Veröffentlichen eines oder mehrerer Builds einer Snapshot-Version. Wenn es n Builds einer Snapshot-Version gibt, gibt es n + 1

CodeArtifact Versionen: n Build-Versionen, alle mit dem Status`Unlisted`, und eine Snapshot-Version (der letzte veröffentlichte Build) mit dem Status`Published`. CodeArtifact Die Snapshot-Version (d. h. die Version mit einer Versionszeichenfolge, die „-SNAPSHOT“ enthält) enthält einen identischen Satz von Assets wie der zuletzt veröffentlichte Build. Die einfachste Methode, diese Struktur zu erstellen, `curl` ist wie folgt:

1. Veröffentlichen Sie alle Assets aller Builds mit`curl`.
2. Veröffentlichen Sie die `maven-metadata.xml` Datei des letzten Builds (d. h. den Build mit dem aktuellsten Datums- und Zeitstempel) mit. `curl` Dadurch wird eine Version mit „`-SNAPSHOT`“ in der Versionszeichenfolge und mit den richtigen Elementen erstellt.
3. Verwenden Sie die UpdatePackageVersionsStatusAPI, um den Status aller nicht neuesten Build-Versionen auf `Unlisted` festzulegen.

Verwenden Sie die folgenden `curl` Befehle, um Snapshot-Assets (wie .jar- und .pom-Dateien) für die Snapshot-Version `1.0-SNAPSHOT` eines Pakets zu veröffentlichen: `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
```

Wenn Sie diese Beispiele verwenden:

- Ersetze es *`my_domain`* durch deinen CodeArtifact Domainnamen.
- *`111122223333`*Ersetzen Sie es durch die AWS-Konto ID des Inhabers Ihrer CodeArtifact Domain.
- *`us-west-2`*Ersetzen Sie durch die, AWS-Region in der sich Ihre CodeArtifact Domain befindet.
- Ersetze es *`my_maven_repo`* durch deinen CodeArtifact Repository-Namen.

> **Important**
>
> Sie müssen dem Wert des `--data-binary` Parameters das `@` Zeichen voranstellen. Wenn Sie den Wert in Anführungszeichen setzen, `@` muss er innerhalb der Anführungszeichen stehen.

Möglicherweise müssen Sie für jeden Build mehr als zwei Assets hochladen. Beispielsweise könnten neben den Haupt-JAR- und Javadoc-Dateien auch Javadoc- und JAR-Quelldateien vorhanden sein. `pom.xml` Es ist nicht notwendig, Prüfsummendateien für die Paketversions-Assets zu veröffentlichen, da CodeArtifact automatisch Prüfsummen für jedes hochgeladene Asset generiert werden. Um zu überprüfen, ob die Assets korrekt hochgeladen wurden, rufen Sie die generierten Prüfsummen mit dem `list-package-version-assets` Befehl ab und vergleichen Sie sie mit den ursprünglichen Prüfsummen. Weitere Informationen zum CodeArtifact Umgang mit Maven-Prüfsummen finden Sie unter. Verwenden Sie Maven-Prüfsummen

Verwenden Sie den folgenden curl-Befehl, um die `maven-metadata.xml` Datei für die neueste Build-Version zu veröffentlichen:

```
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
```

Die `maven-metadata.xml` Datei muss auf mindestens eines der Assets in der neuesten Build-Version im `<snapshotVersions>` Element verweisen. Darüber hinaus muss der `<timestamp>` Wert vorhanden sein und mit dem Zeitstempel in den Asset-Dateinamen übereinstimmen. Für den zuvor veröffentlichten `20210729.171330-2` Build `maven-metadata.xml` wäre der Inhalt von beispielsweise:

```
<?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>
```

Nach der Veröffentlichung `maven-metadata.xml` besteht der letzte Schritt darin, allen anderen Build-Versionen (d. h. allen Build-Versionen außer dem letzten Build) den Paketversionsstatus von zuzuweisen`Unlisted`. Wenn die `1.0-SNAPSHOT` Version beispielsweise zwei Builds hat, wobei der erste Build ist, lautet der Befehl`20210728.194552-1`, auf den dieser Build gesetzt werden `Unlisted` soll:

```
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
```

Schnappschüsse und externe Verbindungen

Maven-Snapshots können nicht über eine externe Verbindung aus einem öffentlichen Maven-Repository abgerufen werden. AWS CodeArtifact unterstützt nur den Import von Maven-Release-Versionen.

Snapshots und Upstream-Repositorys

Im Allgemeinen funktionieren Maven-Snapshots genauso wie Release-Versionen von Maven, wenn sie mit Upstream-Repositorys verwendet werden. Es gibt jedoch eine Einschränkung, wenn Sie planen, Snapshots derselben Paketversion in zwei Repositorys zu veröffentlichen, die eine Upstream-

Beziehung haben. Nehmen wir zum Beispiel an, es gibt zwei Repositorys in einer AWS CodeArtifact Domain `R` und `U` wo ist ein Upstream von. `U` `R` Wenn Sie einen neuen Build in `R` veröffentlichen und ein Maven-Client den neuesten Build dieser Snapshot-Version anfordert, wird die neueste Version von CodeArtifact zurückgegeben. `U` Dies kann unerwartet sein, da die neueste Version jetzt verfügbar ist`R`, nicht`U`. Es gibt zwei Möglichkeiten, dies zu vermeiden:

1. Veröffentlichen Sie keine Builds einer Snapshot-Version wie `1.0-SNAPSHOT` in`R`, falls `1.0-SNAPSHOT` vorhanden in`U`.
2. Verwenden Sie die Einstellungen zur CodeArtifact Paketherkunft, um Upstreams für dieses Paket in `R` zu deaktivieren. Letzteres ermöglicht es Ihnen, Builds von `1.0-SNAPSHOT` in zu veröffentlichen`R`, `R` verhindert aber auch, dass andere Versionen dieses Pakets abgerufen werden, `U` die nicht bereits gespeichert sind.

Maven-Pakete von Upstreams und externen Verbindungen anfordern

Import von Standard-Asset-Namen

Beim Import einer Maven-Paketversion aus einem öffentlichen Repository wie Maven Central CodeArtifact versucht AWS, alle Assets in dieser Paketversion zu importieren. Wie unter beschriebenEine Paketversion mit Upstream-Repositorys anfordern, erfolgt der Import in den folgenden Fällen:

- Ein Client fordert ein Maven-Asset aus einem CodeArtifact Repository an.
- Die Paketversion ist noch nicht im Repository oder seinen Upstreams vorhanden.
- Es gibt eine erreichbare externe Verbindung zu einem öffentlichen Maven-Repository.

Auch wenn der Client möglicherweise nur ein Asset angefordert hat, CodeArtifact versucht er, alle Assets zu importieren, die er für diese Paketversion finden kann. Wie CodeArtifact festgestellt wird, welche Ressourcen für eine Maven-Paketversion verfügbar sind, hängt vom jeweiligen öffentlichen Repository ab. Einige öffentliche Maven-Repositorien unterstützen das Anfordern einer Liste von Assets, andere jedoch nicht. CodeArtifact Generiert für Repositorys, die keine Möglichkeit bieten, Assets aufzulisten, eine Reihe von Asset-Namen, die wahrscheinlich existieren. Wenn beispielsweise ein Asset der Maven-Paketversion `junit 4.13.2` angefordert CodeArtifact wird, wird versucht, die folgenden Assets zu importieren:

- `junit-4.13.2.pom`
- `junit-4.13.2.jar`
- `junit-4.13.2-javadoc.jar`
- `junit-4.13.2-sources.jar`

Importieren von nicht standardmäßigen Asset-Namen

Wenn ein Maven-Client ein Asset anfordert, das keinem der oben beschriebenen Muster entspricht, wird CodeArtifact geprüft, ob dieses Asset im öffentlichen Repository vorhanden ist. Wenn das Asset vorhanden ist, wird es importiert und dem vorhandenen Paketversionsdatensatz hinzugefügt, falls einer existiert. Die Maven-Paketversion `com.android.tools.build:aapt2 7.3.1-8691043` enthält beispielsweise die folgenden Ressourcen:

- `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`

Wenn ein Client die POM-Datei anfordert und die Ressourcen der Paketversion nicht auflisten kann, ist das POM das einzige importierte Asset. CodeArtifact Dies liegt daran, dass keines der anderen Assets den Standardmustern für Asset-Namen entspricht. Wenn der Client jedoch eines der JAR-Assets anfordert, wird dieses Asset importiert und der vorhandenen Paketversion hinzugefügt, in der gespeichert ist CodeArtifact. Die Paketversionen sowohl im am weitesten nachgelagerten Repository (dem Repository, für das der Client die Anfrage gestellt hat) als auch im Repository, an das die externe Verbindung angeschlossen ist, werden aktualisiert, sodass sie das neue Asset enthalten, wie unter beschrieben. Aufbewahrung von Paketen aus Upstream-Repositorys

Sobald eine Paketversion in einem CodeArtifact Repository gespeichert ist, ist sie normalerweise nicht von Änderungen in den Upstream-Repositorys betroffen. Weitere Informationen finden Sie unter Aufbewahrung von Paketen aus Upstream-Repositorys. Das zuvor beschriebene Verhalten von Maven-Assets mit nicht standardmäßigen Namen stellt jedoch eine Ausnahme von dieser Regel dar. Die Downstream-Paketversion ändert sich zwar nicht, ohne dass ein zusätzlicher Inhalt von einem Client angefordert wird, aber in diesem Fall wird die beibehaltene Paketversion geändert, nachdem sie ursprünglich beibehalten wurde, und ist daher nicht unveränderlich. Dieses Verhalten ist notwendig, da Maven-Assets mit nicht standardmäßigen Namen andernfalls nicht zugänglich

wären. CodeArtifact Dieses Verhalten ist auch möglich, wenn sie zu einer Maven-Paketversion in einem öffentlichen Repository hinzugefügt werden, nachdem die Paketversion in einem Repository aufbewahrt wurde. CodeArtifact

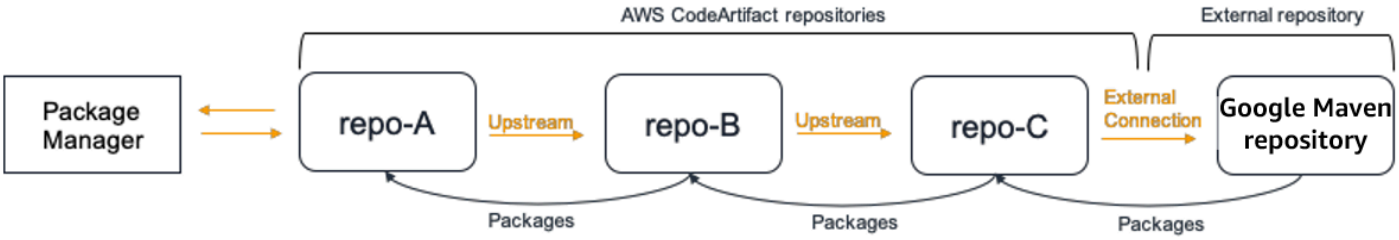
Die Herkunft von Assets wird überprüft

Wenn Sie einer zuvor beibehaltenen Maven-Paketversion ein neues Asset hinzufügen, wird CodeArtifact bestätigt, dass der Ursprung der beibehaltenen Paketversion mit dem Ursprung des neuen Assets identisch ist. Dadurch wird verhindert, dass eine „gemischte“ Paketversion erstellt wird, bei der verschiedene Assets aus unterschiedlichen öffentlichen Repositorys stammen. Ohne diese Prüfung könnte es zu einer Vermischung von Assets kommen, wenn eine Maven-Paketversion in mehr als einem öffentlichen Repository veröffentlicht wird und diese Repositorys Teil des Upstream-Graphen eines CodeArtifact Repositorys sind.

Import neuer Assets und Paketversionsstatus in Upstream-Repositorys

Der Paketversionsstatus von Paketversionen in Upstream-Repositorys kann CodeArtifact verhindern, dass diese Versionen in Downstream-Repositorys beibehalten werden.

Nehmen wir zum Beispiel an, eine Domain hat drei Repositorys:`repo-A`,, und `repo-B``repo-C`, wobei `repo-B` sich ein Upstream von `repo-A` und `repo-C` ein Upstream von befinden. `repo-B`



Die Paketversion `7.3.1` des Maven-Pakets `com.android.tools.build:aapt2` ist in vorhanden `repo-B` und hat den Status. `Published` Es ist nicht vorhanden in`repo-A`. Wenn ein Client ein Asset dieser Paketversion anfordert`repo-A`, lautet die Antwort 200 (OK) und die Maven-Paketversion `7.3.1` wird beibehalten. `repo-A` Wenn der Status der Paketversion `7.3.1` in jedoch `Archived` oder `repo-B` lautet`Disposed`, lautet die Antwort 404 (Not Found), da die Ressourcen der Paketversionen in diesen beiden Status nicht heruntergeladen werden können.

Beachten Sie, dass das Setzen der Paketquellkontrolle auf `upstream=BLOCK` for `com.android.tools.build:aapt2` in `repo-A``repo-B`, und `repo-C` verhindert, dass neue Inhalte für alle Versionen dieses Pakets abgerufen werden`repo-A`, unabhängig vom Status der Paketversion.

Fehlerbehebung in Maven

Die folgenden Informationen können Ihnen bei der Behebung häufiger Probleme mit Maven und helfen. CodeArtifact

Deaktivieren Sie parallel Eingaben, um Fehler 429 zu beheben: Zu viele Anfragen

Ab Version 3.9.0 lädt Maven Paketartefakte parallel hoch (bis zu 5 Dateien gleichzeitig). Dies kann CodeArtifact dazu führen, dass gelegentlich mit dem Fehlerantwortcode 429 (Zu viele Anfragen) geantwortet wird. Wenn dieser Fehler auftritt, können Sie parallel Eingaben deaktivieren, um ihn zu beheben.

Um parallel Puts zu deaktivieren, setzen Sie die `aether.connector.basic.parallelPut` Eigenschaft `false` in Ihrem Profil in Ihrer `settings.xml` Datei auf, wie im folgenden Beispiel gezeigt:

```
<settings>
    <profiles>
        <profile>
            <id>default</id>
            <properties>
                <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
            </properties>
        </profile>
    </profiles>
<settings>
```

Weitere Informationen finden Sie unter Artifact Resolver Configuration Options in der Maven-Dokumentation.

CodeArtifact Mit npm verwenden

In diesen Themen wird beschrieben, wie Sie npm, den Paketmanager Node.js, mit verwenden. CodeArtifact

> **Note**
>
> CodeArtifact unterstützt `node v4.9.1` und später `npm v5.0.0` und später.

Themen

- Konfigurieren und verwenden Sie npm mit CodeArtifact
- Konfiguriere und verwende Yarn mit CodeArtifact
- Unterstützung für npm-Befehle
- Handhabung von NPM-Tags
- Support für npm-kompatible Paketmanager

Konfigurieren und verwenden Sie npm mit CodeArtifact

Nachdem Sie ein Repository erstellt haben CodeArtifact, können Sie den npm-Client verwenden, um Pakete zu installieren und zu veröffentlichen. Die empfohlene Methode zur Konfiguration von npm mit Ihrem Repository-Endpunkt und Autorisierungstoken ist die Verwendung des `aws codeartifact login` Befehls. Sie können npm auch manuell konfigurieren.

Inhalt

- Konfiguration von npm mit dem Login-Befehl
- Konfiguration von npm ohne Verwendung des Login-Befehls
- NPM-Befehle ausführen
- Überprüfung der NPM-Authentifizierung und -Autorisierung
- Zurück zur Standard-NPM-Registrierung
- Fehlerbehebung bei langsamen Installationen mit npm 8.x oder höher

Konfiguration von npm mit dem Login-Befehl

Verwenden Sie den `aws codeartifact login` Befehl, um Anmeldeinformationen für die Verwendung mit npm abzurufen.

> **Note**
>
> Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie dies nicht angeben. `--domain-owner` Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

> **Important**
>
> Wenn Sie npm 10.x oder neuer verwenden, müssen Sie AWS CLI Version 2.9.5 oder neuer verwenden, um den Befehl erfolgreich auszuführen. `aws codeartifact login`

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

Dieser Befehl nimmt die folgenden Änderungen an Ihrer ~/.npmrc-Datei vor:

- Fügt ein Autorisierungstoken hinzu, nachdem Sie es mithilfe Ihrer Anmeldeinformationen abgerufen haben. CodeArtifact AWS
- Setzt die NPM-Registrierung auf das in der Option angegebene Repository. `--repository`
- Für npm 6 und niedriger: Fügt hinzu, `"always-auth=true"` dass das Autorisierungstoken für jeden npm-Befehl gesendet wird.

Die Standardautorisierungszeit nach dem Aufrufen `login` beträgt 12 Stunden und `login` muss aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

Konfiguration von npm ohne Verwendung des Login-Befehls

Sie können npm mit Ihrem CodeArtifact Repository ohne den `aws codeartifact login` Befehl konfigurieren, indem Sie die npm-Konfiguration manuell aktualisieren.

Um npm zu konfigurieren, ohne den Login-Befehl zu verwenden

1. Rufen Sie in einer Befehlszeile ein CodeArtifact Autorisierungstoken ab und speichern Sie es in einer Umgebungsvariable. npm verwendet dieses Token, um sich bei Ihrem Repository zu authentifizieren. CodeArtifact

   > **Note**
   >
   > Der folgende Befehl gilt für macOS- oder Linux-Maschinen. Informationen zur Konfiguration von Umgebungsvariablen auf einem Windows-Computer finden Sie unterÜbergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen.

   ```
   CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

2. Rufen Sie den Endpunkt Ihres CodeArtifact Repositorys ab, indem Sie den folgenden Befehl ausführen. Ihr Repository-Endpunkt wird verwendet, um npm auf Ihr Repository zu verweisen, um Pakete zu installieren oder zu veröffentlichen.

   - Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
   - *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
   - Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format npm
   ```

   Die folgende URL ist ein Beispiel für einen Repository-Endpunkt.

   ```
   https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/
   ```

> **Important**
>
> Die Registrierungs-URL muss mit einem Schrägstrich (/) enden. Andernfalls können Sie keine Verbindung zum Repository herstellen.

3. Verwenden Sie den `npm config set` Befehl, um die Registrierung auf Ihr CodeArtifact Repository festzulegen. Ersetzen Sie die URL durch die Repository-Endpunkt-URL aus dem vorherigen Schritt.

```
npm config set
 registry=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/
```

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den `codeartifact.region.on.aws` Endpunkt.

4. Verwenden Sie den `npm config set` Befehl, um Ihr Autorisierungstoken zu Ihrer NPM-Konfiguration hinzuzufügen.

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:_authToken=$CODEARTIFACT_AUTH_TOKEN
```

Für npm 6 oder niedriger: Damit npm das Authentifizierungstoken immer weitergibt CodeArtifact, auch bei `GET` Anfragen, setzen Sie die `always-auth` Konfigurationsvariable mit. `npm config set`

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:always-auth=true
```

Beispiel für eine npm-Konfigurationsdatei () **`.npmrc`**

Im Folgenden finden Sie eine `.npmrc` Beispieldatei, nachdem Sie die vorherigen Anweisungen befolgt haben, um den CodeArtifact Registrierungsendpunkt festzulegen, ein Authentifizierungstoken hinzuzufügen und zu konfigurieren`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
```

NPM-Befehle ausführen

Nachdem Sie den npm-Client konfiguriert haben, können Sie npm-Befehle ausführen. Unter der Annahme, dass ein Paket in Ihrem Repository oder einem seiner Upstream-Repositorys vorhanden ist, können Sie es mit installieren. `npm install` Verwenden Sie zum Beispiel Folgendes, um das `lodash` Paket zu installieren.

```
npm install lodash
```

Verwenden Sie den folgenden Befehl, um ein neues npm-Paket in einem CodeArtifact Repository zu veröffentlichen.

```
npm publish
```

Informationen zum Erstellen von npm-Paketen finden Sie unter Creating Node.js Modules auf der NPM-Dokumentationswebsite. Eine Liste der npm-Befehle, die von unterstützt werden CodeArtifact, finden Sie unter npm-Befehlsunterstützung.

Überprüfung der NPM-Authentifizierung und -Autorisierung

Durch das Aufrufen des `npm ping` Befehls können Sie Folgendes überprüfen:

- Sie haben Ihre Anmeldeinformationen korrekt konfiguriert, sodass Sie sich bei einem Repository CodeArtifact authentifizieren können.
- Die Autorisierungskonfiguration gewährt Ihnen die `ReadFromRepository` Erlaubnis.

Die Ausgabe eines erfolgreichen Aufrufs von `npm ping` sieht wie folgt aus.

```
$ 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
```

Die `-d` Option veranlasst npm, zusätzliche Debug-Informationen auszudrucken, einschließlich der Repository-URL. Anhand dieser Informationen können Sie leicht bestätigen, dass npm so konfiguriert ist, dass es das von Ihnen erwartete Repository verwendet.

Zurück zur Standard-NPM-Registrierung

Durch die Konfiguration von npm mit CodeArtifact wird die npm-Registrierung auf das angegebene Repository festgelegt. CodeArtifact Sie können den folgenden Befehl ausführen, um die npm-Registrierung auf ihre Standardregistrierung zurückzusetzen, wenn Sie mit der Verbindung fertig sind. CodeArtifact

```
npm config set registry https://registry.npmjs.com/
```

Fehlerbehebung bei langsamen Installationen mit npm 8.x oder höher

In den NPM-Versionen 8.x und höher gibt es ein bekanntes Problem: Wenn eine Anfrage an ein Paket-Repository gestellt wird und das Repository den Client zu Amazon S3 umleitet, anstatt die Assets direkt zu streamen, kann der npm-Client pro Abhängigkeit mehrere Minuten lang hängen bleiben.

Da CodeArtifact Repositorys so konzipiert sind, dass sie die Anfrage immer an Amazon S3 weiterleiten, tritt dieses Problem manchmal auf, das aufgrund der langen NPM-Installationszeiten zu langen Build-Zeiten führt. Fälle dieses Verhaltens werden als Fortschrittsbalken angezeigt, der mehrere Minuten lang angezeigt wird.

Um dieses Problem zu vermeiden, verwenden Sie entweder die `progress=false` Flags `--no-progress` oder mit `npm` CLI-Befehlen, wie im folgenden Beispiel gezeigt.

```
npm install lodash --no-progress
```

Konfiguriere und verwende Yarn mit CodeArtifact

Nachdem Sie ein Repository erstellt haben, können Sie den Yarn-Client verwenden, um npm-Pakete zu verwalten.

> **Note**
>
> `Yarn 1.X`liest und verwendet Informationen aus Ihrer npm-Konfigurationsdatei (.npmrc), tut dies jedoch nicht. `Yarn 2.X` Die Konfiguration für `Yarn 2.X` muss in der Datei .yarnrc.yml definiert werden.

Inhalt

- Konfigurieren Sie Yarn 1.X mit dem Befehl aws codeartifact login
- Konfiguriere Yarn 2.X mit dem Befehl yarn config set

Konfigurieren Sie Yarn 1.X mit dem Befehl **`aws codeartifact login`**

Denn `Yarn 1.X` Sie können Yarn mit dem CodeArtifact `aws codeartifact login` Befehl konfigurieren. Der `login` Befehl konfiguriert Ihre ~/.npmrc-Datei mit Ihren CodeArtifact Repository-Endpunktinformationen und Anmeldeinformationen. Mit verwenden `yarn` Befehle `Yarn 1.X` die Konfigurationsinformationen aus der Datei ~/.npmrc.

Zur Konfiguration mit dem Login-Befehl **`Yarn 1.X`**

1. Falls Sie dies noch nicht getan haben, konfigurieren Sie Ihre AWS Anmeldeinformationen für die Verwendung mit AWS CLI, wie unter beschriebenErste Schritte mit CodeArtifact.
2. Um den `aws codeartifact login` Befehl erfolgreich auszuführen, muss npm installiert sein. Installationsanweisungen finden Sie in der npm-Dokumentation unter Node.js und npm herunterladen und installieren.
3. Verwenden Sie den `aws codeartifact login` Befehl, um CodeArtifact Anmeldeinformationen abzurufen und Ihre ~/.npmrc-Datei zu konfigurieren.

   - Ersetze es durch deinen Domainnamen. *`my_domain`* CodeArtifact
   - *`111122223333`*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn du auf ein Repository in einer Domain zugreifst, die dir gehört, musst du nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

- Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

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

Der `login` Befehl nimmt die folgenden Änderungen an Ihrer ~/.npmrc-Datei vor:

- Fügt ein Autorisierungstoken hinzu, nachdem Sie es mithilfe Ihrer Anmeldeinformationen abgerufen haben. CodeArtifact AWS
- Setzt die NPM-Registrierung auf das in der Option angegebene Repository. `--repository`
- Für npm 6 und niedriger: Fügt hinzu, `"always-auth=true"` dass das Autorisierungstoken für jeden npm-Befehl gesendet wird.

Die Standardautorisierungszeit nach dem Aufrufen `login` beträgt 12 Stunden und `login` muss aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

4. Für npm 7.X und 8.X müssen Sie `always-auth=true` zu Ihrer ~/.npmrc-Datei etwas hinzufügen, um Yarn verwenden zu können.

   - Öffne deine ~/.npmrc-Datei in einem Texteditor und füge eine neue Zeile hinzu. `always-auth=true`

Sie können den `yarn config list` Befehl verwenden, um zu überprüfen, ob Yarn die richtige Konfiguration verwendet. Überprüfen Sie nach dem Ausführen des Befehls die Werte im `info npm config` Abschnitt. Der Inhalt sollte dem folgenden Snippet ähneln.

```
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
}
```

Konfiguriere Yarn 2.X mit dem Befehl **`yarn config set`**

Das folgende Verfahren beschreibt, wie Sie konfigurieren, `Yarn 2.X` indem Sie Ihre `.yarnrc.yml` Konfiguration über die Befehlszeile mit dem `yarn config set` Befehl aktualisieren.

Um die **`yarnrc.yml`** Konfiguration von der Befehlszeile aus zu aktualisieren

1. Falls Sie dies noch nicht getan haben, konfigurieren Sie Ihre AWS Anmeldeinformationen für die AWS CLI, wie unter beschriebenErste Schritte mit CodeArtifact.
2. Verwenden Sie den `aws codeartifact get-repository-endpoint` Befehl, um den Endpunkt Ihres CodeArtifact Repositorys abzurufen.

   - Ersetze es *`my_domain`* durch deinen CodeArtifact Domainnamen.
   - *`111122223333`*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn du auf ein Repository in einer Domain zugreifst, die dir gehört, musst du nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
   - Ersetze es *`my_repo`* durch deinen CodeArtifact Repository-Namen.

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format npm
   ```

3. Aktualisieren Sie den `npmRegistryServer` Wert in Ihrer Datei .yarnrc.yml mit Ihrem Repository-Endpunkt.

   ```
   yarn config set npmRegistryServer
    "https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"
   ```

4. Rufen Sie ein CodeArtifact Autorisierungstoken ab und speichern Sie es in einer Umgebungsvariablen.

   > **Note**
   >
   > Der folgende Befehl gilt für macOS- oder Linux-Maschinen. Informationen zur Konfiguration von Umgebungsvariablen auf einem Windows-Computer finden Sie unterÜbergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen.

   - *`my_domain`*Ersetzen Sie es durch Ihren CodeArtifact Domainnamen.

- *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn du auf ein Repository in einer Domain zugreifst, die dir gehört, musst du nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
- Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

5. Verwenden Sie den `yarn config set` Befehl, um Ihr CodeArtifact Authentifizierungstoken zu Ihrer .yarnrc.yml-Datei hinzuzufügen. Ersetzen Sie die URL im folgenden Befehl durch Ihre Repository-Endpunkt-URL aus Schritt 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
 "${CODEARTIFACT_AUTH_TOKEN}"
```

6. Verwenden Sie den `yarn config set` Befehl, um den Wert von `npmAlwaysAuth` to festzulegen`true`. Ersetzen Sie die URL im folgenden Befehl durch Ihre Repository-Endpunkt-URL aus Schritt 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAlwaysAuth'
 "true"
```

Nach der Konfiguration sollte Ihre Konfigurationsdatei .yarnrc.yml einen Inhalt haben, der dem folgenden Codeausschnitt ähnelt.

```
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/"
```

Sie können den Befehl auch verwenden, um die `yarn config` Werte von und zu überprüfen. `npmRegistries` `npmRegistryServer`

Unterstützung für npm-Befehle

In den folgenden Abschnitten werden die npm-Befehle zusammengefasst, die von CodeArtifact Repositorys unterstützt werden, zusätzlich zu bestimmten Befehlen, die nicht unterstützt werden.

Inhalt

- Unterstützte Befehle, die mit einem Repository interagieren
- Unterstützte clientseitige Befehle
- Befehle werden nicht unterstützt

Unterstützte Befehle, die mit einem Repository interagieren

In diesem Abschnitt sind NPM-Befehle aufgeführt, bei denen der NPM-Client eine oder mehrere Anfragen an die Registry stellt, mit der er konfiguriert wurde (z. B. mit`npm config set registry`). Es wurde überprüft, dass diese Befehle korrekt funktionieren, wenn sie in einem Repository aufgerufen werden. CodeArtifact

Befehl	Beschreibung
Bugs	Versucht die Adresse der Bug-Tracker-URL eines Pakets zu erraten und versucht dann, es zu öffnen.
ci	Installiert ein Projekt mit einem Neuanfang.
verwerfen	Weist eine Version eines Pakets ab.
dist-tag	Ändert die Tags für die Paketverteilung.
Dokumente	Versucht, den Speicherort der Dokumentations-URL eines Pakets zu erraten, und versucht dann, es mit dem `--browser` Konfigura tionsparameter zu öffnen.

Befehl	Beschreibung
Arzt	Führt eine Reihe von Prüfungen durch, um sicherzustellen, dass Ihre npm-Installation über alles verfügt, was sie zur Verwaltung Ihrer JavaScript Pakete benötigt.
install	Installiert ein Paket.
install-ci-test	Installiert ein völlig neues Projekt und führt Tests durch. Alias:`npm cit`. Dieser Befehl wird ausgeführt und unmittelbar `npm ci` gefolgt von einem`npm test`.
installieren/testen	Installiert das Paket und führt Tests aus. Wird ausgeführt und unmittelbar `npm install` gefolgt von einem`npm test`.
veraltet	Überprüft die konfigurierte Registrierung, um festzustellen, ob installierte Pakete derzeit veraltet sind.
pingen	Pingt die konfigurierte oder angegebene npm-Registrierung an und überprüft die Authentif izierung.
veröffentlichen	Veröffentlicht eine Paketversion in der Registrie rung.
update	Errät den Speicherort der Repository-URL eines Pakets und versucht dann, es mit dem `--browser` Konfigurationsparameter zu öffnen.
anzeigen	Zeigt Paket-Metadaten an. Kann zum Drucken von Metadateneigenschaften verwendet werden.

Unterstützte clientseitige Befehle

Diese Befehle erfordern keine direkte Interaktion mit einem Repository und müssen daher auch CodeArtifact nichts tun, um sie zu unterstützen.

Befehl	Beschreibung
bauen	Baut ein Paket.
Cache	Manipuliert den Paket-Cache.
Abschluss	Aktiviert die Tabulatorvervollständigung in allen npm-Befehlen.
Konfiguration	Aktualisiert den Inhalt der Benutzer- und `npmrc` Globaldateien.
deduplizieren	Durchsucht den lokalen Paketbaum und versucht, die Struktur zu vereinfachen, indem Abhängigkeiten im Baum weiter nach oben verschoben werden, wo sie effektiver von mehreren abhängigen Paketen gemeinsam genutzt werden können.
bearbeiten	Bearbeitet ein installiertes Paket. Wählt eine Abhängigkeit im aktuellen Arbeitsverzeichnis aus und öffnet den Paketordner im Standarde ditor.
erkunden	Durchsucht ein installiertes Paket. Erzeugt eine Subshell im Verzeichnis des angegeben en installierten Pakets. Wenn ein Befehl angegeben ist, wird er in der Subshell ausgeführt, die dann sofort beendet wird.
help	Ruft Hilfe zu npm ab.
Hilfesuche	Durchsucht die NPM-Hilfedokumentation.

Befehl	Beschreibung
init	Erzeugt eine `package.json` Datei.
Link	Verweist symbolisch auf einen Paketordner.
ls	Listet die installierten Pakete auf.
packen	Erzeugt einen Tarball aus einem Paket.
prefix	Zeigt das Präfix an. Dies ist das nächstgel egene übergeordnete Verzeichnis, das eine `package.json` Datei enthält, sofern nicht `-g` ebenfalls angegeben.
pflaumen	Entfernt Pakete, die nicht in der Abhängigk eitsliste des übergeordneten Pakets aufgeführt sind.
neu aufbauen	Führt den `npm build` Befehl für die entsprech enden Ordner aus.
neu starten	Führt die Stopp-, Restart- und Startskripten eines Pakets sowie die zugehörigen Vor- und Nachskripte aus.
Root	Druckt den aktuellen `node_modules` Ordner in der Standardausgabe aus.
Run-Skript	Führt beliebige Paketskripten aus.
shrinkwrap	Sperrt abhängige Versionen für die Veröffent lichung.
deinstallieren	Deinstalliert ein Paket.

Befehle werden nicht unterstützt

Diese npm-Befehle werden von Repositorys nicht unterstützt CodeArtifact .

Befehl	Beschreibung	Hinweise
Zugriff	Legt die Zugriffsebene für veröffentlichte Pakete fest.	CodeArtifact verwendet ein Berechtigungsmodell, das sich vom öffentlichen npmjs-Rep ository unterscheidet.
Benutzer hinzufügen	Fügt ein Registrierungsbenu tzerkonto hinzu	CodeArtifact verwendet ein Benutzermodell, das sich vom öffentlichen npmjs-Repository unterscheidet.
Prüfung	Führt ein Sicherheitsaudit durch.	CodeArtifact verkauft derzeit keine Daten zu Sicherhei tslücken.
Haken	Verwaltet NPM-Hooks, einschließlich Hinzufüge n, Entfernen, Auflisten und Aktualisieren.	CodeArtifact unterstützt derzeit keinerlei Mechanism us zur Benachrichtigung über Änderungen.
login	Authentifiziert einen Benutzer. Dies ist ein Alias für `npm adduser`.	CodeArtifact verwendet ein Authentifizierungsmodell, das sich vom öffentlichen npmjs-Repository unterscheidet. Weitere Informationen finden Sie unter Authentifizierung mit npm.
logout	Meldet sich von der Registrie rung ab.	CodeArtifact verwendet ein Authentifizierungsmodell, das sich vom öffentlichen npmjs-Repository unterscheidet. Es gibt keine Möglichkeit, sich von einem CodeArtifact Repository abzumelden, aber Authentifizierungstoken laufen nach ihrer konfigurierbaren

Befehl	Beschreibung	Hinweise
		Ablaufzeit ab. Die Standardd auer des Tokens beträgt 12 Stunden.
Besitzer	Verwaltet Paketbesitzer.	CodeArtifact verwendet ein Berechtigungsmodell, das sich vom öffentlichen npmjs-Rep ository unterscheidet.
profile	Ändert die Einstellungen in Ihrem Registrierungsprofil.	CodeArtifact verwendet ein Benutzermodell, das sich vom öffentlichen npmjs-Repository unterscheidet.
search	Durchsucht die Registrie rung nach Paketen, die den Suchbegriffen entsprechen.	CodeArtifact unterstützt eingeschränkte Suchfunkt ionen mit dem Befehl list-pack ages.
Stern	Markiert deine Lieblings pakete.	CodeArtifact unterstützt derzeit keinerlei Favoriten-Mechanismus.
Sterne	Zeigt Pakete an, die als Favoriten markiert sind.	CodeArtifact unterstützt derzeit keinerlei Favoriten-Mechanismus.
Team	Verwaltet Organisationsteams und Teammitgliedschaften.	CodeArtifact verwendet ein Benutzer- und Gruppenmi tgliedschaftsmodell, das sich vom öffentlichen npmjs-Rep ository unterscheidet. Weitere Informationen finden Sie unter Identitäten (Benutzer, Gruppen und Rollen) im IAM-Benutzerhandbuch.

Befehl	Beschreibung	Hinweise
Token	Verwaltet Ihre Authentif izierungstoken.	CodeArtifact verwendet ein anderes Modell zum Abrufen von Authentifizierungstoken. Informationen finden Sie unter Authentifizierung mit npm.
Veröffentlichung rückgängig machen	Entfernt ein Paket aus der Registrierung.	CodeArtifact unterstützt nicht das Entfernen einer Paketvers ion aus einem Repository mit dem npm-Client. Sie können den Befehl delete-package-ver sion verwenden.
Whoami	Zeigt den NPM-Benut zernamen an.	CodeArtifact verwendet ein Benutzermodell, das sich vom öffentlichen npmjs-Repository unterscheidet.

Handhabung von NPM-Tags

npm-Registries unterstützen Tags, bei denen es sich um Zeichenkettenaliase für Paketversionen handelt. Sie können Tags verwenden, um einen Alias anstelle von Versionsnummern bereitzustellen. Sie könnten beispielsweise ein Projekt mit mehreren Entwicklungsströmen haben und für jeden Stream ein anderes Tag (z. B. `stablebeta`,`dev`,,`canary`) verwenden. Weitere Informationen finden Sie unter dist-tag auf der npm-Website.

Standardmäßig verwendet npm das `latest` Tag, um die aktuelle Version eines Pakets zu identifizieren. `npm install` *`pkg`*(ohne `@`*`version`* oder `@`*`tag`* Spezifizierer) installiert das neueste Tag. In der Regel verwenden Projekte das neueste Tag nur für stabile Release-Versionen. Andere Tags werden für instabile Versionen oder Vorabversionen verwendet.

Bearbeiten Sie Tags mit dem NPM-Client

Die drei `npm dist-tag` Befehle (`add`rm, und`ls`) funktionieren in CodeArtifact Repositorys genauso wie in der NPM-Standardregistrierung.

npm-Tags und die API CopyPackageVersions

Wenn Sie die `CopyPackageVersions` API verwenden, um eine npm-Paketversion zu kopieren, werden alle Tags, die dieser Version einen Alias geben, in das Ziel-Repository kopiert. Wenn eine kopierte Version über ein Tag verfügt, das auch im Ziel-Repository vorhanden ist, wird beim Kopiervorgang der Tag-Wert im Ziel-Repository so festgelegt, dass er dem Wert im Quell-Repository entspricht.

Nehmen wir zum Beispiel an, dass sowohl Repository S als auch Repository D eine einzige Version des `web-helper` Pakets mit dem neuesten Tag-Set enthalten, wie in dieser Tabelle dargestellt.

Repository	Package name	Paket-Tags
S	`web-helper`	neueste Version (Alias für Version 1.0.1)
D	`web-helper`	latest (Alias für Version 1.0.0)

`CopyPackageVersions`wird aufgerufen, um `web-helper` 1.0.1 von S nach D zu kopieren. Nach Abschluss des Vorgangs hat das `latest` Tag on `web-helper` im Projektarchiv D den Alias 1.0.1, nicht 1.0.0.

Wenn Sie Tags nach dem Kopieren ändern müssen, verwenden Sie den `npm dist-tag` Befehl, um Tags direkt im Ziel-Repository zu ändern. Weitere Informationen zur `CopyPackageVersions` API finden Sie unter Pakete zwischen Repositorys kopieren.

npm-Tags und Upstream-Repositorys

Wenn npm die Tags für ein Paket anfordert und Versionen dieses Pakets auch in einem Upstream-Repository vorhanden sind, werden die Tags CodeArtifact zusammengeführt, bevor sie an den Client zurückgegeben werden. Ein Repository mit dem Namen R hat beispielsweise ein Upstream-Repository mit dem Namen U. Die folgende Tabelle zeigt die Tags für ein Paket mit dem Namen`web-helper`, das in beiden Repositorys vorhanden ist.

Repository	Package name	Paket-Tags
R	`web-helper`	aktuell (Alias für Version 1.0.0)

Repository	Package name	Paket-Tags
U	`web-helper`	alpha (Alias für Version 1.0.1)

In diesem Fall empfängt der npm-Client, wenn er die Tags für das `web-helper` Paket aus dem Repository R abruft, sowohl die neuesten als auch die Alpha-Tags. Die Versionen, auf die die Tags verweisen, werden sich nicht ändern.

Wenn dasselbe Tag für dasselbe Paket sowohl im Upstream- als auch im Downstream-Repository vorhanden ist, wird das Tag CodeArtifact verwendet, das im Upstream-Repository vorhanden ist. Nehmen wir zum Beispiel an, dass die Tags auf Webhelper so geändert wurden, dass sie wie folgt aussehen.

Repository	Package name	Paket-Tags
R	`web-helper`	aktuell (Alias für Version 1.0.0)
U	`web-helper`	aktuell (Alias für Version 1.0.1)

In diesem Fall, wenn der npm-Client die Tags für das Paket web-helper aus dem Repository R abruft, verwendet das neueste Tag einen Alias für die Version 1.0.1, da sich diese im Upstream-Repository befindet. Dies macht es einfach, neue Paketversionen in einem Upstream-Repository, die noch nicht in einem Downstream-Repository vorhanden sind, durch Ausführen zu konsumieren. `npm update`

Die Verwendung des Tags im Upstream-Repository kann problematisch sein, wenn neue Versionen eines Pakets in einem Downstream-Repository veröffentlicht werden. Nehmen wir zum Beispiel an, dass das neueste Tag im Paket web-helper sowohl in R als auch in U identisch ist.

Repository	Package name	Paket-Tags
R	`web-helper`	neueste Version (Alias für Version 1.0.1)
U	`web-helper`	latest (Alias für Version 1.0.1)

Wenn Version 1.0.2 auf R veröffentlicht wird, aktualisiert npm das neueste Tag auf 1.0.2.

Repository	Package name	Paket-Tags
R	`web-helper`	neueste Version (Alias für Version 1.0.2)
U	`web-helper`	latest (Alias für Version 1.0.1)

Der npm-Client sieht diesen Tag-Wert jedoch nie, da der Wert von latest in U 1.0.1 ist. Wenn es unmittelbar nach der Veröffentlichung von 1.0.2 `npm install` gegen das Repository R ausgeführt wird, wird 1.0.1 anstelle der Version installiert, die gerade veröffentlicht wurde. Um die zuletzt veröffentlichte Version zu installieren, müssen Sie die genaue Paketversion wie folgt angeben.

```
npm install web-helper@1.0.2
```

Support für npm-kompatible Paketmanager

Diese anderen Paketmanager sind mit dem NPM-Paketformat CodeArtifact und dem NPM-Wire-Protokoll kompatibel und funktionieren mit diesen:

- pnpm-Paketmanager. Die neueste Version, mit der bestätigt wurde, dass sie funktioniert, CodeArtifact ist 3.3.4, die am 18. Mai 2019 veröffentlicht wurde.
- Garn-Paketmanager. Die neueste Version, mit der bestätigt wurde, dass sie funktioniert, CodeArtifact ist 1.21.1, die am 11. Dezember 2019 veröffentlicht wurde.

> **Note**
>
> Wir empfehlen die Verwendung von Yarn 2.x mit. CodeArtifact Yarn 1.x hat keine HTTP-Wiederholungen, was bedeutet, dass es anfälliger für zeitweilige Dienstfehler ist, die zu Statuscodes oder Fehlern der Stufe 500 führen. Es gibt keine Möglichkeit, eine andere Wiederholungsstrategie für Yarn 1.x zu konfigurieren, aber diese wurde in Yarn 2.x hinzugefügt. Sie können Yarn 1.x verwenden, müssen jedoch möglicherweise Wiederholungen auf höherer Ebene in Build-Skripten hinzufügen. Führen Sie beispielsweise Ihren Garn-Befehl in einer Schleife aus, sodass er es erneut versucht, wenn das Herunterladen von Paketen fehlschlägt.

Verwenden CodeArtifact mit NuGet

In diesen Themen wird beschrieben, wie `NuGet` Pakete mithilfe von verwendet und veröffentlicht CodeArtifact werden.

> **Note**
>
> AWS CodeArtifact unterstützt NuGetnur.exe-Version 4.8 und höher.

Themen

- CodeArtifact Mit Visual Studio verwenden
- Verwendung CodeArtifact mit der Nuget- oder Dotnet-CLI
- NuGet Normalisierung von Paketnamen, Version und Assetnamen
- NuGet kompatibilität

CodeArtifact Mit Visual Studio verwenden

Sie können Pakete CodeArtifact direkt in Visual Studio mit dem CodeArtifact Credential Provider verwenden. Der Credential Provider vereinfacht die Einrichtung und Authentifizierung Ihrer CodeArtifact Repositorys in Visual Studio und ist im verfügbar. AWS Toolkit for Visual Studio

> **Note**
>
> Der AWS Toolkit for Visual Studio ist nicht für Visual Studio für Mac verfügbar.

Informationen zur Konfiguration und Verwendung NuGet mit CLI-Tools finden Sie unterVerwendung CodeArtifact mit der Nuget- oder Dotnet-CLI.

Themen

- Konfigurieren Sie Visual Studio mit dem CodeArtifact Credential Provider
- Verwenden Sie die Visual Studio Package Manager-Konsole

Konfigurieren Sie Visual Studio mit dem CodeArtifact Credential Provider

Der CodeArtifact Credential Provider vereinfacht die Einrichtung und die fortgesetzte Authentifizierung zwischen Visual Studio CodeArtifact und Visual Studio. CodeArtifact Authentifizierungstoken sind maximal 12 Stunden gültig. Um zu vermeiden, dass das Token während der Arbeit in Visual Studio manuell aktualisiert werden muss, ruft der Anmeldeinformationsanbieter regelmäßig ein neues Token ab, bevor das aktuelle Token abläuft.

> **Important**
>
> Um den Anmeldeinformationsanbieter zu verwenden, stellen Sie sicher, dass alle vorhandenen AWS CodeArtifact Anmeldeinformationen aus Ihrer `nuget.config` Datei gelöscht werden, die möglicherweise manuell oder durch Ausführen von `aws codeartifact login` to configure hinzugefügt wurden. NuGet

Verwenden Sie CodeArtifact in Visual Studio mit dem AWS Toolkit for Visual Studio

1. Installieren Sie das AWS Toolkit for Visual Studio mithilfe der folgenden Schritte. Das Toolkit ist mithilfe dieser Schritte mit Visual Studio 2017 und 2019 kompatibel. AWS CodeArtifact unterstützt Visual Studio 2015 und früher nicht.

   1. Das Toolkit for Visual Studio für Visual Studio 2017 und Visual Studio 2019 wird im Visual Studio Marketplace vertrieben. Sie können das Toolkit auch in Visual Studio mithilfe von Tools ≫ Erweiterungen und Updates (Visual Studio 2017) oder Erweiterungen ≫ Erweiterungen verwalten (Visual Studio 2019) installieren und aktualisieren.
   2. Nachdem das Toolkit installiert wurde, öffnen Sie es, indem Sie im Menü Ansicht die Option AWS Explorer wählen.
2. Konfigurieren Sie das Toolkit for Visual Studio mit Ihren AWS Anmeldeinformationen, indem Sie die Schritte unter Bereitstellen von AWS Anmeldeinformationen im AWS Toolkit for Visual Studio Benutzerhandbuch befolgen.
3. (Optional) Legen Sie das AWS Profil fest, mit CodeArtifact dem Sie es verwenden möchten. Wenn nicht festgelegt, CodeArtifact wird das Standardprofil verwendet. Um das Profil festzulegen, gehen Sie zu Tools > NuGet Package Manager > CodeArtifact AWS Profil auswählen.
4. Fügen Sie Ihr CodeArtifact Repository als Paketquelle in Visual Studio hinzu.

1. Navigieren Sie im AWS Explorer-Fenster zu Ihrem Repository, klicken Sie mit der rechten Maustaste und wählen Sie`Copy NuGet Source Endpoint`.
2. Verwenden Sie den Befehl Tools > Optionen und scrollen Sie zu NuGet Package Manager.
3. Wählen Sie den Knoten Paketquellen aus.
4. Wählen Sie + aus, bearbeiten Sie den Namen, fügen Sie den in Schritt 3a kopierten Repository-URL-Endpunkt in das Feld Quelle ein und wählen Sie Aktualisieren aus.
5. Markieren Sie das Kontrollkästchen für Ihre neu hinzugefügte Paketquelle, um sie zu aktivieren.

> **Note**
>
> Wir empfehlen, Ihrem CodeArtifact Repository eine externe Verbindung zu NuGet.org hinzuzufügen und die Nuget.org-Paketquelle in Visual Studio zu deaktivieren. Wenn Sie eine externe Verbindung verwenden, werden alle von NuGet.org abgerufenen Pakete in Ihrem Repository gespeichert. CodeArtifact Falls NuGet.org nicht mehr verfügbar ist, sind Ihre Anwendungsabhängigkeiten weiterhin für CI-Builds und lokale Entwicklung verfügbar. Weitere Informationen zu externen Verbindungen finden Sie unterEin CodeArtifact Repository mit einem öffentlichen Repository Connect.

5. Starten Sie Visual Studio neu, damit die Änderungen wirksam werden.

Nach der Konfiguration kann Visual Studio Pakete aus Ihrem CodeArtifact Repository, einem seiner Upstream-Repositorys oder NuGetvon.org verwenden, wenn Sie eine externe Verbindung hinzugefügt haben. Weitere Informationen zum Durchsuchen und Installieren von NuGet Paketen in Visual Studio finden Sie in der NuGet Dokumentation unter Installieren und Verwalten von Paketen in Visual Studio mithilfe des NuGet Paket-Managers.

Verwenden Sie die Visual Studio Package Manager-Konsole

Die Visual Studio Package Manager-Konsole verwendet nicht die Visual Studio-Version des CodeArtifact Credential Providers. Um sie zu verwenden, müssen Sie den Befehlszeilenanbieter für Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Verwendung CodeArtifact mit der Nuget- oder Dotnet-CLI.

Verwendung CodeArtifact mit der Nuget- oder Dotnet-CLI

Sie können CLI-Tools wie `nuget` und verwenden`dotnet`, um Pakete zu veröffentlichen und zu konsumieren CodeArtifact. Dieses Dokument enthält Informationen zur Konfiguration der CLI-Tools und deren Verwendung zum Veröffentlichen oder Verwenden von Paketen.

Themen

- Konfigurieren Sie die Nuget- oder Dotnet-CLI
- Verbrauchen Sie Pakete NuGet von CodeArtifact
- Veröffentlichen Sie NuGet Pakete in CodeArtifact
- CodeArtifact NuGet Referenz zum Credential Provider
- CodeArtifact NuGet Versionen des Credential Providers

Konfigurieren Sie die Nuget- oder Dotnet-CLI

Sie können die Nuget- oder Dotnet-CLI mit dem CodeArtifact NuGet Credential Provider, mit oder manuell konfigurieren. AWS CLI Die Konfiguration NuGet mit dem Credential Provider wird dringend empfohlen, um die Einrichtung zu vereinfachen und die Authentifizierung fortzusetzen.

Methode 1: Konfiguration mit dem CodeArtifact NuGet Credential Provider

Der CodeArtifact NuGet Credential Provider vereinfacht die Authentifizierung und Konfiguration CodeArtifact mit NuGet CLI-Tools. CodeArtifact Authentifizierungstoken sind maximal 12 Stunden gültig. Um zu vermeiden, dass das Token manuell aktualisiert werden muss, während die Nuget- oder Dotnet-CLI verwendet wird, ruft der Anmeldeinformationsanbieter regelmäßig ein neues Token ab, bevor das aktuelle Token abläuft.

> **Important**
>
> Um den Anbieter für Anmeldeinformationen zu verwenden, stellen Sie sicher, dass alle vorhandenen AWS CodeArtifact Anmeldeinformationen aus Ihrer `nuget.config` Datei gelöscht werden, die möglicherweise manuell oder durch Ausführen von to configure zuvor hinzugefügt wurden. `aws codeartifact login` NuGet

Installieren und konfigurieren Sie den CodeArtifact NuGet Credential Provider

dotnet

1. Laden Sie die neueste Version von AWS herunter. CodeArtifact. NuGet. CredentialProvider Tool von NuGet .org mit dem folgenden `dotnet` Befehl.

   ```
   dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
   ```

2. Verwenden Sie den `codeartifact-creds install` Befehl, um den Credential Provider in den NuGet Plugins-Ordner zu kopieren.

   ```
   dotnet codeartifact-creds install
   ```

3. (Optional): Legen Sie das AWS Profil fest, das Sie mit dem Anmeldeinformationsanbieter verwenden möchten. Wenn nicht festgelegt, verwendet der Anmeldeinformationsanbieter das Standardprofil. Weitere Informationen zu AWS CLI Profilen finden Sie unter Benannte Profile.

   ```
   dotnet codeartifact-creds configure set profile profile_name
   ```

nuget

Führen Sie die folgenden Schritte aus, um den CodeArtifact NuGet Credential Provider mithilfe der NuGet CLI aus einem Amazon S3 S3-Bucket zu installieren und zu konfigurieren. Der Anmeldeinformationsanbieter verwendet das AWS CLI Standardprofil. Weitere Informationen zu Profilen finden Sie unter Benannte Profile.

1. Laden Sie die neueste Version des CodeArtifact NuGet Credential Providers (codeartifact-nuget-credentialprovider.zip) aus einem Amazon S3 S3-Bucket herunter.

   Informationen zum Anzeigen und Herunterladen früherer Versionen finden Sie unter. CodeArtifact NuGet Versionen des Credential Providers

2. Entpacken Sie die Datei.
3. Kopieren Sie die AWS. CodeArtifact. NuGetCredentialProviderOrdner aus dem Netfx-Ordner zu `%user_profile%/.nuget/plugins/netfx/` unter Windows oder `~/.nuget/plugins/netfx` unter Linux oder macOS.
4. Kopieren Sie die AWS. CodeArtifact. NuGetCredentialProviderOrdner aus dem Netcore-Ordner zu `%user_profile%/.nuget/plugins/netcore/` unter Windows oder `~/.nuget/plugins/netcore` unter Linux oder macOS.

Nachdem Sie ein Repository erstellt und den Anmeldeinformationsanbieter konfiguriert haben, können Sie die Tools `nuget` oder `dotnet` CLI verwenden, um Pakete zu installieren und zu veröffentlichen. Weitere Informationen erhalten Sie unter Verbrauchen Sie Pakete NuGet von CodeArtifact und Veröffentlichen Sie NuGet Pakete in CodeArtifact.

Methode 2: Konfigurieren Sie Nuget oder Dotnet mit dem Login-Befehl

Der `codeartifact login` Befehl im AWS CLI fügt Ihrer NuGet Konfigurationsdatei einen Repository-Endpunkt und ein Autorisierungstoken hinzu, sodass Nuget oder Dotnet eine Verbindung zu Ihrem CodeArtifact Repository herstellen können. Dadurch wird die NuGet Konfiguration auf Benutzerebene geändert, die sich `%appdata%\NuGet\NuGet.Config` für Windows `~/.config/NuGet/NuGet.Config` und/oder `~/.nuget/NuGet/NuGet.Config` für Mac/Linux befindet. Weitere Informationen zu NuGet Konfigurationen finden Sie unter Allgemeine Konfigurationen. NuGet

Konfigurieren Sie Nuget oder Dotnet mit dem Befehl **`login`**

1. Konfigurieren Sie Ihre AWS Anmeldeinformationen für die AWS CLI, wie unter beschrieben. Erste Schritte mit CodeArtifact
2. Stellen Sie sicher, dass das NuGet CLI-Tool (`nuget`oder`dotnet`) ordnungsgemäß installiert und konfiguriert wurde. Anweisungen finden Sie in der Nuget - oder Dotnet-Dokumentation.
3. Verwenden Sie den CodeArtifact `login` Befehl, um Anmeldeinformationen für die Verwendung mit abzurufen. NuGet

> **Note**
>
> Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie keine Daten angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

dotnet

> **Important**
>
> Linux- und macOS-Benutzer: Da die Verschlüsselung auf Nicht-Windows-Plattformen nicht unterstützt wird, werden Ihre abgerufenen Anmeldeinformationen als Klartext in Ihrer Konfigurationsdatei gespeichert.

```
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
```

Der Login-Befehl wird:

- Rufen Sie CodeArtifact mithilfe Ihrer AWS Anmeldeinformationen ein Autorisierungstoken ab.
- Aktualisieren Sie Ihre NuGet Konfiguration auf Benutzerebene mit einem neuen Eintrag für Ihre NuGet Paketquelle. Die Quelle, die auf Ihren CodeArtifact Repository-Endpunkt verweist, wird aufgerufen*domain_name/repo_name*.

Der Standardzeitraum für die Autorisierung nach dem Aufruf `login` beträgt 12 Stunden. Sie `login` müssen aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

Nachdem Sie ein Repository erstellt und die Authentifizierung konfiguriert haben, können Sie die `msbuild` CLI-Clients `nugetdotnet`, oder verwenden, um Pakete zu installieren und zu veröffentlichen. Weitere Informationen erhalten Sie unter Verbrauchen Sie Pakete NuGet von CodeArtifact und Veröffentlichen Sie NuGet Pakete in CodeArtifact.

Methode 3: Konfigurieren Sie Nuget oder Dotnet ohne den Login-Befehl

Für die manuelle Konfiguration müssen Sie Ihrer NuGet Konfigurationsdatei einen Repository-Endpunkt und ein Autorisierungstoken hinzufügen, damit Nuget oder Dotnet eine Verbindung zu Ihrem CodeArtifact Repository herstellen können.

Konfigurieren Sie Nuget oder Dotnet manuell, um eine Verbindung zu Ihrem Repository herzustellen. CodeArtifact

1. Ermitteln Sie Ihren CodeArtifact Repository-Endpunkt mithilfe des Befehls `get-repository-endpoint` AWS CLI .

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format nuget
```

Beispielausgabe:

```
{
   "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/nuget/my_repo/"
}
```

2. Holen Sie sich von Ihrem Paketmanager mithilfe des `get-authorization-token` AWS CLI Befehls ein Autorisierungstoken, um eine Verbindung zu Ihrem Repository herzustellen.

```
aws codeartifact get-authorization-token --domain my_domain
```

Beispielausgabe:

```
{
   "authorizationToken": "eyJ2I...viOw",
   "expiration": 1601616533.0
}
```

3. Erstellen Sie die vollständige Repository-Endpunkt-URL, indem `/v3/index.json` Sie sie an die `get-repository-endpoint` in Schritt 3 zurückgegebene URL anhängen.
4. Konfigurieren Sie Nuget oder Dotnet so, dass der Repository-Endpunkt aus Schritt 1 und das Autorisierungstoken aus Schritt 2 verwendet werden.

> **Note**
>
> Die Quell-URL muss auf enden, damit Nuget oder Dotnet erfolgreich eine Verbindung zu einem Repository herstellen können. `/v3/index.json` CodeArtifact

dotnet

Linux- und macOS-Benutzer: Da Verschlüsselung auf Nicht-Windows-Plattformen nicht unterstützt wird, müssen Sie dem folgenden Befehl das `--store-password-in-clear-text` Flag hinzufügen. Beachten Sie, dass dadurch Ihr Passwort als Klartext in Ihrer Konfigurationsdatei gespeichert wird.

```
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**
>
> Verwenden Sie den `dotnet nuget update source` Befehl, um eine bestehende Quelle zu aktualisieren.

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
```

Beispielausgabe:

```
Package source with Name: domain_name/repo_name added successfully.
```

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den `codeartifact.region.on.aws` Endpunkt.

Verbrauchen Sie Pakete NuGet von CodeArtifact

Sobald Sie NuGet mit konfiguriert haben CodeArtifact, können Sie NuGet Pakete nutzen, die in Ihrem CodeArtifact Repository oder einem seiner Upstream-Repositorys gespeichert sind.

Um eine Paketversion aus einem CodeArtifact Repository oder einem seiner Upstream-Repositorys mit `nuget` oder zu verwenden`dotnet`, führen Sie den folgenden Befehl aus und ersetzen *`packageName`* Sie ihn durch den Namen des Pakets, das Sie nutzen möchten, und *`packageSourceName`* durch den Quellnamen für Ihr CodeArtifact Repository in Ihrer NuGet Konfigurationsdatei. Wenn Sie den `login` Befehl zur NuGet Konfiguration Ihrer Konfiguration verwendet haben, lautet *`domain_name/repo_name`* der Quellname.

> **Note**
>
> Wenn ein Paket angefordert wird, speichert der NuGet Client im Cache, welche Versionen dieses Pakets existieren. Aufgrund dieses Verhaltens kann eine Installation für ein Paket fehlschlagen, das zuvor angefordert wurde, bevor die gewünschte Version verfügbar wurde. Um diesen Fehler zu vermeiden und ein vorhandenes Paket erfolgreich zu installieren, können Sie entweder den NuGet Cache vor einer Installation mit `nuget locals all --clear` oder leeren oder vermeiden`dotnet nuget locals all --clear`, dass der Cache während der `restore` Befehle `install` und verwendet wird, indem Sie die `-NoCache` Option für `nuget` oder die `--no-cache` Option für angeben`dotnet`.

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

Um eine bestimmte Version eines Pakets zu installieren

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

Weitere Informationen finden Sie unter Pakete mit der CLI nuget.exe verwalten oder Pakete mit der dotnet-CLI installieren und verwalten in der Microsoft-Dokumentation.

Konsumieren Sie NuGet Pakete von .org NuGet

Sie können NuGet Pakete von NuGet.org über ein CodeArtifact Repository konsumieren, indem Sie das Repository mit einer externen Verbindung zu NuGet.org konfigurieren. Pakete, die von NuGet.org konsumiert werden, werden aufgenommen und in Ihrem CodeArtifact Repository gespeichert. Weitere Hinweise zum Hinzufügen externer Verbindungen finden Sie unterEin CodeArtifact Repository mit einem öffentlichen Repository Connect.

Veröffentlichen Sie NuGet Pakete in CodeArtifact

Sobald Sie NuGet mit konfiguriert haben, können Sie `nuget` oder verwenden CodeArtifact, um Paketversionen in Repositorys `dotnet` zu CodeArtifact veröffentlichen.

Um eine Paketversion in ein CodeArtifact Repository zu übertragen, führen Sie den folgenden Befehl mit dem vollständigen Pfad zu Ihrer `.nupkg` Datei und dem Quellnamen für Ihr CodeArtifact Repository in Ihrer NuGet Konfigurationsdatei aus. Wenn Sie den `login` Befehl zur NuGet Konfiguration Ihrer Konfiguration verwendet haben, lautet der Quellname`domain_name/repo_name`.

> **Note**
>
> Sie können ein NuGet Paket erstellen, wenn Sie keins veröffentlichen möchten. Weitere Informationen finden Sie in der Microsoft-Dokumentation unter Workflow zur Paketerstellung.

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 Referenz zum Credential Provider

Der CodeArtifact NuGet Credential Provider erleichtert die Konfiguration und Authentifizierung NuGet Ihrer Repositorys. CodeArtifact

CodeArtifact NuGet Befehle des Credential Providers

Dieser Abschnitt enthält die Liste der Befehle für den CodeArtifact NuGet Credential Provider. Diesen Befehlen muss ein Präfix `dotnet codeartifact-creds` wie im folgenden Beispiel vorangestellt werden.

```
dotnet codeartifact-creds command
```

- `configure set profile profile`: Konfiguriert den Anmeldeinformationsanbieter für die Verwendung des bereitgestellten Profils. AWS
- `configure unset profile`: Entfernt das konfigurierte Profil, falls es gesetzt ist.
- `install`: Kopiert den Anmeldeinformationsanbieter in den `plugins` Ordner.
- `install --profile profile`: Kopiert den Anbieter für Anmeldeinformationen in den `plugins` Ordner und konfiguriert ihn so, dass er das angegebene Profil verwendet. AWS
- `uninstall`: Deinstalliert den Anmeldeinformationsanbieter. Dadurch werden die Änderungen an der Konfigurationsdatei nicht entfernt.
- `uninstall --delete-configuration`: Deinstalliert den Anmeldeinformationsanbieter und entfernt alle Änderungen an der Konfigurationsdatei.

CodeArtifact NuGet Logs des Credential Providers

Um die Protokollierung für den CodeArtifact NuGet Credential Provider zu aktivieren, müssen Sie die Protokolldatei in Ihrer Umgebung einrichten. Die Logs des Credential Providers enthalten hilfreiche Debugging-Informationen wie:

- Das AWS Profil, das zum Herstellen von Verbindungen verwendet wird
- Irgendwelche Authentifizierungsfehler

- Wenn der angegebene Endpunkt keine CodeArtifact URL ist

Legen Sie die Logdatei des CodeArtifact NuGet Credential Providers fest

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

Nachdem die Protokolldatei eingerichtet wurde, hängt jeder `codeartifact-creds` Befehl seine Protokollausgabe an den Inhalt dieser Datei an.

CodeArtifact NuGet Versionen des Credential Providers

Die folgende Tabelle enthält Informationen zum Versionsverlauf und Download-Links für den CodeArtifact NuGet Credential Provider.

Version	Änderungen	Datum der Veröffentlichung	Link herunterladen (S3)
1.0.2 (aktuell)	Aktualisierte Abhängigkeiten	26.06.2024	Laden Sie v1.0.2 herunter
1.0.1	Unterstützung für Net5-, Net6- und SSO-Profile hinzugefügt	05.03.2022	Laden Sie v1.0.1 herunter
1.0.0	Erste Version des CodeArtifact NuGet Credential Providers	20.11.2020	Laden Sie v1.0.0 herunter

NuGet Normalisierung von Paketnamen, Version und Assetnamen

CodeArtifact normalisiert Paket- und Assetnamen und Paketversionen, bevor sie gespeichert werden, was bedeutet, dass sich die Namen oder Versionen in denen unterscheiden CodeArtifact können, die bei der Veröffentlichung des Pakets oder Assets angegeben wurden.

Normalisierung von Paketnamen: CodeArtifact normalisiert NuGet Paketnamen, indem alle Buchstaben in Kleinbuchstaben umgewandelt werden.

Normalisierung der Paketversion: CodeArtifact normalisiert NuGet Paketversionen nach demselben Muster wie. NuGet Die folgenden Informationen stammen aus den normalisierten Versionsnummern aus der Dokumentation. NuGet

- Führende Nullen werden aus den Versionsnummern entfernt:
  - `1.00`wird behandelt als `1.0`
  - `1.01.1`wird behandelt als `1.1.1`
  - `1.00.0.1`wird behandelt als `1.0.0.1`
- Eine Null im vierten Teil der Versionsnummer wird weggelassen:
  - `1.0.0.0`wird behandelt als `1.0.0`
  - `1.0.01.0`wird behandelt als `1.0.1`
- SemVer Die Build-Metadaten von 2.0.0 wurden entfernt:
  - `1.0.7+r3456`wird behandelt als `1.0.7`

Normalisierung von Paket-Assetnamen: CodeArtifact Konstruiert den NuGet Paket-Assetnamen aus dem normalisierten Paketnamen und der Paketversion.

Der nicht normalisierte Paketname und der Versionsname können mit API- und CLI-Anfragen verwendet werden, da CodeArtifact eine Normalisierung der Paketnamen- und Versionseingaben für diese Anfragen durchgeführt wird. Zum Beispiel `--version 12.0.03.0` würden Eingaben von `--package Newtonsoft.JSON` und normalisiert werden und ein Paket zurückgeben, das den normalisierten Paketnamen und die Version von hat. `newtonsoft.json` `12.0.3`

Sie müssen den normalisierten Paket-Assetnamen in API- und CLI-Anfragen verwenden, da die `--asset` Eingabe CodeArtifact nicht normalisiert wird.

Sie müssen in normalisierte Namen und Versionen verwenden. ARNs

Verwenden Sie den Befehl, um den normalisierten Namen eines Pakets zu finden. `aws codeartifact list-packages` Weitere Informationen finden Sie unter Listet Paketnamen auf.

Verwenden Sie den Befehl, um den nicht normalisierten Namen eines Pakets zu finden. `aws codeartifact describe-package-version` Der nicht normalisierte Name des Pakets wird in dem Feld zurückgegeben. `displayName` Weitere Informationen finden Sie unter Details und Abhängigkeiten der Paketversion anzeigen und aktualisieren.

NuGet kompatibilität

Dieses Handbuch enthält Informationen zur CodeArtifact Kompatibilität mit verschiedenen NuGet Tools und Versionen.

Themen

- Allgemeine NuGet Kompatibilität
- NuGet Unterstützung für die Befehlszeile

Allgemeine NuGet Kompatibilität

AWS CodeArtifact unterstützt NuGet 4.8 und höher.

AWS CodeArtifact unterstützt nur Version 3 des NuGet HTTP-Protokolls. Dies bedeutet, dass einige CLI-Befehle, die Version V2 des Protokolls verwenden, nicht unterstützt werden. Weitere Informationen finden Sie im Abschnitt Unterstützung für den Befehl nuget.exe.

AWS CodeArtifact unterstützt PowerShellGet 2.x nicht.

NuGet Unterstützung für die Befehlszeile

AWS CodeArtifact unterstützt die CLI-Tools NuGet (`nuget.exe`) und .NET Core (`dotnet`).

Unterstützung für den Befehl nuget.exe

Da CodeArtifact nur das HTTP-Protokoll V3 NuGet von unterstützt wird, funktionieren die folgenden Befehle nicht, wenn sie für CodeArtifact Ressourcen verwendet werden:

- `list`: Der `nuget list` Befehl zeigt eine Liste von Paketen aus einer bestimmten Quelle an. Um eine Liste der Pakete in einem CodeArtifact Repository abzurufen, können Sie den Listet Paketnamen auf Befehl von der AWS CLI verwenden.

CodeArtifact Mit Python verwenden

In diesen Themen wird beschriebenpip, wie Sie den Python-Paketmanager und twine das Python-Hilfsprogramm zur Paketveröffentlichung mit verwenden CodeArtifact.

Themen

- Konfigurieren und verwenden Sie Pip mit CodeArtifact
- Konfigurieren und verwenden Sie Twine mit CodeArtifact
- Normalisierung von Python-Paketnamen
- Python-Kompatibilität
- Python-Pakete von Upstreams und externen Verbindungen anfordern

Konfigurieren und verwenden Sie Pip mit CodeArtifact

pip ist der Paket-Installer für Python-Pakete. Um Pip zur Installation von Python-Paketen aus Ihrem CodeArtifact Repository zu verwenden, müssen Sie zuerst den Pip-Client mit Ihren CodeArtifact Repository-Informationen und Anmeldeinformationen konfigurieren.

pip kann nur zur Installation von Python-Paketen verwendet werden. Um Python-Pakete zu veröffentlichen, können Sie Twine verwenden. Weitere Informationen finden Sie unter Konfigurieren und verwenden Sie Twine mit CodeArtifact.

Konfigurieren Sie pip mit dem Befehl **`login`**

Konfigurieren Sie zunächst Ihre AWS Anmeldeinformationen für die AWS CLI, wie unter beschriebenErste Schritte mit CodeArtifact. Verwenden Sie dann den CodeArtifact `login` Befehl, um die Anmeldeinformationen abzurufen und `pip` mit ihnen zu konfigurieren.

> **Note**
>
> Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie dies nicht angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

Führen Sie den folgenden Befehl aus, um Pip zu konfigurieren.

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

`login`ruft CodeArtifact mithilfe Ihrer AWS Anmeldeinformationen ein Autorisierungstoken ab. Der `login` Befehl wird `pip` für die Verwendung mit konfiguriert, CodeArtifact indem er bearbeitet`~/.config/pip/pip.conf`, `index-url` um den Wert auf das in der `--repository` Option angegebene Repository zu setzen.

Der Standardzeitraum für die Autorisierung nach einem Anruf `login` beträgt 12 Stunden und `login` muss aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

Konfigurieren Sie pip ohne den Login-Befehl

Wenn Sie den `login` Befehl nicht zur Konfiguration verwenden könnenpip, können Sie. `pip config`

1. Verwenden Sie den AWS CLI , um ein neues Autorisierungstoken abzurufen.

   > **Note**
   >
   > Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie das `--domain-owner` nicht angeben. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

   ```
   CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

2. Dient `pip config` zum Festlegen der CodeArtifact Registrierungs-URL und der Anmeldeinformationen. Mit dem folgenden Befehl wird nur die aktuelle Umgebungskonfigurationsdatei aktualisiert. Um die systemweite Konfigurationsdatei zu aktualisieren, ersetzen Sie sie durch`site`. `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**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den Endpunkt. `codeartifact.region.on.aws`

> **Important**
>
> Die Registrierungs-URL muss mit einem Schrägstrich (/) enden. Andernfalls können Sie keine Verbindung zum Repository herstellen.

Beispiel für eine Pip-Konfigurationsdatei

Im Folgenden finden Sie ein Beispiel für eine `pip.conf` Datei nach dem Festlegen der CodeArtifact Registrierungs-URL und der Anmeldeinformationen.

```
[global]
index-url = https://aws:eyJ2ZX...@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/pypi/my_repo/simple/
```

Führen Sie pip aus

Um `pip` Befehle auszuführen, müssen Sie `pip` mit CodeArtifact konfigurieren. Weitere Informationen finden Sie in der folgenden Dokumentation.

1. Folgen Sie den Schritten im Einrichtung mit AWS CodeArtifact Abschnitt, um Ihr AWS Konto, Ihre Tools und Berechtigungen zu konfigurieren.
2. Folgen Sie zur Konfiguration den Schritten `twine` unterKonfigurieren und verwenden Sie Twine mit CodeArtifact.

Unter der Annahme, dass ein Paket in Ihrem Repository oder einem seiner Upstream-Repositorys vorhanden ist, können Sie es mit `pip install` installieren. Verwenden Sie beispielsweise den folgenden Befehl, um das `requests` Paket zu installieren.

```
pip install requests
```

Verwenden Sie die `-i` Option, um vorübergehend zur Installation von Paketen von https://pypi.org statt von Ihrem CodeArtifact Repository zurückzukehren.

```
pip install -i https://pypi.org/simple requests
```

Konfigurieren und verwenden Sie Twine mit CodeArtifact

Twine ist ein Hilfsprogramm zur Paketveröffentlichung für Python-Pakete. Um Twine zu verwenden, um Python-Pakete in Ihrem CodeArtifact Repository zu veröffentlichen, müssen Sie Twine zuerst mit Ihren CodeArtifact Repository-Informationen und Anmeldeinformationen konfigurieren.

Twine kann nur zum Veröffentlichen von Python-Paketen verwendet werden. Um Python-Pakete zu installieren, können Sie pip verwenden. Weitere Informationen finden Sie unter Konfigurieren und verwenden Sie Pip mit CodeArtifact.

Konfigurieren Sie Twine mit dem Befehl **`login`**

Konfigurieren Sie zunächst Ihre AWS Anmeldeinformationen für die AWS CLI, wie unter beschriebenErste Schritte mit CodeArtifact. Verwenden Sie dann den CodeArtifact `login` Befehl, um Anmeldeinformationen abzurufen und Twine damit zu konfigurieren.

> **Note**
>
> Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie dies nicht angeben. `--domain-owner` Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

Führen Sie den folgenden Befehl aus, um Twine zu konfigurieren.

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

`login`ruft CodeArtifact mithilfe Ihrer AWS Anmeldeinformationen ein Autorisierungstoken ab. Der `login` Befehl konfiguriert Twine für die Verwendung mit, CodeArtifact indem es bearbeitet wird`~/.pypirc`, um das in der `--repository` Option angegebene Repository mit Anmeldeinformationen hinzuzufügen.

Der Standardzeitraum für die Autorisierung nach dem Aufrufen `login` beträgt 12 Stunden und `login` muss aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

Konfigurieren Sie Twine ohne den Befehl **`login`**

Wenn Sie den `login` Befehl nicht zur Konfiguration von Twine verwenden können, können Sie die `~/.pypirc` Datei- oder Umgebungsvariablen verwenden. Um die `~/.pypirc` Datei zu verwenden, fügen Sie ihr die folgenden Einträge hinzu. Das Passwort muss ein Authentifizierungstoken sein, das von der `get-authorization-token` API abgerufen wurde.

```
[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**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den Endpunkt. `codeartifact.region.on.aws`

Gehen Sie wie folgt vor, um Umgebungsvariablen zu verwenden.

> **Note**
>
> Wenn Sie auf ein Repository in einer Domäne zugreifen, die Sie besitzen, müssen Sie das nicht angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

```
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 111122223333 --repository my_repo --format pypi --query
 repositoryEndpoint --output text`
```

Führen Sie Twine aus

Bevor Sie Twine zum Veröffentlichen von Python-Paket-Assets verwenden, müssen Sie zunächst CodeArtifact Berechtigungen und Ressourcen konfigurieren.

1. Folgen Sie den Schritten im Einrichtung mit AWS CodeArtifact Abschnitt, um Ihr AWS Konto, Ihre Tools und Berechtigungen zu konfigurieren.
2. Konfigurieren Sie Twine, indem Sie den Schritten unter Konfigurieren Sie Twine mit dem Befehl `login` oder Konfigurieren Sie Twine ohne den Befehl `login` folgen.

Nachdem Sie Twine konfiguriert haben, können Sie Befehle ausführen`twine`. Verwenden Sie den folgenden Befehl, um Python-Paket-Assets zu veröffentlichen.

```
twine upload --repository codeartifact mypackage-1.0.tgz
```

Informationen zum Erstellen und Verpacken Ihrer Python-Anwendung finden Sie unter Generieren von Distributionsarchiven auf der Python Packaging Authority-Website.

Normalisierung von Python-Paketnamen

CodeArtifact normalisiert Paketnamen, bevor sie gespeichert werden, was bedeutet, dass sich die Paketnamen in CodeArtifact möglicherweise von dem Namen unterscheiden, der bei der Veröffentlichung des Pakets angegeben wurde.

Bei Python-Paketen wird der Paketname bei der Normalisierung in Kleinbuchstaben geschrieben und alle Instanzen der Zeichen`.`,`-`, und `_` werden durch ein einzelnes Zeichen ersetzt. `-` Die Paketnamen `pigeon_cli` und und `pigeon.cli` werden also normalisiert und gespeichert als. `pigeon-cli` Der nicht normalisierte Name kann von pip und twine verwendet werden, aber der normalisierte Name muss in CodeArtifact CLI- oder API-Anfragen (wie) und in verwendet werden. `list-package-versions` ARNs Weitere Informationen zur Normalisierung von Python-Paketnamen finden Sie unter PEP 503 in der Python-Dokumentation.

Python-Kompatibilität

CodeArtifact unterstützt keine PyPIs `XML-RPC` oder. `JSON` APIs

CodeArtifact unterstützt PyPIs `Legacy` APIs, mit Ausnahme der API. `simple` CodeArtifact Unterstützt zwar nicht den `/simple/` API-Endpunkt, aber den `/simple/<project>/` Endpunkt.

Weitere Informationen finden Sie im Folgenden im GitHub Repository der Python Packaging Authority.

- XML-RPC-API
- JSON-API
- Legacy-API

Unterstützung für Pip-Befehle

In den folgenden Abschnitten werden die Pip-Befehle zusammengefasst, die von CodeArtifact Repositorys unterstützt werden, zusätzlich zu bestimmten Befehlen, die nicht unterstützt werden.

Themen

- Unterstützte Befehle, die mit einem Repository interagieren
- Unterstützte clientseitige Befehle

Unterstützte Befehle, die mit einem Repository interagieren

In diesem Abschnitt sind `pip` Befehle aufgeführt, bei denen der `pip` Client eine oder mehrere Anfragen an die Registry stellt, mit der er konfiguriert wurde. Es wurde überprüft, ob diese Befehle korrekt funktionieren, wenn sie in einem CodeArtifact Repository aufgerufen werden.

Befehl	Beschreibung
install	Pakete installieren.
herunterladen	Pakete herunterladen.

CodeArtifact implementiert nicht`pip search`. Wenn Sie `pip` mit einem CodeArtifact Repository konfiguriert haben, `pip search` werden beim Ausführen Pakete von PyPI gesucht und angezeigt.

Unterstützte clientseitige Befehle

Diese Befehle erfordern keine direkte Interaktion mit einem Repository und müssen daher auch CodeArtifact nichts tun, um sie zu unterstützen.

Befehl	Beschreibung
deinstallieren	Pakete deinstallieren.
einfrieren	Gibt installierte Pakete im Anforderungsformat aus.
Liste	Listet die installierten Pakete auf.
show	Zeigt Informationen über installierte Pakete an.
überprüfen	Stellen Sie sicher, dass die installierten Pakete kompatible Abhängigkeiten haben.
config	Verwalten Sie die lokale und globale Konfiguration.
Rad	Baue Räder nach deinen Anforderungen.
Hash	Berechnet die Hashes von Paketarchiven.
Fertigstellung	Hilft bei der Befehlsvervollständigung.
debug	Zeigt Informationen an, die für das Debuggen nützlich sind.
help	Zeigt Hilfe für Befehle an.

Python-Pakete von Upstreams und externen Verbindungen anfordern

Wenn Sie eine Python-Paketversion von pypi.org importieren, CodeArtifact werden alle Assets in dieser Paketversion importiert. Während die meisten Python-Pakete eine kleine Anzahl von

Assets enthalten, enthalten einige über 100 Assets, typischerweise zur Unterstützung mehrerer Hardwarearchitekturen und Python-Interpreter.

Es ist üblich, dass neue Assets für eine bestehende Paketversion auf pypi.org veröffentlicht werden. Beispielsweise veröffentlichen einige Projekte neue Assets, wenn neue Versionen von Python veröffentlicht werden. Wenn ein Python-Paket von CodeArtifact with installiert wird`pip install`, werden die im CodeArtifact Repository gespeicherten Paketversionen aktualisiert, um den neuesten Satz von Ressourcen von pypi.org widerzuspiegeln.

In ähnlicher Weise werden neue Elemente, die für eine Paketversion in einem CodeArtifact Upstream-Repository verfügbar sind, aber im aktuellen CodeArtifact Repository nicht vorhanden sind, bei `pip install` der Ausführung im aktuellen Repository beibehalten.

Paketversionen wurden entfernt

Einige Paketversionen in pypi.org sind als Yanked markiert, was dem Paketinstallationsprogramm (wie pip) mitteilt, dass die Version nicht installiert werden sollte, es sei denn, sie ist die einzige, die einem Versionsbezeichner entspricht (entweder mit oder). == === Weitere Informationen finden Sie unter PEP_592.

Wenn eine Paketversion in ursprünglich von einer externen Verbindung zu pypi.org abgerufen CodeArtifact wurde, CodeArtifact stellt Sie bei der Installation der Paketversion aus einem CodeArtifact Repository sicher, dass die aktualisierten Yankeed-Metadaten der Paketversion von pypi.org abgerufen werden.

Woher weiß ich, ob eine Paketversion gelöscht wurde

Um zu überprüfen, ob eine Paketversion eingebunden ist CodeArtifact, können Sie versuchen, sie mit zu installieren. `pip install` *`packageName`*`===`*`packageVersion`* Wenn die Paketversion gesperrt ist, erhalten Sie eine Warnmeldung, die der folgenden ähnelt:

```
WARNING: The candidate selected for download or install is a yanked version
```

Um zu überprüfen, ob eine Paketversion in pypi.org entfernt wurde, können Sie die pypi.org-Liste der Paketversion aufrufen. `https://pypi.org/project/`*`packageName`*`/`*`packageVersion`*`/`

Den Sperrstatus für private Pakete festlegen

CodeArtifact unterstützt nicht das Setzen von Yankeed-Metadaten für Pakete, die direkt in Repositorys veröffentlicht wurden. CodeArtifact

Warum wird das Abrufen der neuesten Yankeed-Metadaten oder Assets für eine Paketversion CodeArtifact nicht unterstützt?

CodeArtifact Stellt normalerweise sicher, dass beim Abrufen einer Python-Paketversion aus einem CodeArtifact Repository die abgerufenen Metadaten den neuesten Wert auf pypi.org haben. up-to-date Darüber hinaus wird die Liste der Assets in der Paketversion mit dem neuesten Satz auf pypi.org und allen Upstream-Repositorys aktualisiert. CodeArtifact Dies gilt unabhängig davon, ob Sie die Paketversion zum ersten Mal installieren und sie von pypi.org in Ihr CodeArtifact Repository CodeArtifact importieren, oder ob Sie das Paket schon einmal installiert haben. Es gibt jedoch Fälle, in denen der Paketmanager-Client, wie z. B. pip, die neuesten abgerufenen Metadaten nicht von pypi.org oder Upstream-Repositorys abruft. Stattdessen CodeArtifact werden die Daten zurückgegeben, die bereits in Ihrem Repository gespeichert sind. In diesem Abschnitt werden die drei Möglichkeiten beschrieben, wie dies geschehen kann:

Upstream-Konfiguration: Wenn die externe Verbindung zu pypi.org aus dem Repository oder seinen Upstreams entfernt wird disassociate-external-connection, werden die abgerufenen Metadaten nicht mehr von pypi.org aktualisiert. Ebenso sind beim Entfernen eines Upstream-Repositorys die Assets aus dem entfernten Repository und den Upstreams des entfernten Repositorys für das aktuelle Repository nicht mehr verfügbar. Das Gleiche gilt, wenn Sie die Kontrolle über die Herkunft von CodeArtifact Paketen verwenden, um zu verhindern, dass neue Versionen eines bestimmten Pakets abgerufen werden. Durch diese Einstellung `upstream=BLOCK` wird verhindert, dass gelöschte Metadaten aktualisiert werden.

Paketversionsstatus: Wenn Sie den Status einer Paketversion auf etwas anderes als `Published` oder setzen`Unlisted`, werden gelöschte Metadaten und Assets der Paketversion nicht aktualisiert. Ähnlich verhält es sich, wenn Sie eine bestimmte Paketversion abrufen (sagen wir`torch 2.0.1`) und dieselbe Paketversion in einem Upstream-Repository mit einem Status vorhanden ist, der nicht `Published` oder ist`Unlisted`, blockiert dies auch die Übertragung von Metadaten und die Weitergabe von Inhalten aus dem Upstream-Repository zum aktuellen Repository. Das liegt daran, dass andere Paketversionsstatus ein Hinweis darauf sind, dass die Versionen in keinem Repository mehr verwendet werden sollen.

Direktes Veröffentlichen: Wenn Sie eine bestimmte Paketversion direkt in einem CodeArtifact Repository veröffentlichen, wird dadurch verhindert, dass Metadaten entfernt und die Inhalte der Paketversion aus den Upstream-Repositorys und pypi.org aktualisiert werden. Nehmen wir zum Beispiel an, Sie laden ein Asset aus der Paketversion herunter`torch 2.0.1`, z. B. mit einem Webbrowser`torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`, und veröffentlichen es dann mit Twine as in Ihrem CodeArtifact Repository. `torch 2.0.1` CodeArtifact verfolgt,

dass die Paketversion durch direkte Veröffentlichung in Ihrem Repository in die Domain gelangt ist, nicht über eine externe Verbindung zu pypi.org oder einem Upstream-Repository. In diesem Fall werden die abgerufenen Metadaten CodeArtifact nicht mit den Upstream-Repositorys oder pypi.org synchronisiert. Das Gleiche gilt, wenn Sie `torch 2.0.1` in einem Upstream-Repository veröffentlichen — das Vorhandensein der Paketversion verhindert die Weitergabe von Metadaten und Inhalten an Repositorys weiter unten im Upstream-Diagramm.

CodeArtifact Mit Ruby verwenden

In diesen Themen wird beschrieben, wie Sie die Tools RubyGems und Bundler mit verwenden, um CodeArtifact Ruby-Gems zu installieren und zu veröffentlichen.

> **Note**
>
> CodeArtifact empfiehlt Ruby 3.3 oder höher und funktioniert nicht mit Ruby 2.6 oder älter.

Themen

- Konfiguration und Verwendung von RubyGems und Bundler mit CodeArtifact
- RubyGems Befehlsunterstützung
- Bundler-Kompatibilität

Konfiguration und Verwendung von RubyGems und Bundler mit CodeArtifact

Nachdem Sie ein Repository in erstellt haben CodeArtifact, können Sie RubyGems (`gem`) und Bundler (`bundle`) verwenden, um Gems zu installieren und zu veröffentlichen. In diesem Thema wird beschrieben, wie Sie die Paketmanager so konfigurieren, dass sie sich bei einem CodeArtifact Repository authentifizieren und dieses verwenden.

Configure RubyGems (**`gem`**) und Bundler (**`bundle`**) mit CodeArtifact

Um RubyGems (`gem`) oder Bundler (`bundle`) zu verwenden, um Gems zu veröffentlichen oder von dort Gems zu konsumieren AWS CodeArtifact, müssen Sie sie zunächst mit Ihren CodeArtifact Repository-Informationen konfigurieren, einschließlich der Zugangsdaten für den Zugriff darauf. Folgen Sie den Schritten in einem der folgenden Verfahren, um die Tools `gem` und `bundle` CLI mit Ihren CodeArtifact Repository-Endpunktinformationen und Anmeldeinformationen zu konfigurieren.

Configure RubyGems and Bundler mithilfe der Konsolenanweisungen

Sie können die Konfigurationsanweisungen in der Konsole verwenden, um Ihre Ruby-Paketmanager mit Ihrem CodeArtifact Repository zu verbinden. Die Konsolenanweisungen enthalten

benutzerdefinierte Befehle, die Sie ausführen können, um Ihre Paketmanager einzurichten, ohne Ihre CodeArtifact Informationen suchen und eingeben zu müssen.

1. Öffnen Sie die AWS CodeArtifact Konsole unter https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Wählen Sie im Navigationsbereich Repositories und dann das Repository aus, das Sie für die Installation oder Übertragung von Ruby Gems verwenden möchten.
3. Wählen Sie Verbindungsanweisungen anzeigen.
4. Wählen Sie Ihr Betriebssystem.
5. Wählen Sie den Ruby-Paketmanager-Client aus, den Sie mit Ihrem CodeArtifact Repository konfigurieren möchten.
6. Folgen Sie den generierten Anweisungen, um den Package Manager-Client so zu konfigurieren, dass er Ruby-Gems aus dem Repository installiert oder Ruby-Gems im Repository veröffentlicht.

Manuelles Konfigurieren RubyGems und Bündeln

Wenn Sie die Konfigurationsanweisungen von der Konsole aus nicht verwenden können oder wollen, können Sie die folgenden Anweisungen verwenden, um manuell eine Verbindung zwischen Ihren Ruby-Paketmanagern und Ihrem CodeArtifact Repository herzustellen.

1. Verwenden Sie in einer Befehlszeile den folgenden Befehl, um ein CodeArtifact Autorisierungstoken abzurufen und es in einer Umgebungsvariablen zu speichern.

   - Ersetzen Sie es *my_domain* durch Ihren CodeArtifact Domainnamen.
   - *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

   macOS and Linux

   ```
   export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

   Windows

   - Windows (unter Verwendung der Standard-Befehlsshell):

```
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
```

- Windows PowerShell:

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --
output text
```

2. Um Ruby-Gems in Ihrem Repository zu veröffentlichen, verwenden Sie den folgenden Befehl, um den Endpunkt Ihres CodeArtifact Repositorys abzurufen und ihn in der `RUBYGEMS_HOST` Umgebungsvariablen zu speichern. Die `gem` CLI verwendet diese Umgebungsvariable, um zu bestimmen, wo Gems veröffentlicht werden.

> **Note**
>
> Anstatt die `RUBYGEMS_HOST` Umgebungsvariable zu verwenden, können Sie dem Repository-Endpunkt bei der Verwendung des `gem push` Befehls `--host` diese Option zur Verfügung stellen.

- Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
- *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
- Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

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

Die folgenden Befehle rufen den Repository-Endpunkt ab, kürzen die nachfolgenden / Befehle und speichern sie dann in einer Umgebungsvariablen.

- Windows (unter Verwendung der Standard-Befehlsshell):

```
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%
```

- Windows 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("/")
```

Die folgende URL ist ein Beispiel für einen Repository-Endpunkt:

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/
```

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den `codeartifact.region.on.aws` Endpunkt.

3. Um Ruby-Gems in Ihrem Repository zu veröffentlichen, müssen Sie sich CodeArtifact bei authentifizieren, RubyGems indem Sie Ihre `~/.gem/credentials` Datei so bearbeiten, dass sie Ihr Authentifizierungstoken enthält. Erstellen Sie ein `~/.gem/` Verzeichnis und eine `~/.gem/credentials` Datei, falls das Verzeichnis oder die Datei nicht existiert.

   macOS and Linux

   ```
   echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/
   credentials
   ```

Windows

- Windows (unter Verwendung der Standard-Befehlsshell):

```
echo :codeartifact_api_key: Bearer %CODEARTIFACT_AUTH_TOKEN% >> %USERPROFILE
%/.gem/credentials
```

- Windows PowerShell:

```
echo ":codeartifact_api_key: Bearer $env:CODEARTIFACT_AUTH_TOKEN" | Add-
Content ~/.gem/credentials
```

4. Um Ruby Gems aus Ihrem Repository gem zu installieren, müssen Sie Ihrer `.gemrc` Datei die Repository-Endpunktinformationen und das Authentifizierungstoken hinzufügen. Sie können es der Globaldatei (`~/.gemrc`) oder Ihrer `.gemrc` Projektdatei hinzufügen. Die CodeArtifact Informationen, die Sie hinzufügen müssen, sind `.gemrc` eine Kombination aus dem Repository-Endpunkt und dem Authentifizierungstoken. Sie ist wie folgt formatiert:

```
https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/
```

- Für das Authentifizierungstoken können Sie die `CODEARTIFACT_AUTH_TOKEN` Umgebungsvariable verwenden, die in einem früheren Schritt festgelegt wurde.
- Um den Repository-Endpunkt abzurufen, können Sie den Wert der `RUBYGEMS_HOST` Umgebungsvariablen lesen, die zuvor festgelegt wurde, oder Sie können den folgenden `get-repository-endpoint` Befehl verwenden und die Werte nach Bedarf ersetzen:

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint
 --output text
```

Wenn Sie den Endpunkt gefunden haben, fügen Sie ihn mit einem Texteditor `aws:${CODEARTIFACT_AUTH_TOKEN}@` an der entsprechenden Position hinzu. Sobald Sie den Repository-Endpunkt und die Authentifizierungstoken-Zeichenfolge erstellt haben, fügen Sie sie mit dem folgenden `echo` Befehl zum `:sources:` Abschnitt Ihrer `.gemrc` Datei hinzu:

> **Warning**
>
> CodeArtifact unterstützt das Hinzufügen von Repositorys als Quellen mit dem `gem sources -add` Befehl nicht. Sie müssen die Quelle direkt zur Datei hinzufügen.

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 (unter Verwendung der Standard-Befehlsshell):

```
echo ":sources:
    - https://aws:%CODEARTIFACT_AUTH_TOKEN
%@my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/"
 > "%USERPROFILE%\.gemrc"
```

- Windows 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
```

5. Um Bundler zu verwenden, müssen Sie Bundler mit Ihrer Repository-Endpunkt-URL und Ihrem Authentifizierungstoken konfigurieren, indem Sie den folgenden `bundle config` Befehl ausführen:

   macOS and Linux

   ```
   bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
   ```

   Windows

   - Windows (unter Verwendung der Standard-Befehlsshell):

```
bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
```

- Windows PowerShell:

```
bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
```

Nachdem Sie nun RubyGems (`gem`) und Bundler (`bundle`) mit Ihrem CodeArtifact Repository konfiguriert haben, können Sie sie verwenden, um Ruby-Gems zu veröffentlichen und daraus zu nutzen.

Ruby-Gems installieren von CodeArtifact

Verwenden Sie die folgenden Verfahren, um Ruby Gems aus einem CodeArtifact Repository mit den Tools `gem` oder `bundle` CLI zu installieren.

Installieren Sie Ruby Gems mit **`gem`**

Sie können die CLI RubyGems (`gem`) verwenden, um schnell eine bestimmte Version eines Ruby-Gems aus Ihrem CodeArtifact Repository zu installieren.

Um Ruby-Gems aus einem CodeArtifact Repository zu installieren mit **`gem`**

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie die `gem` CLI so, dass sie Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Configure RubyGems (`gem`) und Bundler (`bundle`) mit CodeArtifact

   > **Note**
   >
   > Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Verwenden Sie den folgenden Befehl, um Ruby Gems von zu installieren CodeArtifact:

   ```
   gem install my_ruby_gem --version 1.0.0
   ```

Installieren Sie Ruby Gems mit **`bundle`**

Sie können die Bundler (`bundle`) CLI verwenden, um die Ruby-Gems zu installieren, die in Ihrem `Gemfile` konfiguriert sind.

Um Ruby-Gems aus einem CodeArtifact Repository zu installieren mit **`bundle`**

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie die `bundle` CLI so, dass sie Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Configure RubyGems (`gem`) und Bundler (`bundle`) mit CodeArtifact

   > **Note**
   >
   > Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Fügen Sie Ihre CodeArtifact Repository-Endpunkt-URL zu Ihrem `Gemfile` AS hinzu`source`, um konfigurierte Ruby-Gems aus Ihrem CodeArtifact Repository und seinen Upstreams zu installieren.

   ```
   source "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   ruby/my_repo/"

   gem 'my_ruby_gem'
   ```

3. Verwenden Sie den folgenden Befehl, um die Ruby-Gems wie in Ihrem `Gemfile` angegeben zu installieren:

   ```
   bundle install
   ```

Veröffentlichen von Ruby-Gems auf CodeArtifact

Gehen Sie wie folgt vor, um Ruby-Gems mithilfe der `gem` CLI in einem CodeArtifact Repository zu veröffentlichen.

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie die `gem` CLI so, dass sie Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Configure RubyGems (`gem`) und Bundler (`bundle`) mit CodeArtifact

> **Note**
>
> Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Verwenden Sie den folgenden Befehl, um Ruby-Gems in einem CodeArtifact Repository zu veröffentlichen. Beachten Sie, dass Sie, wenn Sie die `RUBYGEMS_HOST` Umgebungsvariable nicht festgelegt haben, Ihren CodeArtifact Repository-Endpunkt in der `--host` Option angeben müssen.

```
gem push --key codeartifact_api_key my_ruby_gem-0.0.1.gem
```

RubyGems Befehlsunterstützung

CodeArtifact unterstützt die `gem push` Befehle `gem install` und. CodeArtifact unterstützt die folgenden `gem` Befehle nicht:

- `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`

Bundler-Kompatibilität

Dieses Handbuch enthält Informationen CodeArtifact zur Kompatibilität mit Bundler.

Bundler-Kompatibilität

AWS CodeArtifact empfiehlt Bundler 2.4.11 oder höher. Wenn Sie Probleme bei der Installation haben, aktualisieren Sie die Bundler-CLI auf die neueste Version.

Unterstützung für Bundler-Versionen

In Bundler-Versionen vor 2.4.11 gibt es ein Limit von 500 Abhängigkeiten, die im Gemfile definiert werden können, bevor Bundler beschließt, den vollständigen Index abzufragen. `specs.4.8.gz` Da der vollständige Index CodeArtifact nicht unterstützt wird, funktioniert die Angabe von mehr als 500 Abhängigkeiten nicht, CodeArtifact wenn Bundler-Versionen unter 2.4.11 verwendet werden.

Um mehr als 500 Abhängigkeiten in Ihrem Gemfile mit zu definieren CodeArtifact, aktualisieren Sie Bundler auf Version 2.4.11 oder höher.

Unterstützung für Bundler-Operationen

CodeArtifactDie Unterstützung für RubyGems beinhaltet nicht den Bundler Compact Index APIs (die `/versions` API wird nicht unterstützt). CodeArtifact unterstützt nur die Dependencies-API.

CodeArtifact Unterstützt außerdem nicht die verschiedenen Spezifikationen APIs, wie `specs.4.8.gz` z.

Verwendung CodeArtifact mit Swift

In diesen Themen wird beschrieben, wie Sie den Swift Package Manager verwenden CodeArtifact , um Swift-Pakete zu installieren und zu veröffentlichen.

> **Note**
>
> CodeArtifact unterstützt Swift 5.8 und höher sowie Xcode 14.3 und höher. CodeArtifact empfiehlt Swift 5.9 und höher und Xcode 15 und höher.

Themen

- Konfigurieren Sie den Swift Package Manager mit CodeArtifact
- Swift-Pakete konsumieren und veröffentlichen
- Schnelle Normalisierung von Paketnamen und Namespace
- Schnelle Problembehebung

Konfigurieren Sie den Swift Package Manager mit CodeArtifact

Um den Swift Package Manager zum Veröffentlichen von Paketen oder zum Konsumieren von Paketen zu verwenden AWS CodeArtifact, müssen Sie zunächst Anmeldeinformationen für den Zugriff auf Ihr CodeArtifact Repository einrichten. Die empfohlene Methode zur Konfiguration der Swift Package Manager-CLI mit Ihren CodeArtifact Anmeldeinformationen und Ihrem Repository-Endpunkt ist die Verwendung des `aws codeartifact login` Befehls. Sie können den Swift Package Manager auch manuell konfigurieren.

Konfigurieren Sie Swift mit dem Login-Befehl

Verwenden Sie den `aws codeartifact login` Befehl, um den Swift Package Manager mit zu konfigurieren CodeArtifact.

> **Note**
>
> Um den Login-Befehl zu verwenden, ist Swift 5.8 oder höher erforderlich und Swift 5.9 oder höher wird empfohlen.

Der `aws codeartifact login` Befehl führt Folgendes aus:

1. Rufen Sie ein Authentifizierungstoken von Ihrer Umgebung ab CodeArtifact und speichern Sie es dort. Wie die Anmeldeinformationen gespeichert werden, hängt vom Betriebssystem der Umgebung ab:

   a. macOS: In der macOS Keychain App wird ein Eintrag erstellt.

   b. Linux und Windows: In der `~/.netrc` Datei wird ein Eintrag erstellt.

   Wenn in allen Betriebssystemen ein Eintrag für Anmeldeinformationen vorhanden ist, ersetzt dieser Befehl diesen Eintrag durch ein neues Token.

2. Rufen Sie die URL Ihres CodeArtifact Repository-Endpunkts ab und fügen Sie sie Ihrer Swift-Konfigurationsdatei hinzu. Der Befehl fügt die Repository-Endpunkt-URL zur Konfigurationsdatei auf Projektebene hinzu, die sich unter `/path/to/project/.swiftpm/configuration/registries.json` befindet.

> **Note**
>
> Der `aws codeartifact login` Befehl ruft `swift package-registry` Befehle auf, die von dem Verzeichnis aus ausgeführt werden müssen, das die `Package.swift` Datei enthält. Aus diesem Grund muss der `aws codeartifact login` Befehl innerhalb des Swift-Projekts ausgeführt werden.

Um Swift mit dem Login-Befehl zu konfigurieren

1. Navigieren Sie zum Swift-Projektverzeichnis, das die `Package.swift` Datei Ihres Projekts enthält.
2. Führen Sie den Befehl `aws codeartifact login` aus.

   Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie dies nicht angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

```
aws codeartifact login --tool swift --domain my_domain \
--domain-owner 111122223333 --repository my_repo \
[--namespace my_namespace]
```

Die `--namespace` Option konfiguriert die Anwendung so, dass sie nur Pakete aus Ihrem CodeArtifact Repository verwendet, wenn sie sich im angegebenen Namespace befinden. CodeArtifact Namespaces sind ein Synonym für Bereiche und werden verwendet, um Code in logische Gruppen zu organisieren und Namenskollisionen zu verhindern, die auftreten können, wenn Ihre Codebasis mehrere Bibliotheken umfasst.

Der Standardzeitraum für die Autorisierung nach dem Aufrufen `login` beträgt 12 Stunden und `login` muss aufgerufen werden, um das Token regelmäßig zu aktualisieren. Weitere Hinweise zu dem mit dem `login` Befehl erstellten Autorisierungstoken finden Sie unterMit dem Befehl erstellte Tokens `login`.

Konfigurieren Sie Swift ohne den Login-Befehl

Es wird zwar empfohlen, Swift mit dem `aws codeartifact login` Befehl zu konfigurieren, Sie können den Swift Package Manager jedoch auch ohne den Login-Befehl konfigurieren, indem Sie die Swift Package Manager-Konfiguration manuell aktualisieren.

Im folgenden Verfahren verwenden Sie den, AWS CLI um Folgendes zu tun:

1. Rufen Sie ein Authentifizierungstoken von Ihrer Umgebung ab CodeArtifact und speichern Sie es dort. Wie die Anmeldeinformationen gespeichert werden, hängt vom Betriebssystem der Umgebung ab:
   a. macOS: In der macOS Keychain App wird ein Eintrag erstellt.
   b. Linux und Windows: In der `~/.netrc` Datei wird ein Eintrag erstellt.
2. Rufen Sie die URL Ihres CodeArtifact Repository-Endpunkts ab.
3. Fügen Sie in der `~/.swiftpm/configuration/registries.json` Konfigurationsdatei einen Eintrag mit Ihrer Repository-Endpunkt-URL und Ihrem Authentifizierungstyp hinzu.

Um den Swift ohne den Login-Befehl zu konfigurieren

1. Verwenden Sie in einer Befehlszeile den folgenden Befehl, um ein CodeArtifact Autorisierungstoken abzurufen und es in einer Umgebungsvariablen zu speichern.
   - Ersetzen Sie es *`my_domain`* durch Ihren CodeArtifact Domainnamen.
   - *`111122223333`*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

macOS and Linux

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

Windows

- Windows (unter Verwendung der Standard-Befehlsshell):

```
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
```

- Windows PowerShell:

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --
output text
```

2. Rufen Sie den Endpunkt Ihres CodeArtifact Repositorys ab, indem Sie den folgenden Befehl ausführen. Ihr Repository-Endpunkt wird verwendet, um den Swift Package Manager auf Ihr Repository zu verweisen, um Pakete zu konsumieren oder zu veröffentlichen.

   - Ersetze es *my_domain* durch deinen CodeArtifact Domainnamen.
   - *111122223333*Ersetzen Sie es durch die AWS Konto-ID des Inhabers der Domain. Wenn Sie auf ein Repository in einer Domain zugreifen, die Sie besitzen, müssen Sie nichts angeben`--domain-owner`. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
   - Ersetze es *my_repo* durch deinen CodeArtifact Repository-Namen.

   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 (unter Verwendung der Standard-Befehlsshell):

```
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
```

- Windows PowerShell:

```
$env:CODEARTIFACT_REPO = aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format
 swift --query repositoryEndpoint --output text
```

Die folgende URL ist ein Beispiel für einen Repository-Endpunkt.

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
swift/my_repo/
```

> **Note**
>
> Um einen Dual-Stack-Endpunkt zu verwenden, verwenden Sie den `codeartifact.region.on.aws` Endpunkt.

> **Important**
>
> Sie müssen `login` an das Ende des Repository-URL-Endpunkts anhängen, wenn Sie ihn zur Konfiguration des Swift Package Managers verwenden. Dies wird in den Befehlen dieses Verfahrens für Sie erledigt.

3. Wenn diese beiden Werte in Umgebungsvariablen gespeichert sind, übergeben Sie sie mit dem folgenden `swift package-registry login` Befehl an Swift:

macOS and Linux

```
swift package-registry login ${CODEARTIFACT_REPO}login --token
 ${CODEARTIFACT_AUTH_TOKEN}
```

Windows

- Windows (unter Verwendung der Standard-Befehlsshell):

```
swift package-registry login %CODEARTIFACT_REPO%login --token
 %CODEARTIFACT_AUTH_TOKEN%
```

- Windows PowerShell:

```
swift package-registry login $Env:CODEARTIFACT_REPO+"login" --token
 $Env:CODEARTIFACT_AUTH_TOKEN
```

4. Aktualisieren Sie als Nächstes die von Ihrer Anwendung verwendete Paketregistrierung, sodass alle Abhängigkeiten aus Ihrem CodeArtifact Repository abgerufen werden. Dieser Befehl muss in dem Projektverzeichnis ausgeführt werden, in dem Sie versuchen, die Paketabhängigkeit aufzulösen:

   macOS and Linux

   ```
   $ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
   ```

   Windows

   - Windows (unter Verwendung der Standard-Befehlsshell):

   ```
   $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
   ```

   - Windows PowerShell:

   ```
   $ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
   ```

   Die `--scope` Option konfiguriert die Anwendung so, dass sie nur Pakete aus Ihrem CodeArtifact Repository verwendet, wenn sie sich im angegebenen Bereich befinden. Bereiche sind

gleichbedeutend mit CodeArtifact Namespaces und werden verwendet, um Code in logische Gruppen zu organisieren und Namenskollisionen zu verhindern, die auftreten können, wenn Ihre Codebasis mehrere Bibliotheken umfasst.

5. Sie können überprüfen, ob die Konfiguration korrekt eingerichtet wurde, indem Sie den Inhalt der `.swiftpm/configuration/registries.json` Datei auf Projektebene anzeigen, indem Sie den folgenden Befehl in Ihrem Projektverzeichnis ausführen:

```
$ 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
}
```

Nachdem Sie den Swift Package Manager mit Ihrem CodeArtifact Repository konfiguriert haben, können Sie ihn verwenden, um Swift-Pakete zu veröffentlichen und zu verwenden. Weitere Informationen finden Sie unter Swift-Pakete konsumieren und veröffentlichen.

Swift-Pakete konsumieren und veröffentlichen

Swift-Pakete werden konsumiert von CodeArtifact

Gehen Sie wie folgt vor, um Swift-Pakete aus einem AWS CodeArtifact Repository zu konsumieren.

Um Swift-Pakete aus einem CodeArtifact Repository zu konsumieren

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie den Swift Package Manager so, dass er Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

> **Note**
>
> Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Bearbeiten Sie die `Package.swift` Datei in Ihrem Anwendungsprojektordner, um die Paketabhängigkeiten zu aktualisieren, die von Ihrem Projekt verwendet werden sollen.

   a. Wenn die `Package.swift` Datei keinen `dependencies` Abschnitt enthält, fügen Sie einen hinzu.

   b. Fügen Sie im `dependencies` Abschnitt der `Package.swift` Datei das Paket hinzu, das Sie verwenden möchten, indem Sie dessen Paket-ID hinzufügen. Die Paketkennung besteht aus dem Bereich und dem Paketnamen, die durch einen Punkt getrennt sind. Ein Beispiel finden Sie im Codeausschnitt, der einem späteren Schritt folgt.

   > **Tip**
   >
   > Um die Paket-ID zu finden, können Sie die CodeArtifact Konsole verwenden. Suchen Sie die spezifische Paketversion, die Sie verwenden möchten, und lesen Sie auf der Seite mit der Paketversion nach den Anweisungen für die Kurzanleitung zur Installation.

   c. Wenn die `Package.swift` Datei keinen `targets` Abschnitt enthält, fügen Sie einen hinzu.

   d. Fügen Sie im `targets` Abschnitt die Ziele hinzu, die die Abhängigkeit verwenden müssen.

   Der folgende Ausschnitt ist ein Beispielausschnitt, der konfigurierte `targets` Abschnitte `dependencies` und Abschnitte in einer Datei zeigt: `Package.swift`

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
```

```
    ],
...
```

3. Nachdem alles konfiguriert ist, verwenden Sie den folgenden Befehl, um die Paketabhängigkeiten von herunterzuladen. CodeArtifact

```
swift package resolve
```

Swift-Pakete CodeArtifact in Xcode konsumieren

Gehen Sie wie folgt vor, um Swift-Pakete aus einem CodeArtifact Repository in Xcode zu konsumieren.

Um Swift-Pakete aus einem CodeArtifact Repository in Xcode zu konsumieren

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie den Swift Package Manager so, dass er Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

> **Note**
>
> Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Fügen Sie die Pakete als Abhängigkeit zu Ihrem Projekt in Xcode hinzu.

   a. Wählen Sie „Datei“ > „Pakete hinzufügen“.

   b. Suchen Sie mithilfe der Suchleiste nach Ihrem Paket. Ihre Suche muss im Formular erfolgen`package_scope.package_name`.

   c. Sobald Sie es gefunden haben, wählen Sie das Package aus und wählen Sie Paket hinzufügen.

   d. Sobald das Package verifiziert ist, wählen Sie die Paketprodukte aus, die Sie als Abhängigkeit hinzufügen möchten, und wählen Sie Paket hinzufügen.

Wenn Sie Probleme bei der Verwendung Ihres CodeArtifact Repositorys mit Xcode haben, finden Sie Informationen Schnelle Problembehebung zu häufig auftretenden Problemen und möglichen Lösungen.

Swift-Pakete veröffentlichen auf CodeArtifact

CodeArtifact empfiehlt Swift 5.9 oder höher und die Verwendung des `swift package-registry publish` Befehls zur Veröffentlichung von Swift-Paketen. Wenn Sie eine frühere Version verwenden, müssen Sie einen Curl-Befehl verwenden, um Swift-Pakete zu veröffentlichen. CodeArtifact

Veröffentlichen von CodeArtifact Paketen mit dem Befehl **`swift package-registry publish`**

Verwenden Sie das folgende Verfahren mit Swift 5.9 oder höher, um Swift-Pakete mit dem Swift Package Manager in einem CodeArtifact Repository zu veröffentlichen.

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie den Swift Package Manager so, dass er Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

   > **Note**
   >
   > Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung 12 Stunden vergangen sind.

2. Navigieren Sie zum Swift-Projektverzeichnis, das die `Package.swift` Datei für Ihr Paket enthält.
3. Führen Sie den folgenden `swift package-registry publish` Befehl aus, um das Paket zu veröffentlichen. Der Befehl erstellt ein Paketquellarchiv und veröffentlicht es in Ihrem CodeArtifact Repository.

   ```
   swift package-registry publish packageScope.packageName packageVersion
   ```

   Zum Beispiel:

   ```
   swift package-registry publish myScope.myPackage 1.0.0
   ```

4. Sie können überprüfen, ob das Paket veröffentlicht wurde und im Repository vorhanden ist, indem Sie in der Konsole nachschauen oder den `aws codeartifact list-packages` Befehl wie folgt verwenden:

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

Sie können die einzelne Version des Pakets mit dem folgenden `aws codeartifact list-package-versions` Befehl auflisten:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

CodeArtifact Pakete mit Curl veröffentlichen

Es wird zwar empfohlen, den `swift package-registry publish` Befehl zu verwenden, der in Swift 5.9 oder höher enthalten ist, aber Sie können Curl auch verwenden, um Swift-Pakete zu veröffentlichen. CodeArtifact

Gehen Sie wie folgt vor, um Swift-Pakete mit Curl in einem AWS CodeArtifact Repository zu veröffentlichen.

1. Falls nicht, erstellen und aktualisieren Sie die `CODEARTIFACT_REPO` Umgebungsvariablen `CODEARTIFACT_AUTH_TOKEN` und, indem Sie die Schritte unter befolgen. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

   > **Note**
   >
   > Das Autorisierungs-Token ist 12 Stunden gültig. Sie müssen Ihre `CODEARTIFACT_AUTH_TOKEN` Umgebungsvariable mit neuen Anmeldeinformationen aktualisieren, wenn seit ihrer Erstellung 12 Stunden vergangen sind.

2. Wenn Sie kein Swift-Paket erstellt haben, können Sie dies zunächst tun, indem Sie die folgenden Befehle ausführen:

   ```
   mkdir testDir && cd testDir
   swift package init
   git init .
   swift package archive-source
   ```

3. Verwenden Sie den folgenden Curl-Befehl, um Ihr Swift-Paket zu CodeArtifact veröffentlichen in:

   macOS and Linux

   ```
   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"
```

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. Sie können überprüfen, ob das Paket veröffentlicht wurde und im Repository vorhanden ist, indem Sie in der Konsole nachschauen oder den `aws codeartifact list-packages` Befehl wie folgt verwenden:

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

Sie können die einzelne Version des Pakets mit dem folgenden `aws codeartifact list-package-versions` Befehl auflisten:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Swift-Pakete abrufen von GitHub und erneutes Veröffentlichen von CodeArtifact

Gehen Sie wie folgt vor, um ein Swift-Paket aus einem Repository abzurufen GitHub und es erneut in einem Repository zu veröffentlichen. CodeArtifact

Um ein Swift-Paket abzurufen GitHub und es erneut zu veröffentlichen CodeArtifact

1. Falls nicht, folgen Sie den Schritten unter So konfigurieren Sie den Swift Package Manager so, dass er Ihr CodeArtifact Repository mit den richtigen Anmeldeinformationen verwendet. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

> **Note**
>
> Das generierte Autorisierungstoken ist 12 Stunden gültig. Sie müssen ein neues erstellen, wenn seit der Erstellung eines Tokens 12 Stunden vergangen sind.

2. Klonen Sie das Git-Repository des Swift-Pakets, das Sie abrufen und erneut veröffentlichen möchten, indem Sie den folgenden `git clone` Befehl verwenden. Informationen zum Klonen von GitHub Repositorys finden Sie in der Dokumentation unter Ein Repository klonen. GitHub

```
git clone repoURL
```

3. Navigieren Sie zu dem Repository, das Sie gerade geklont haben:

```
cd repoName
```

4. Erstellen Sie das Paket und veröffentlichen Sie es in CodeArtifact.

   a. Empfehlung: Wenn Sie Swift 5.9 oder höher verwenden, können Sie den folgenden `swift package-registry publish` Befehl verwenden, um das Paket zu erstellen und es in Ihrem konfigurierten CodeArtifact Repository zu veröffentlichen.

   ```
   swift package-registry publish packageScope.packageName versionNumber
   ```

   Zum Beispiel:

   ```
   swift package-registry publish myScope.myPackage 1.0.0
   ```

   b. Wenn Sie eine Swift-Version verwenden, die älter als 5.9 ist, müssen Sie den `swift archive-source` Befehl verwenden, um das Paket zu erstellen, und dann einen Curl-Befehl verwenden, um es zu veröffentlichen.

      i. Wenn Sie die Variablen `CODEARTIFACT_AUTH_TOKEN` und die `CODEARTIFACT_REPO` Umgebungsvariablen noch nicht konfiguriert haben oder es mehr als 12 Stunden her ist, dass Sie das getan haben, folgen Sie den Schritten unter. Konfigurieren Sie Swift ohne den Login-Befehl

      ii. Erstellen Sie das Swift-Paket mit dem `swift package archive-source` folgenden Befehl:

```
swift package archive-source
```

Wenn dies erfolgreich ist, werden Sie es `Created` *package_name*`.zip` in der Befehlszeile sehen.

iii. Verwenden Sie den folgenden Curl-Befehl, um das Swift-Paket zu CodeArtifact veröffentlichen in:

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. Sie können überprüfen, ob das Paket veröffentlicht wurde und im Repository vorhanden ist, indem Sie in der Konsole nachschauen oder den `aws codeartifact list-packages` Befehl wie folgt verwenden:

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

Sie können die einzelne Version des Pakets mit dem folgenden `aws codeartifact list-package-versions` Befehl auflisten:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Schnelle Normalisierung von Paketnamen und Namespace

CodeArtifact normalisiert Paketnamen und Namespaces, bevor sie gespeichert werden, was bedeutet, dass sich die Namen darin von denen unterscheiden CodeArtifact können, die bei der Veröffentlichung des Pakets angegeben wurden.

Normalisierung von Paketnamen und Namespaces: CodeArtifact normalisiert Swift-Paketnamen und Namespaces, indem alle Buchstaben in Kleinbuchstaben umgewandelt werden.

Normalisierung der Paketversion: CodeArtifact Normalisiert keine Swift-Paketversionen. Beachten Sie, dass CodeArtifact nur Semantic Versioning 2.0-Versionsmuster unterstützt werden. Weitere Informationen zur Semantic Versioning finden Sie unter Semantic Versioning 2.0.0.

Der nicht normalisierte Paketname und der Namespace können mit API- und CLI-Anfragen verwendet werden, da CodeArtifact die Eingaben für diese Anfragen normalisiert werden. Zum Beispiel `--namespace myScope` würden Eingaben von `--package myPackage` und normalisiert werden und ein Paket zurückgeben, das den normalisierten Paketnamen und den Namespace von hat. `mypackage` `myscope`

Sie müssen normalisierte Namen in verwenden ARNs, z. B. in IAM-Richtlinien.

Verwenden Sie den Befehl, um den normalisierten Namen eines Pakets zu ermitteln. `aws codeartifact list-packages` Weitere Informationen finden Sie unter Listet Paketnamen auf.

Schnelle Problembehebung

Die folgenden Informationen können Ihnen bei der Behebung häufiger Probleme mit Swift und helfen CodeArtifact.

Ich erhalte einen 401-Fehler in Xcode, auch nachdem ich den Swift Package Manager konfiguriert habe

Problem: Wenn Sie versuchen, ein Paket aus Ihrem CodeArtifact Repository als Abhängigkeit zu Ihrem Swift-Projekt in Xcode hinzuzufügen, erhalten Sie einen unautorisierten 401-Fehler, auch wenn Sie die Anweisungen zum Herstellen einer Verbindung mit Swift befolgt haben. CodeArtifact

Mögliche Lösungen: Dies kann durch ein Problem mit der macOS Keychain-App verursacht werden, in der Ihre CodeArtifact Anmeldeinformationen gespeichert sind. Um dies zu beheben, empfehlen wir, die Keychain-App zu öffnen und alle CodeArtifact Einträge zu löschen und den Swift Package

Manager erneut mit Ihrem CodeArtifact Repository zu konfigurieren, indem Sie den Anweisungen unter folgen. Konfigurieren Sie den Swift Package Manager mit CodeArtifact

Xcode hängt auf dem CI-Computer, da der Schlüsselbund zur Eingabe des Passworts aufgefordert wird

Problem: Wenn Sie versuchen, Swift-Pakete CodeArtifact als Teil eines Xcode-Builds auf einem Continuous Integration (CI) -Server abzurufen, z. B. mit GitHub Aktionen, CodeArtifact kann die Authentifizierung mit hängen bleiben und schließlich mit einer Fehlermeldung ähnlich der folgenden fehlschlagen:

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

Mögliche Lösungen: Dies wird dadurch verursacht, dass Anmeldeinformationen auf CI-Computern nicht im Schlüsselbund gespeichert werden und Xcode nur Anmeldeinformationen unterstützt, die im Schlüsselbund gespeichert sind. Um dieses Problem zu beheben, empfehlen wir, den Schlüsselbundeintrag mithilfe der folgenden Schritte manuell zu erstellen:

1. Bereite den Schlüsselbund vor.

```
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. Besorgen Sie sich ein CodeArtifact Authentifizierungstoken und Ihren Repository-Endpunkt.

```
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. Erstellen Sie den Schlüsselbundeintrag manuell.

```
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 $SERVER -r htps "${KEYCHAIN_NAME}"

security add-internet-password -a token \
                            -s $SERVER \
                            -w $CODEARTIFACT_AUTH_TOKEN \
                            -r htps \
                            -U \
                            "${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}"
```

Weitere Informationen zu diesem Fehler und der Lösung finden Sie unter https://github.com/apple/swift-package-manager/issues/7236.

Verwendung CodeArtifact mit generischen Paketen

In diesen Themen erfahren Sie, wie Sie generische Pakete verwenden und für sie veröffentlichen AWS CodeArtifact.

Themen

- Überblick über generische Pakete
- Unterstützte Befehle für generische Pakete
- Generische Pakete veröffentlichen und konsumieren

Überblick über generische Pakete

Mithilfe des `generic` Paketformats können Sie jeden beliebigen Dateityp hochladen, um ein Paket in einem CodeArtifact Repository zu erstellen. Generische Pakete sind keiner bestimmten Programmiersprache, keinem bestimmten Dateityp oder einem bestimmten Paketmanagement-Ökosystem zugeordnet. Dies kann nützlich sein, um beliebige Build-Artefakte wie Anwendungsinstallationsprogramme, Modelle für maschinelles Lernen, Konfigurationsdateien und andere zu speichern und zu versionieren.

Ein generisches Paket besteht aus einem Paketnamen, einem Namespace, einer Version und einem oder mehreren Assets (oder Dateien). Generische Pakete können zusammen mit Paketen anderer Formate in einem einzigen CodeArtifact Repository existieren.

Sie können das AWS CLI oder SDK verwenden, um mit generischen Paketen zu arbeiten. Eine vollständige Liste der AWS CLI Befehle, die mit generischen Paketen funktionieren, finden Sie unterUnterstützte Befehle für generische Pakete.

Allgemeine Paketeinschränkungen

- Sie werden niemals aus Upstream-Repositorys abgerufen. Sie können nur von dem Repository abgerufen werden, in dem sie veröffentlicht wurden.
- Sie können keine Abhängigkeiten deklarieren, von denen zurückgegeben ListPackageVersionDependenciesoder in der angezeigt werden sollen AWS Management Console .
- Sie können README- und LICENSE-Dateien speichern, sie werden jedoch nicht von CodeArtifact interpretiert. Die Informationen in diesen Dateien werden nicht von

GetPackageVersionReadmeoder DescribePackageVersionzurückgegeben und erscheinen auch nicht in der AWS Management Console.

- Wie bei allen Paketen in CodeArtifact gibt es Beschränkungen für die Asset-Größe und die Anzahl der Assets pro Paket. Weitere Informationen zu Beschränkungen und Kontingenten CodeArtifact finden Sie unterKontingente in AWS CodeArtifact.
- Die darin enthaltenen Asset-Namen müssen den folgenden Regeln entsprechen:
  - Assetnamen können Unicode-Buchstaben und -Zahlen verwenden. Insbesondere sind die folgenden Unicode-Zeichenkategorien zulässig: Kleinbuchstabe (`Ll`), Modifikationsbuchstabe (`Lm`), Anderer Buchstabe (`Lo`), Titelbuchstabe (`Lt`), Großbuchstabe (), Buchstabennummer (`Lu`) und Dezimalzahl (`Nl`). `Nd`
  - Die folgenden Sonderzeichen sind zulässig:`~!@^&()-_+[]{};,.`
  - Vermögenswerte können nicht benannt werden oder `. ..`
  - Leerzeichen sind das einzig zulässige Leerzeichen. Elementnamen dürfen nicht mit einem Leerzeichen beginnen oder enden und auch keine aufeinanderfolgenden Leerzeichen enthalten.

Unterstützte Befehle für generische Pakete

Sie können das AWS CLI oder SDK verwenden, um mit generischen Paketen zu arbeiten. Die folgenden CodeArtifact Befehle funktionieren mit generischen Paketen:

- copy-package-versions(siehePakete zwischen Repositorys kopieren)
- Paket löschen (siehe) Löschen eines Pakets (AWS CLI)
- delete-package-versions(siehe) Löschen einer Paketversion (AWS CLI)
- Paket beschreiben
- describe-package-version(siehe) Details und Abhängigkeiten der Paketversion anzeigen und aktualisieren
- dispose-package-versions(siheEntsorgung von Paketversionen)
- get-package-version-asset(siheLaden Sie die Ressourcen der Paketversion herunter)
- list-package-version-assets(siheListet die Ressourcen der Paketversion auf)
- list-package-versions(siehePaketversionen auflisten)
- Pakete auflisten (siehe) Listet Paketnamen auf
- publish-package-version(siehe) Veröffentlichen eines generischen Pakets

- put-package-origin-configuration(siehe Die Einstellungen zur Herkunft des Pakets werden bearbeitet)

> **Note**
>
> Sie können die `publish` Origin-Control-Einstellung verwenden, um die Veröffentlichung eines generischen Paketnamens in einem Repository zuzulassen oder zu blockieren. Die `upstream` Einstellung gilt jedoch nicht für generische Pakete, da sie nicht aus einem Upstream-Repository abgerufen werden können.

- update-package-versions-status(siehe Der Status der Paketversion wird aktualisiert)

Generische Pakete veröffentlichen und konsumieren

Verwenden Sie den `publish-package-version` Befehl, um eine generische Paketversion und die zugehörigen Elemente zu veröffentlichen. Sie können die Elemente eines generischen Pakets mithilfe des `list-package-version-asset` Befehls auflisten und mit diesem Befehl herunterladen`get-package-version-asset`. Das folgende Thema enthält step-by-step Anweisungen zum Veröffentlichen von generischen Paketen oder zum Herunterladen generischer Paketressourcen mithilfe dieser Befehle.

Veröffentlichen eines generischen Pakets

Ein generisches Paket besteht aus einem Paketnamen, einem Namespace, einer Version und einem oder mehreren Assets (oder Dateien). In diesem Thema wird gezeigt, wie ein Paket mit dem Namen`my-package`, dem Namespace `my-ns` und der Version `1.0.0` veröffentlicht wird und das ein Objekt mit dem Namen enthält. `asset.tar.gz`

Voraussetzungen:

- Richten Sie das ein und konfigurieren Sie AWS Command Line Interface mit CodeArtifact (siehe Einrichtung mit AWS CodeArtifact)
- Besitzen Sie eine CodeArtifact Domain und ein Repository (siehe Erste Schritte mit dem AWS CLI)

Um ein generisches Paket zu veröffentlichen

1. Verwenden Sie den folgenden Befehl, um den SHA256 Hash für jede Datei zu generieren, die Sie in eine Paketversion hochladen möchten, und platzieren Sie den Wert in einer

Umgebungsvariablen. Dieser Wert wird als Integritätsprüfung verwendet, um sicherzustellen, dass sich der Dateiinhalt nach dem ursprünglichen Senden nicht geändert hat.

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. Rufen Sie `publish-package-version` auf, um das Asset hochzuladen und eine neue Paketversion zu erstellen.

> **Note**
>
> Wenn Ihr Paket mehr als ein Asset enthält, können Sie jedes Asset `publish-package-version` einmal aufrufen, um es hochzuladen. Geben Sie das `--unfinished` Argument für jeden Aufruf von an an an an`publish-package-version`, außer wenn das endgültige Asset hochgeladen wird. Durch das Auslassen `--unfinished` wird der Status der Paketversion auf gesetzt und es wird verhindert`Published`, dass weitere Inhalte in die Paketversion hochgeladen werden. Alternativ können Sie `--unfinished` bei jedem Aufruf von die Option „einschließen“ und dann den Status der Paketversion `Published` mithilfe des `update-package-versions-status` Befehls auf „setzen“. `publish-package-version`

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%
```

Nachfolgend sehen Sie die Ausgabe.

```
{
    "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": "43f24850b7b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-
SHA-256",
            "SHA-512":
 "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fced7bd1e80a0dca9ce320d95
SHA-512"
        }
    }
}
```

Generische Paketressourcen auflisten

Verwenden Sie den `list-package-version-assets` Befehl, um die in einem generischen Paket enthaltenen Ressourcen aufzulisten. Weitere Informationen finden Sie unter Listet die Ressourcen der Paketversion auf.

Das folgende Beispiel listet die Elemente `1.0.0` der Paketversion auf`my-package`.

Um die Ressourcen der Paketversion aufzulisten

- Rufen Sie `list-package-version-assets` auf, um die in einem generischen Paket enthaltenen Ressourcen aufzulisten.

  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
  ```

  Nachfolgend sehen Sie die Ausgabe.

  ```
  {
      "assets": [
          {
              "name": "asset.tar.gz",
              "size": 11,
              "hashes": {
                  "MD5": "41bba98d5b9219c43089eEXAMPLE-MD5",
                  "SHA-1": "69b215c25dd4cda1d997a786ec6EXAMPLE-SHA-1",
                  "SHA-256":
   "43f24850b7b7b7d79c5fa652418518fbdf427e602b1edabe6EXAMPLE-SHA-256",
                  "SHA-512":
   "3947382ac2c180ee3f2aba4f8788241527c8db9dfe9f4b039abe9fc560aaf5a1fced7bd1e80a0dca9ce320d95
  SHA-512"
              }
          }
      ],
      "package": "my-package",
      "format": "generic",
      "namespace": "my-ns",
      "version": "1.0.0",
  ```

```
    "versionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
}
```

Generische Paket-Assets werden heruntergeladen

Verwenden Sie den `get-package-version-asset` Befehl, um die Assets aus einem generischen Paket herunterzuladen. Weitere Informationen finden Sie unter Laden Sie die Ressourcen der Paketversion herunter.

Im folgenden Beispiel wird das Asset `asset.tar.gz` aus `1.0.0` der Version des Pakets in `my-package` das aktuelle Arbeitsverzeichnis in eine ebenfalls benannte Datei heruntergeladen`asset.tar.gz`.

Um Assets aus der Paketversion herunterzuladen

- Rufen Sie `get-package-version-asset` auf, um Ressourcen aus einem generischen Paket herunterzuladen.

  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
  ```

  Nachfolgend sehen Sie die Ausgabe.

  ```
  {
      "assetName": "asset.tar.gz",
      "packageVersion": "1.0.0",
      "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
  ```

```
}
```

Verwenden CodeArtifact mit CodeBuild

In diesen Themen wird beschrieben, wie Pakete in einem CodeArtifact Repository in einem AWS CodeBuild Build-Projekt verwendet werden.

Themen

- Verwenden von NPM-Paketen in CodeBuild
- Verwenden von Python-Paketen in CodeBuild
- Verwenden von Maven-Paketen in CodeBuild
- Verwenden von Paketen NuGet in CodeBuild
- Zwischenspeichern von Abhängigkeiten

Verwenden von NPM-Paketen in CodeBuild

Die folgenden Schritte wurden mit den Betriebssystemen getestet, die in den Docker-Images von aufgeführt sind. CodeBuild

Richten Sie Berechtigungen mit IAM-Rollen ein

Diese Schritte sind erforderlich, wenn Sie npm-Pakete von CodeArtifact in verwenden. CodeBuild

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Rollen. Bearbeiten Sie auf der Seite Rollen die Rolle, die von Ihrem CodeBuild Build-Projekt verwendet wird. Diese Rolle muss über die folgenden Berechtigungen verfügen.

```
{
  "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**
>
> Wenn Sie die Option auch zum Veröffentlichen von Paketen verwenden CodeBuild möchten, fügen Sie die **`codeartifact:PublishPackageVersion`** entsprechende Berechtigung hinzu.

Weitere Informationen finden Sie unter Ändern einer Rolle im IAM-Benutzerhandbuch.

Melden Sie sich an und verwenden Sie npm

Um npm-Pakete von zu verwenden CodeBuild, führen Sie den `login` Befehl aus dem `pre-build` Abschnitt Ihres Projekts aus, aus `buildspec.yaml` dem Pakete abgerufen werden `npm` sollen. CodeArtifact Weitere Informationen finden Sie unter Authentifizierung mit npm.

Nach `login` erfolgreicher Ausführung können Sie `npm` Befehle aus dem `build` Abschnitt zur Installation von NPM-Paketen ausführen.

Linux

> **Note**
>
> Ein Upgrade ist nur erforderlich, `pip3 install awscli --upgrade --user` wenn Sie ein älteres CodeBuild Image verwenden. AWS CLI Wenn Sie die neuesten Image-Versionen verwenden, können Sie diese Zeile entfernen.

```
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 msiexec "/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
```

Verwenden von Python-Paketen in CodeBuild

Die folgenden Schritte wurden mit den Betriebssystemen getestet, die in den Docker-Images von CodeBuild aufgeführt sind.

Richten Sie Berechtigungen mit IAM-Rollen ein

Diese Schritte sind erforderlich, wenn Sie Python-Pakete von CodeArtifact in verwenden CodeBuild.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

2. Wählen Sie im Navigationsbereich Rollen. Bearbeiten Sie auf der Seite Rollen die Rolle, die von Ihrem CodeBuild Build-Projekt verwendet wird. Diese Rolle muss über die folgenden Berechtigungen verfügen.

```
{
  "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**
>
> Wenn Sie die Option auch zum Veröffentlichen von Paketen verwenden CodeBuild möchten, fügen Sie die **`codeartifact:PublishPackageVersion`** entsprechende Berechtigung hinzu.

Weitere Informationen finden Sie unter Ändern einer Rolle im IAM-Benutzerhandbuch.

Melden Sie sich an und verwenden Sie Pip oder Twine

Um Python-Pakete von zu verwenden CodeBuild, führen Sie den `login` Befehl aus dem `pre-build` Abschnitt Ihrer `buildspec.yaml` Projektdatei aus, aus dem CodeArtifact Pakete abgerufen werden `pip` sollen. Weitere Informationen finden Sie unter CodeArtifact Mit Python verwenden.

Nach `login` erfolgreicher Ausführung können Sie `pip` Befehle aus dem `build` Abschnitt ausführen, um Python-Pakete zu installieren oder zu veröffentlichen.

Linux

> **Note**
>
> Ein Upgrade AWS CLI mit ist nur erforderlich, `pip3 install awscli --upgrade --user` wenn Sie ein älteres CodeBuild Image verwenden. Wenn Sie die neuesten Image-Versionen verwenden, können Sie diese Zeile entfernen.

Um Python-Pakete zu installieren mit`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
```

Um Python-Pakete zu veröffentlichen mit`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

Um Python-Pakete zu installieren mit`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 msiexec "/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
```

Um Python-Pakete zu veröffentlichen mit`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 msiexec "/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
```

Verwenden von Maven-Paketen in CodeBuild

Richten Sie Berechtigungen mit IAM-Rollen ein

Diese Schritte sind erforderlich, wenn Sie Maven-Pakete von CodeArtifact in verwenden. CodeBuild

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Rollen. Bearbeiten Sie auf der Seite Rollen die Rolle, die von Ihrem CodeBuild Build-Projekt verwendet wird. Diese Rolle muss über die folgenden Berechtigungen verfügen.

```
{
  "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**
>
> Wenn Sie es auch zum Veröffentlichen von Paketen verwenden CodeBuild möchten, fügen Sie die **`codeartifact:PutPackageMetadata`** Berechtigungen **`codeartifact:PublishPackageVersion`** und hinzu.

Weitere Informationen finden Sie unter Ändern einer Rolle im IAM-Benutzerhandbuch.

Verwenden Sie Gradle oder MVN

Um Maven-Pakete mit `gradle` oder zu verwenden`mvn`, speichern Sie das CodeArtifact Authentifizierungstoken in einer Umgebungsvariablen, wie unter Ein Authentifizierungstoken an eine Umgebungsvariable übergeben beschrieben. Im Folgenden wird ein Beispiel gezeigt.

> **Note**
>
> Ein Upgrade AWS CLI mit ist nur erforderlich, `pip3 install awscli --upgrade --user` wenn Sie ein älteres Image verwenden. CodeBuild Wenn Sie die neuesten Image-Versionen verwenden, können Sie diese Zeile entfernen.

```
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`
```

Um Gradle zu verwenden:

Wenn Sie auf die `CODEARTIFACT_AUTH_TOKEN` Variable in Ihrer `build.gradle` Gradle-Datei verwiesen haben, wie unter CodeArtifact Mit Gradle verwenden beschrieben, können Sie Ihren Gradle-Build aus dem Abschnitt aufrufen. `buildspec.yaml` `build`

```
build:
  commands:
    - gradle build
```

Um mvn zu verwenden:

Sie müssen Ihre Maven-Konfigurationsdateien konfigurieren (`settings.xml`und`pom.xml`) den Anweisungen unter CodeArtifact Mit mvn verwenden folgen.

Verwenden von Paketen NuGet in CodeBuild

Die folgenden Schritte wurden mit den Betriebssystemen getestet, die in den Docker-Images von CodeBuild aufgeführt sind.

Themen

- Richten Sie Berechtigungen mit IAM-Rollen ein
- Verbrauchen Sie Pakete NuGet
- Mit NuGet Paketen bauen
- Veröffentlichen Sie NuGet Pakete

Richten Sie Berechtigungen mit IAM-Rollen ein

Diese Schritte sind erforderlich, wenn Sie NuGet Pakete von CodeArtifact in CodeBuild verwenden.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen. Bearbeiten Sie auf der Seite Rollen die Rolle, die von Ihrem CodeBuild Build-Projekt verwendet wird. Diese Rolle muss über die folgenden Berechtigungen verfügen.

```
{
  "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**
>
> Wenn Sie die Option auch zum Veröffentlichen von Paketen verwenden CodeBuild möchten, fügen Sie die **`codeartifact:PublishPackageVersion`** entsprechende Berechtigung hinzu.

Weitere Informationen finden Sie unter Ändern einer Rolle im IAM-Benutzerhandbuch.

Verbrauchen Sie Pakete NuGet

Um NuGet Pakete zu konsumieren CodeBuild, fügen Sie Folgendes in die `buildspec.yaml` Datei Ihres Projekts ein.

1. Installieren Sie in `install` diesem Abschnitt den CodeArtifact Credential Provider, um Befehlszeilentools wie das Erstellen `msbuild` und `dotnet` Veröffentlichen von Paketen zu CodeArtifact konfigurieren.
2. Fügen Sie in `pre-build` diesem Abschnitt Ihr CodeArtifact Repository zu Ihrer NuGet Konfiguration hinzu.

Sehen Sie sich die folgenden `buildspec.yaml` Beispiele an. Weitere Informationen finden Sie unter Verwenden CodeArtifact mit NuGet.

Nachdem der Credential Provider installiert und Ihre Repository-Quelle hinzugefügt wurde, können Sie NuGet CLI-Tool-Befehle aus dem `build` Abschnitt ausführen, um NuGet Pakete zu konsumieren.

Linux

Um NuGet Pakete zu konsumieren mit`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

Um NuGet Pakete zu konsumieren mit`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
```

Mit NuGet Paketen bauen

Um mit NuGet Paketen von zu bauen CodeBuild, fügen Sie Folgendes in die `buildspec.yaml` Datei Ihres Projekts ein.

1. Installieren Sie in `install` diesem Abschnitt den CodeArtifact Credential Provider, um Befehlszeilentools wie das Erstellen `msbuild` und `dotnet` Veröffentlichen von Paketen zu CodeArtifact konfigurieren.

2. Fügen Sie in `pre-build` diesem Abschnitt Ihr CodeArtifact Repository zu Ihrer NuGet Konfiguration hinzu.

Sehen Sie sich die folgenden `buildspec.yaml` Beispiele an. Weitere Informationen finden Sie unter Verwenden CodeArtifact mit NuGet.

Nachdem der Credential Provider installiert und Ihre Repository-Quelle hinzugefügt wurde, können Sie NuGet CLI-Tool-Befehle wie `dotnet build` im `build` Abschnitt ausführen.

Linux

Um NuGet Pakete zu erstellen mit`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
```

Um NuGet Pakete zu bauen mit`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

Um NuGet Pakete zu bauen mit`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
```

Um NuGet Pakete zu bauen mit`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
```

Veröffentlichen Sie NuGet Pakete

Um NuGet Pakete von zu veröffentlichen CodeBuild, fügen Sie Folgendes in die `buildspec.yaml` Datei Ihres Projekts ein.

1. Installieren Sie in `install` diesem Abschnitt den CodeArtifact Credential Provider, um Befehlszeilentools wie das Erstellen `msbuild` und `dotnet` Veröffentlichen von Paketen zu CodeArtifact konfigurieren.
2. Fügen Sie in `pre-build` diesem Abschnitt Ihr CodeArtifact Repository zu Ihrer NuGet Konfiguration hinzu.

Sehen Sie sich die folgenden `buildspec.yaml` Beispiele an. Weitere Informationen finden Sie unter Verwenden CodeArtifact mit NuGet.

Nachdem der Credential Provider installiert und Ihre Repository-Quelle hinzugefügt wurde, können Sie NuGet CLI-Tool-Befehle aus dem `build` Abschnitt ausführen und Ihre NuGet Pakete veröffentlichen.

Linux

Um NuGet Pakete zu veröffentlichen mit`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

Um NuGet Pakete zu veröffentlichen mit`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
```

Zwischenspeichern von Abhängigkeiten

Sie können das lokale Caching aktivieren CodeBuild , um die Anzahl der Abhängigkeiten zu reduzieren, aus denen CodeArtifact für jeden Build abgerufen werden muss. Weitere Informationen finden Sie unter Build-Caching AWS CodeBuild imAWS CodeBuild Benutzerhandbuch. Nachdem Sie einen benutzerdefinierten lokalen Cache aktiviert haben, fügen Sie das Cache-Verzeichnis der `buildspec.yaml` Datei Ihres Projekts hinzu.

Wenn Sie beispielsweise Folgendes verwenden`mvn`, verwenden Sie Folgendes.

```
cache:
```

```
  paths:
    - '/root/.m2/**/*'
```

Verwenden Sie für andere Tools die in dieser Tabelle aufgeführten Cache-Ordner.

Tool	Cache-Verzeichnis
mvn	/root/.m2/**/*
gradle	/root/.gradle/caches/**/*
pip	/root/.cache/pip/**/*
npm	/root/.npm/**/*
nuget	/root/.nuget/**/*
yarn (classic)	/root/.cache/yarn/**/*

Überwachung CodeArtifact

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer CodeArtifact anderen AWS Lösungen. AWS bietet die folgenden Überwachungstools, mit denen Sie beobachten CodeArtifact, melden können, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen ergreifen können:

- Sie können Amazon verwenden EventBridge , um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse im AWS Rahmen von Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch undCodeArtifact Veranstaltungsformat und Beispiel.
- Sie können CloudWatch Amazon-Metriken verwenden, um die CodeArtifact Nutzung nach Vorgängen anzuzeigen. CloudWatch Die Metriken umfassen alle Anfragen CodeArtifact, die an gestellt wurden, und die Anfragen werden nach Konten geordnet angezeigt. Sie können diese Metriken in CloudWatch Metriken anzeigen, indem Sie zum Namespace Usage/By AWS Resource navigieren. AWS Weitere Informationen finden Sie unter Verwenden von CloudWatch Amazon-Metriken im CloudWatch Amazon-Benutzerhandbuch.

Themen

- CodeArtifact Ereignisse überwachen
- Verwenden Sie ein Ereignis, um eine CodePipeline Ausführung zu starten
- Verwenden Sie ein Ereignis, um eine Lambda-Funktion auszuführen

CodeArtifact Ereignisse überwachen

CodeArtifact ist in Amazon integriert EventBridge, einen Service, der Ereignisse automatisiert und darauf reagiert, einschließlich Änderungen in einem CodeArtifact Repository. Sie können Regeln für Ereignisse erstellen und konfigurieren, was passiert, wenn ein Ereignis mit einer Regel übereinstimmt. EventBridge hieß früher CloudWatch Events.

Die folgenden Aktionen können durch ein Ereignis ausgelöst werden:

- Eine AWS Lambda Funktion aufrufen.
- Aktivierung einer AWS Step Functions Zustandsmaschine.
- Benachrichtigung über ein Amazon SNS SNS-Thema oder eine Amazon SQS SQS-Warteschlange.
- Starten einer Pipeline in. AWS CodePipeline

CodeArtifact erzeugt ein Ereignis, wenn eine Paketversion erstellt, geändert oder gelöscht wird. Im Folgenden finden Sie Beispiele für CodeArtifact Ereignisse:

- Veröffentlichen einer neuen Paketversion (z. B. durch Ausführen`npm publish`).
- Hinzufügen eines neuen Assets zu einer vorhandenen Paketversion (z. B. durch Verschieben einer neuen JAR-Datei in ein vorhandenes Maven-Paket).
- Kopieren einer Paketversion von einem Repository in ein anderes mit`copy-package-versions`. Weitere Informationen finden Sie unter Pakete zwischen Repositorys kopieren.
- Löschen von Paketversionen mit`delete-package-versions`. Weitere Informationen finden Sie unter Löschen Sie ein Paket oder eine Paketversion.
- Löschen von Paketversionen mit`delete-package`. Für jede Version des gelöschten Pakets wird ein Ereignis veröffentlicht. Weitere Informationen finden Sie unter Löschen Sie ein Paket oder eine Paketversion.
- Beibehaltung einer Paketversion in einem Downstream-Repository, wenn sie aus einem Upstream-Repository abgerufen wurde. Weitere Informationen finden Sie unter Arbeiten mit Upstream-Repositorys in CodeArtifact.
- Aufnahme einer Paketversion aus einem externen Repository in ein CodeArtifact Repository. Weitere Informationen finden Sie unter Ein CodeArtifact Repository mit einem öffentlichen Repository Connect.

Ereignisse werden sowohl an das Konto übermittelt, dem die Domain gehört, als auch an das Konto, das das Repository verwaltet. Nehmen wir zum Beispiel an, dass das Konto `111111111111` Eigentümer der Domain `my_domain` ist. `222222222222`Das Konto erstellt ein Repository unter dem `my_domain` Namen`repo2`. Wenn eine neue Paketversion veröffentlicht wird`repo2`, erhalten beide Konten die EventBridge Ereignisse. Das Konto, das die Domäne besitzt (`111111111111`) empfängt Ereignisse für alle Repositorys in der Domäne. Wenn ein einzelnes Konto sowohl die Domain als auch das darin enthaltene Repository besitzt, wird nur ein einziges Ereignis zugestellt.

In den folgenden Themen wird das CodeArtifact Veranstaltungsformat beschrieben. Sie zeigen Ihnen, wie Sie CodeArtifact Ereignisse konfigurieren und wie Sie Ereignisse mit anderen AWS Diensten verwenden. Weitere Informationen finden Sie unter Erste Schritte mit Amazon EventBridge im EventBridge Amazon-Benutzerhandbuch.

CodeArtifact Veranstaltungsformat und Beispiel

Im Folgenden finden Sie Ereignisfelder und Beschreibungen sowie ein Beispiel für ein CodeArtifact Ereignis.

CodeArtifact Format des Ereignisses

Alle CodeArtifact Ereignisse enthalten die folgenden Felder.

Feld „Ereignis“	Beschreibung
version	Die Version des -Ereignisformats. Derzeit gibt es nur eine einzige Version,`0`.
id	Eine eindeutige Kennung für das Ereignis.
detail-type	Der Ereignistyp. Dies bestimmt die Felder im `detail` Objekt. Derjenige, der `detail-type` derzeit unterstützt wird, ist`CodeArtifact Package Version State Change`.
Quelle	Die Quelle des Ereignisses. Denn CodeArtifact das wird es sein`aws.codeartifact` .
Konto	Die AWS Konto-ID des Kontos, das das Ereignis empfängt.
time	Der genaue Zeitpunkt, zu dem das Ereignis ausgelöst wurde.
Region	Die Region, in der das Ereignis ausgelöst wurde.
Ressourcen	Eine Liste, die den ARN des Pakets enthält, das sich geändert hat. Die Liste enthält

Feld „Ereignis“	Beschreibung
	einen Eintrag. Hinweise zum ARN-Format von Paketen finden Sie unterGewähren Sie Schreibzugriff auf Pakete.
domainName	Die Domäne, die das Repository enthält, das das Paket enthält.
Besitzer der Domain	Die AWS Konto-ID des Besitzers der Domain.
Name des Repositorys	Das Repository, das das Paket enthält.
Repository-Administrator	Die AWS Konto-ID des Administrators des Repositorys.
Paket-Format	Das Format des Pakets, das das Ereignis ausgelöst hat.
Paket-Namespace	Der Namespace des Pakets, das das Ereignis ausgelöst hat.
Paketname	Der Name des Pakets, das das Ereignis ausgelöst hat.
Paketversion	Die Version des Pakets, das das Ereignis ausgelöst hat.
packageVersionState	Der Status der Paketversion, als das Ereignis ausgelöst wurde. Die möglichen Werte sind `Unfinished` , `Published` , `Unlisted`, `Archived` und `Disposed`.

Feld „Ereignis“	Beschreibung
packageVersionRevision	Ein Wert, der den Status der Assets und Metadaten der Paketversion eindeutig identifiziert, als das Ereignis ausgelöst wurde. Wenn die Paketversion geändert wird (z. B. durch Hinzufügen einer weiteren JAR-Datei zu einem Maven-Paket), `packageVersionRevision` ändern sich die Änderungen.
Änderungen. Hinzugefügte Vermögenswerte	Die Anzahl der Assets, die einem Paket hinzugefügt wurden, das ein Ereignis ausgelöst hat. Beispiele für ein Asset sind eine Maven-JAR-Datei oder ein Python-Rad.
Änderungen. Entfernte Objekte	Die Anzahl der Elemente, die aus einem Paket entfernt wurden, das ein Ereignis ausgelöst hat.
changes.assetsUpdated	Die Anzahl der in dem Paket geänderten Assets, die das Ereignis ausgelöst haben.
changes.metadataUpdated	Ein boolescher Wert, der auf gesetzt wird, wenn das Ereignis geänderte Metadaten auf Paketebene enthält`true`. Ein Ereignis könnte beispielsweise eine Maven-Datei ändern. `pom.xml`
changes.statusChanged	Ein boolescher Wert, der auf gesetzt wird, `true` wenn das Ereignis geändert wird (z. B. wenn es von zu geändert `packageVersionStatus` wird). `packageVersionStatus` `Unfinished` `Published`
OperationType	Beschreibt den allgemeinen Typ der Änderung der Paketversion. Die möglichen Werte sind `Created`, `Updated` und `Deleted`.

Feld „Ereignis“	Beschreibung
sequenceNumber	Eine Ganzzahl, die eine Ereignisnummer für ein Paket angibt. Jedes Ereignis in einem Paket erhöht die Anzahl, `sequenceNumber` sodass Ereignisse nacheinander angeordnet werden können. Ein Ereignis kann die um eine beliebige `sequenceNumber` Ganzzahl erhöhen. **Note** EventBridge Ereignisse werden möglicherweise nicht in der richtigen Reihenfolge empfangen. `sequenceNumber` kann verwendet werden, um ihre tatsächliche Reihenfolge zu bestimmen.
eventDeduplicationId	Eine ID, die verwendet wird, um doppelte EventBridge Ereignisse zu unterscheiden. In seltenen Fällen EventBridge kann dieselbe Regel mehr als einmal für ein einzelnes Ereignis oder einen geplanten Zeitpunkt ausgelöst werden. Oder es kann sein, dass dasselbe Ziel mehr als einmal für eine bestimmte ausgelöste Regel aufgerufen wird.

CodeArtifact Beispiel für ein Ereignis

Das Folgende ist ein Beispiel für ein CodeArtifact Ereignis, das ausgelöst werden könnte, wenn ein npm-Paket veröffentlicht wird.

```
{
      "version":"0",
      "id":"73f03fec-a137-971e-6ac6-07c8ffffffff",
      "detail-type":"CodeArtifact Package Version State Change",
```

```
      "source":"aws.codeartifact",
      "account":"123456789012",
      "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":"0E5DE26A4CD79FDF3EBC4924FFFFFFFF",
        "changes":{
          "assetsAdded":1,
          "assetsRemoved":0,
          "metadataUpdated":true,
          "assetsUpdated":0,
          "statusChanged":true
        },
        "operationType":"Created",
        "sequenceNumber":1,
        "eventDeduplicationId":"2mEO0A2Ke07rWUTBXk3CAiQhdTXF4N94LNaT/ffffff="
      }
    }
```

Verwenden Sie ein Ereignis, um eine CodePipeline Ausführung zu starten

Dieses Beispiel zeigt, wie eine EventBridge Amazon-Regel so konfiguriert wird, dass eine AWS CodePipeline Ausführung beginnt, wenn eine Paketversion in einem CodeArtifact Repository veröffentlicht, geändert oder gelöscht wird.

Themen

- EventBridgeBerechtigungen konfigurieren
- Erstellen Sie die EventBridge Regel

- Erstellen Sie das EventBridge Regelziel

EventBridgeBerechtigungen konfigurieren

Sie müssen Berechtigungen hinzufügen EventBridge , die CodePipeline zum Aufrufen der von Ihnen erstellten Regel verwendet werden sollen. Um diese Berechtigungen mithilfe von AWS Command Line Interface (AWS CLI) hinzuzufügen, folgen Sie Schritt 1 unter Erstellen einer CloudWatch Ereignisregel für eine CodeCommit Quelle (CLI) im AWS CodePipeline Benutzerhandbuch.

Erstellen Sie die EventBridge Regel

Verwenden Sie den `put-rule` Befehl mit den `--event-pattern` Parametern `--name` und, um die Regel zu erstellen. Das Ereignismuster gibt Werte an, die mit den Inhalten der einzelnen Ereignisse abgeglichen werden. Das Ziel wird ausgelöst, wenn das Muster mit dem Ereignis übereinstimmt. Das folgende Muster entspricht beispielsweise CodeArtifact Ereignissen aus dem `myrepo` Repository in der `my_domain` Domäne.

```
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"]}}'
```

Erstellen Sie das EventBridge Regelziel

Mit dem folgenden Befehl wird der Regel ein Ziel hinzugefügt, sodass eine CodePipeline Ausführung ausgelöst wird, wenn ein Ereignis der Regel entspricht. Geben Sie für den `RoleArn` Parameter den Amazon-Ressourcennamen (ARN) der zuvor in diesem Thema erstellten Rolle an.

```
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'
```

Verwenden Sie ein Ereignis, um eine Lambda-Funktion auszuführen

Dieses Beispiel zeigt Ihnen, wie Sie eine EventBridge Regel konfigurieren, die eine AWS Lambda Funktion startet, wenn eine Paketversion in einem CodeArtifact Repository veröffentlicht, geändert oder gelöscht wird.

Weitere Informationen finden Sie unter Tutorial: Verwendung von AWS Lambda Funktionen planen EventBridge im EventBridge Amazon-Benutzerhandbuch.

Themen

- Erstellen Sie die EventBridge Regel
- Erstellen Sie das EventBridge Regelziel
- Berechtigungen konfigurieren EventBridge

Erstellen Sie die EventBridge Regel

Um eine Regel zu erstellen, die eine Lambda-Funktion startet, verwenden Sie den `put-rule` Befehl mit den `--event-pattern` Optionen `--name` und. Das folgende Muster spezifiziert npm-Pakete im `@types` Gültigkeitsbereich eines beliebigen Repositorys in der `my_domain` Domäne.

```
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"]}}'
```

Erstellen Sie das EventBridge Regelziel

Der folgende Befehl fügt der Regel, die die Lambda-Funktion ausführt, wenn ein Ereignis der Regel entspricht, ein Ziel hinzu. Geben Sie für den `arn` Parameter den Amazon-Ressourcennamen (ARN) der Lambda-Funktion an.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

Berechtigungen konfigurieren EventBridge

Verwenden Sie den `add-permission` Befehl, um der Regel Berechtigungen zum Aufrufen einer Lambda-Funktion zu erteilen. Geben Sie für den `--source-arn` Parameter den ARN der Regel an, die Sie zuvor in diesem Beispiel erstellt haben.

```
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
```

Sicherheit in CodeArtifact

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den geltenden Compliance-Programmen finden Sie unter AWS-Services in Umfang nach Compliance-Programm . CodeArtifact
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung anwenden können CodeArtifact. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen CodeArtifact , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS-Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer CodeArtifact Ressourcen unterstützen.

Themen

- Datenschutz in AWS CodeArtifact
- Überwachung CodeArtifact
- Konformitätsvalidierung für AWS CodeArtifact
- AWS CodeArtifact Authentifizierung und Tokens
- Resilienz in AWS CodeArtifact
- Sicherheit der Infrastruktur in AWS CodeArtifact
- Angriffe zur Substitution von Abhängigkeiten
- Identity and Access Management für AWS CodeArtifact

Datenschutz in AWS CodeArtifact

Das AWS Modell der gilt für den Datenschutz in AWS CodeArtifact. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der CodeArtifact API oder auf andere AWS-Services Weise arbeiten oder diese verwenden. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet

werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung

Verschlüsselung ist ein wichtiger Teil der CodeArtifact Sicherheit. Ein Teil der Verschlüsselung, z. B. für Daten während der Übertragung, ist standardmäßig vorgesehen und erfordert nichts von Ihnen. Andere Verschlüsselungen, z. B. für Daten im Ruhezustand, können Sie bei der Erstellung Ihres Projekts oder Builds konfigurieren.

- Verschlüsselung von Daten im Ruhezustand — Alle darin gespeicherten Ressourcen CodeArtifact werden mithilfe von AWS KMS keys (KMS-Schlüsseln) verschlüsselt. Dies schließt alle Ressourcen in allen Paketen in allen Repositorys ein. Für jede Domain wird ein KMS-Schlüssel verwendet, um all ihre Ressourcen zu verschlüsseln. Standardmäßig wird ein AWS verwalteter KMS-Schlüssel verwendet, sodass Sie keinen KMS-Schlüssel erstellen müssen. Wenn Sie möchten, können Sie einen vom Kunden verwalteten KMS-Schlüssel verwenden, den Sie erstellen und konfigurieren. Weitere Informationen finden Sie unter Schlüssel erstellen und Konzepte des AWS Schlüsselverwaltungsdienstes im AWS Key Management Service Benutzerhandbuch. Sie können einen vom Kunden verwalteten KMS-Schlüssel angeben, wenn Sie eine Domäne erstellen. Weitere Informationen finden Sie unter Arbeiten Sie mit Domänen in CodeArtifact.
- Verschlüsselung von Daten während der Übertragung — Die gesamte Kommunikation zwischen Kunden CodeArtifact CodeArtifact und zwischen Kunden sowie deren nachgelagerten Abhängigkeiten ist durch TLS-Verschlüsselung geschützt.

Datenschutz für Datenverkehr

Sie können die Sicherheit Ihrer CodeArtifact Domains und der darin enthaltenen Ressourcen verbessern, indem Sie die Verwendung eines VPC-Endpunkts (Virtual Private Cloud) mit Schnittstelle konfigurieren CodeArtifact . Dazu benötigen Sie kein Internet-Gateway, kein NAT-Gerät oder kein Virtual Private Gateway. Weitere Informationen finden Sie unter Arbeiten mit Amazon VPC-Endpunkten. Weitere Informationen zu AWS PrivateLink VPC-Endpunkten finden Sie unter AWS PrivateLinkund Zugreifen auf AWS-Services über. PrivateLink

Überwachung CodeArtifact

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit AWS CodeArtifact und Leistung Ihrer AWS Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können. AWS bietet Folgendes für die Überwachung Ihrer CodeArtifact Ressourcen und für die Reaktion auf potenzielle Vorfälle:

Themen

- Protokollierung von CodeArtifact API-Aufrufen mit AWS CloudTrail

Protokollierung von CodeArtifact API-Aufrufen mit AWS CloudTrail

CodeArtifact ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden CodeArtifact. CloudTrail erfasst alle API-Aufrufe CodeArtifact als Ereignisse, einschließlich Aufrufe von Paketmanager-Clients.

Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon Simple Storage Service (Amazon S3) -Bucket aktivieren, einschließlich Ereignissen für CodeArtifact. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde CodeArtifact, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

CodeArtifact Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindet CodeArtifact, wird diese Aktivität zusammen mit anderen CloudTrail AWS Serviceereignissen im Ereignisverlauf in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für CodeArtifact, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von

Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können auch andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie unter den folgenden Themen:

- Einen Trail für Ihr AWS-Konto erstellen
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail

Wenn die CloudTrail Protokollierung in Ihrem AWS Konto aktiviert ist, werden API-Aufrufe von CodeArtifact Aktionen in CloudTrail Protokolldateien aufgezeichnet, wo sie zusammen mit anderen AWS Serviceaufzeichnungen geschrieben werden. CloudTrail bestimmt anhand eines Zeitraums und der Dateigröße, wann eine neue Datei erstellt und in sie geschrieben werden soll.

Alle CodeArtifact Aktionen werden von protokolliert CloudTrail. Beispielsweise generieren Aufrufe der Aktionen `ListRepositories` (in den Aktionen AWS CLI,`aws codeartifact list-repositories`), `CreateRepository` (`aws codeartifact create-repository`) und `ListPackages` (`aws codeartifact list-packages`) zusätzlich zu den Paketmanager-Client-Befehlen Einträge in den CloudTrail Protokolldateien. Paketmanager-Clientbefehle stellen in der Regel mehr als eine HTTP-Anfrage an den Server. Jede Anforderung generiert ein separates CloudTrail Protokollereignis.

Kontoübergreifende Übermittlung von Protokollen CloudTrail

Bis zu drei separate Konten erhalten CloudTrail Protokolle für einen einzelnen API-Aufruf:

- Das Konto, das die Anfrage gestellt hat, z. B. das Konto, das angerufen hat. `GetAuthorizationToken`
- Das Repository-Administratorkonto — zum Beispiel das Konto, das das aufgerufene Repository verwaltet. `ListPackages`
- Das Konto des Domaininhabers — zum Beispiel das Konto, dem die Domain gehört, die das Repository enthält, für das eine API aufgerufen wurde.

`ListRepositoriesInDomain`Das APIs sind also Aktionen gegen eine Domain und nicht gegen ein bestimmtes Repository, nur das aufrufende Konto und das Konto des Domaininhabers erhalten das

CloudTrail Protokoll. Denn APIs `ListRepositories` solche sind gegen keine Ressource autorisiert, nur das Konto des Aufrufers erhält das CloudTrail Protokoll.

CodeArtifact Logdateieinträge verstehen

CloudTrail Protokolldateien können einen oder mehrere Protokolleinträge enthalten. In jedem Eintrag werden mehrere Ereignisse im JSON-Format aufgelistet. Ein Protokollereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anforderungsparameter. Protokolleinträge sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Themen

- Beispiel: Ein Logeintrag für den Aufruf der GetAuthorizationToken API
- Beispiel: Ein Protokolleintrag zum Abrufen einer npm-Paketversion

Beispiel: Ein Logeintrag für den Aufruf der GetAuthorizationToken API

Ein von erstellter Protokolleintrag `GetAuthorizationToken` enthält den Domainnamen in das `requestParameters` Feld.

```
{
  "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"
}
```

Beispiel: Ein Protokolleintrag zum Abrufen einer npm-Paketversion

Bei Anfragen aller Paketmanager-Clients, einschließlich des **npm**Clients, werden zusätzliche Daten protokolliert, darunter der Domänenname, der Repository-Name und der Paketname im `requestParameters` Feld. Der URL-Pfad und die HTTP-Methode werden im `additionalEventData` Feld protokolliert.

```
{
   "eventVersion": "1.05",
   "userIdentity": {
       "type": "AssumedRole",
       "principalId": "AIDACKCEVSQ6C2EXAMPLE",
       "arn": "arn:aws:sts::123456789012:assumed-role/Console/example",
       "accountId": "123456789012",
       "accessKeyId": "ASIAIJIOBJIBSREXAMPLE",
       "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"
}
```


Konformitätsvalidierung für AWS CodeArtifact

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Compliance und Governance im Bereich Sicherheit – In diesen Anleitungen für die Lösungsimplementierung werden Überlegungen zur Architektur behandelt. Außerdem werden Schritte für die Bereitstellung von Sicherheits- und Compliance-Features beschrieben.
- Referenz für berechtigte HIPAA-Services – Listet berechtigte HIPAA-Services auf. Nicht alle AWS-Services sind HIPAA-fähig.
- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerelementreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.

- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

AWS CodeArtifact Authentifizierung und Tokens

CodeArtifact erfordert, dass sich Benutzer beim Dienst authentifizieren, um Paketversionen zu veröffentlichen oder zu nutzen. Sie müssen sich beim CodeArtifact Dienst authentifizieren, indem Sie mit Ihren AWS Anmeldeinformationen ein Autorisierungstoken erstellen. Um ein Autorisierungstoken zu erstellen, müssen Sie über die richtigen Berechtigungen verfügen. Informationen zu den Berechtigungen, die zum Erstellen eines Autorisierungstokens erforderlich sind, finden Sie im `GetAuthorizationToken` Eintrag in derAWS CodeArtifact Referenz zu Berechtigungen. Weitere allgemeine Informationen zu CodeArtifact Berechtigungen finden Sie unterWie AWS CodeArtifact funktioniert mit IAM.

Um ein Autorisierungstoken von abzurufen CodeArtifact, müssen Sie die GetAuthorizationToken API aufrufen. Wenn Sie den verwenden AWS CLI, können Sie `GetAuthorizationToken` mit dem `get-authorization-token` Befehl `login` oder aufrufen.

> **Note**
>
> Root-Benutzer können nicht anrufen`GetAuthorizationToken`.

- `aws codeartifact login`: Dieser Befehl macht es einfach, gängige Paketmanager für die Verwendung CodeArtifact in einem einzigen Schritt zu konfigurieren. Beim `login` Aufrufen wird ein Token mit dem Token `GetAuthorizationToken` und dem richtigen CodeArtifact Repository-Endpunkt abgerufen und Ihr Paketmanager wird mit diesem konfiguriert. Die Support-Paketmanager lauten wie folgt:
  - dotnet
  - npm
  - Nuget
  - pip
  - schnell
  - binden

- `aws codeartifact get-authorization-token`: Für Paketmanager, die nicht von unterstützt werden`login`, können Sie Ihren Paketmanager `get-authorization-token` direkt aufrufen und dann nach Bedarf mit dem Token konfigurieren, indem Sie es beispielsweise zu einer Konfigurationsdatei hinzufügen oder es als Umgebungsvariable speichern.

CodeArtifact Autorisierungstoken sind standardmäßig für einen Zeitraum von 12 Stunden gültig. Token können mit einer Lebensdauer zwischen 15 Minuten und 12 Stunden konfiguriert werden. Wenn die Gültigkeitsdauer abläuft, müssen Sie ein anderes Token abrufen. Die Gültigkeitsdauer des Tokens beginnt danach `login` oder `get-authorization-token` wird aufgerufen.

Wenn `login` oder während der Übernahme einer Rolle aufgerufen `get-authorization-token` wird, können Sie die Lebensdauer des Tokens so konfigurieren, dass sie der verbleibenden Zeit in der Sitzungsdauer der Rolle entspricht, indem Sie den Wert `--duration-seconds` auf setzen`0`. Andernfalls ist die Gültigkeitsdauer des Tokens unabhängig von der maximalen Sitzungsdauer der Rolle. Angenommen, Sie rufen an `sts assume-role` und geben eine Sitzungsdauer von 15 Minuten an und rufen dann an, `login` um ein CodeArtifact Autorisierungstoken abzurufen. In diesem Fall ist das Token für den gesamten Zeitraum von 12 Stunden gültig, auch wenn dieser länger als die Sitzungsdauer von 15 Minuten ist. Informationen zur Steuerung der Sitzungsdauer finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

Mit dem Befehl erstellte Tokens **`login`**

Der `aws codeartifact login` Befehl ruft ein Token mit dem Token ab `GetAuthorizationToken` und konfiguriert Ihren Paketmanager mit dem Token und dem richtigen CodeArtifact Repository-Endpunkt.

In der folgenden Tabelle werden die Parameter für den `login` Befehl beschrieben.

Parameter	Erforderlich	Beschreibung
`--tool`	Ja	Der Paketmanager, bei dem Sie sich authentifizieren möchten. Mögliche Werte sind`dotnet`,`npm`, `nugetpip`, `swift` und`twine`.
`--domain`	Ja	Der Domainname, zu dem das Repository gehört.

Parameter	Erforderlich	Beschreibung
`--domain-owner`	Nein	Die ID des Besitzers der Domain. Dieser Parameter ist erforderlich, wenn Sie auf eine Domain zugreifen, die einem AWS Konto gehört, für das Sie nicht authentifiziert sind. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.
`--repository`	Ja	Der Name des Repositorys, für das Sie sich authentifizieren möchten.
`--duration-seconds`	Nein	Die Zeit in Sekunden, in der die Anmeldeinformationen gültig sind. Der Mindestwert ist 900* und der Höchstwert ist 43200.
`--namespace`	Nein	Ordnet Ihrem Repository-Tool einen Namespace zu.
`--dry-run`	Nein	Drucken Sie nur die Befehle aus, die ausgeführt würden, um Ihr Tool mit Ihrem Repository zu verbinden, ohne Änderungen an Ihrer Konfiguration vorzunehmen.
*Ein Wert von 0 ist auch gültig, wenn Sie anrufen `login` und gleichzeitig eine Rolle übernehmen. Wenn Sie `login` mit aufrufen, `--duration-seconds 0` wird ein Token erstellt, dessen Lebensdauer der verbleibenden Zeit in der Sitzungsdauer einer übernommenen Rolle entspricht.		

Das folgende Beispiel zeigt, wie ein Autorisierungstoken mit dem `login` Befehl abgerufen wird.

```
aws codeartifact login \
    --tool dotnet | npm | nuget | pip | swift | twine \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo
```

Spezifische Anleitungen zur Verwendung des `login` Befehls mit npm finden Sie unter. Konfigurieren und verwenden Sie npm mit CodeArtifact Informationen zu Python finden Sie unter CodeArtifact Mit Python verwenden.

Zum Aufrufen der **`GetAuthorizationToken`** API sind Berechtigungen erforderlich

`sts:GetServiceBearerToken`Sowohl die als auch die `codeartifact:GetAuthorizationToken` Berechtigungen sind erforderlich, um die CodeArtifact `GetAuthorizationToken` API aufzurufen.

Um einen Paketmanager mit einem CodeArtifact Repository zu verwenden, muss Ihr IAM-Benutzer oder Ihre IAM-Rolle dies zulassen`sts:GetServiceBearerToken`. Die Berechtigung `sts:GetServiceBearerToken` kann zwar zu einer CodeArtifact Domänenressourcenrichtlinie hinzugefügt werden, hat jedoch keine Auswirkung auf diese Richtlinie.

Mit der **`GetAuthorizationToken`** API erstellte Tokens

Sie können aufrufen`get-authorization-token`, um ein Autorisierungstoken von CodeArtifact abzurufen.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text
```

Mit dem `--duration-seconds` Argument können Sie ändern, wie lange ein Token gültig ist. Der Mindestwert ist 900 und der Höchstwert ist 43200. Im folgenden Beispiel wird ein Token erstellt, das 1 Stunde (3600 Sekunden) gültig ist.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 3600
```

Wenn Sie aufrufen, `get-authorization-token` während Sie eine Rolle übernehmen, ist die Lebensdauer des Tokens unabhängig von der maximalen Sitzungsdauer der Rolle. Sie können das

Token so konfigurieren, dass es abläuft, wenn die Sitzungsdauer der angenommenen Rolle abläuft, indem Sie `--duration-seconds` es auf 0 setzen.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 0
```

Weitere Informationen finden Sie in der folgenden Dokumentation:

- Hinweise zu Tokens und Umgebungsvariablen finden Sie unterÜbergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen.
- Python-Benutzer finden Sie unter Konfigurieren Sie pip ohne den Login-Befehl oderKonfigurieren und verwenden Sie Twine mit CodeArtifact.
- Für Maven-Benutzer finden Sie weitere Informationen unter CodeArtifact Mit Gradle verwenden oder CodeArtifact Mit MVN verwenden.
- Für NPM-Benutzer siehe. Konfiguration von npm ohne Verwendung des Login-Befehls

Übergeben Sie ein Authentifizierungstoken mithilfe einer Umgebungsvariablen

AWS CodeArtifact verwendet von der `GetAuthorizationToken` API bereitgestellte Autorisierungstoken, um Anfragen von Build-Tools wie Maven und Gradle zu authentifizieren und zu autorisieren. Weitere Informationen zu diesen Authentifizierungstoken finden Sie unter. Mit der `GetAuthorizationToken` API erstellte Tokens

Sie können diese Authentifizierungstoken in einer Umgebungsvariablen speichern, die von einem Build-Tool gelesen werden kann, um das Token zu erhalten, das zum Abrufen von Paketen aus einem CodeArtifact Repository oder zum Veröffentlichen von Paketen in diesem Repository benötigt wird.

Aus Sicherheitsgründen ist dieser Ansatz dem Speichern des Tokens in einer Datei vorzuziehen, wo es von anderen Benutzern oder Prozessen gelesen oder versehentlich in die Quellcodeverwaltung eingecheckt werden kann.

1. Konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschriebenInstallieren oder aktualisieren Sie und konfigurieren Sie dann das AWS CLI.
2. Legen Sie die `CODEARTIFACT_AUTH_TOKEN`-Umgebungsvariable fest:

> **Note**
>
> In einigen Szenarien müssen Sie das `--domain-owner` Argument nicht angeben. Weitere Informationen finden Sie unter Kontenübergreifende Domänen.

- macOS oder Linux:

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

- Windows (unter Verwendung der Standard-Befehlsshell):

```
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
```

- Windows PowerShell:

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text
```

Widerrufen von CodeArtifact Autorisierungstoken

Wenn ein authentifizierter Benutzer ein Token für den Zugriff auf CodeArtifact Ressourcen erstellt, ist dieses Token gültig, bis der anpassbare Zugriffszeitraum abgelaufen ist. Die Standardzugriffsdauer beträgt 12 Stunden. Unter bestimmten Umständen möchten Sie möglicherweise den Zugriff auf ein Token widerrufen, bevor der Zugriffszeitraum abgelaufen ist. Sie können den Zugriff auf CodeArtifact Ressourcen widerrufen, indem Sie diese Anweisungen befolgen.

Wenn Sie das Zugriffstoken mit temporären Sicherheitsanmeldedaten wie übernommenen Rollen oder Verbundbenutzerzugriff erstellt haben, können Sie den Zugriff widerrufen, indem Sie eine IAM-Richtlinie aktualisieren, um den Zugriff zu verweigern. Weitere Informationen finden

Sie unter Deaktivierung von Berechtigungen für temporäre Sicherheitsanmeldedaten im IAM-Benutzerhandbuch.

Wenn Sie für die Erstellung des Zugriffstokens langfristige IAM-Benutzeranmeldedaten verwendet haben, müssen Sie die Benutzerrichtlinie ändern, um den Zugriff zu verweigern oder den IAM-Benutzer zu löschen. Weitere Informationen finden Sie unter Ändern der Berechtigungen für einen IAM-Benutzer oder Löschen eines IAM-Benutzers.

Resilienz in AWS CodeArtifact

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. AWS CodeArtifact arbeitet in mehreren Availability Zones und speichert Artefaktdaten und Metadaten in Amazon S3 und Amazon DynamoDB. Ihre verschlüsselten Daten werden redundant in mehreren Einrichtungen und auf mehreren Geräten in jeder Einrichtung gespeichert, wodurch sie hochverfügbar und äußerst robust sind.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Sicherheit der Infrastruktur in AWS CodeArtifact

Als verwalteter Dienst AWS CodeArtifact ist er durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff CodeArtifact über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Angriffe zur Substitution von Abhängigkeiten

Paketmanager vereinfachen das Verpacken und Teilen von wiederverwendbarem Code. Bei diesen Paketen kann es sich um private Pakete handeln, die von einer Organisation zur Verwendung in ihren Anwendungen entwickelt wurden, oder es kann sich um öffentliche Pakete handeln, in der Regel Open-Source-Pakete, die außerhalb einer Organisation entwickelt und über öffentliche Paket-Repositorys verteilt werden. Bei der Anforderung von Paketen verlassen sich Entwickler auf ihren Paketmanager, um neue Versionen ihrer Abhängigkeiten abzurufen. Angriffe zur Substitution von Abhängigkeiten, auch bekannt als Dependency Confusion Attacks, nutzen die Tatsache aus, dass ein Paketmanager normalerweise keine Möglichkeit hat, legitime Versionen eines Pakets von bösartigen Versionen zu unterscheiden.

Angriffe zur Substitution von Abhängigkeiten gehören zu einer Untergruppe von Hacks, die als Software-Supply-Chain-Angriffe bezeichnet werden. Ein Angriff auf die Software-Lieferkette ist ein Angriff, bei dem Sicherheitslücken überall in der Software-Lieferkette ausgenutzt werden.

Ein Angriff zur Substitution von Abhängigkeiten kann sich gegen jeden richten, der sowohl intern entwickelte Pakete als auch Pakete verwendet, die aus öffentlichen Repositorien abgerufen wurden. Die Angreifer identifizieren interne Paketnamen und platzieren dann strategisch bösartigen Code mit demselben Namen in öffentlichen Paket-Repositorys. In der Regel wird der bösartige Code in einem Paket mit einer hohen Versionsnummer veröffentlicht. Paketmanager rufen den bösartigen Code aus diesen öffentlichen Feeds ab, weil sie glauben, dass es sich bei den bösartigen Paketen um die neuesten Versionen des Pakets handelt. Dies führt zu einer „Verwechslung“ oder „Ersetzung“ zwischen dem gewünschten Paket und dem bösartigen Paket, wodurch der Code kompromittiert wird.

Um Angriffe durch die Substitution von Abhängigkeiten zu verhindern, AWS CodeArtifact bietet es Kontrollen zur Paketherkunft. Kontrollen zur Paketherkunft sind Einstellungen, die steuern, wie Pakete zu Ihren Repositorys hinzugefügt werden können. Mit diesen Steuerelementen kann sichergestellt werden, dass Paketversionen nicht gleichzeitig direkt in Ihrem Repository veröffentlicht und aus öffentlichen Quellen aufgenommen werden können. So schützen Sie sich vor Angriffen, die Abhängigkeiten ersetzen. Ursprungskontrollen können für einzelne Pakete und mehrere Pakete eingerichtet werden, indem Ursprungskontrollen für Paketgruppen eingerichtet werden. Weitere

Informationen zu Kontrollen zur Herkunft von Paketen und deren Änderung finden Sie unter Die Einstellungen zur Herkunft des Pakets werden bearbeitet undHerkunftskontrollen für Paketgruppen.

Identity and Access Management für AWS CodeArtifact

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. CodeArtifact IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie AWS CodeArtifact funktioniert mit IAM
- Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact
- Verwenden von Tags zur Steuerung des Zugriffs auf CodeArtifact-Ressourcen
- AWS CodeArtifact Referenz zu Berechtigungen
- Problembehandlung bei AWS CodeArtifact Identität und Zugriff

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. CodeArtifact

Dienstbenutzer — Wenn Sie den CodeArtifact Dienst für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr CodeArtifact Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Unter Problembehandlung bei AWS CodeArtifact Identität und Zugriff finden Sie nützliche Informationen für den Fall, dass Sie keinen Zugriff auf eine Feature in CodeArtifact haben.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die CodeArtifact Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf CodeArtifact. Es ist Ihre Aufgabe,

zu bestimmen, auf welche CodeArtifact Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM nutzen kann CodeArtifact, finden Sie unterWie AWS CodeArtifact funktioniert mit IAM.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf CodeArtifact verfassen können. Beispiele für CodeArtifact identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere

Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen

Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von

Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen

ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und

der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

Wie AWS CodeArtifact funktioniert mit IAM

Bevor Sie IAM zur Verwaltung des Zugriffs auf verwenden, sollten Sie sich darüber informieren CodeArtifact, mit welchen IAM-Funktionen Sie arbeiten können. CodeArtifact

IAM-Funktionen, die Sie mit verwenden können AWS CodeArtifact

IAM-Feature	CodeArtifact Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Ja
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Nein
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Nein

IAM-Feature	CodeArtifact Unterstützung
Serviceverknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie CodeArtifact und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für CodeArtifact

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für CodeArtifact

Beispiele für CodeArtifact identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact

Ressourcenbasierte Richtlinien finden Sie in CodeArtifact

Unterstützt ressourcenbasierte Richtlinien: Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können

Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für CodeArtifact

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der CodeArtifact Aktionen finden Sie unter Aktionen definiert von AWS CodeArtifact in der Serviceautorisierungsreferenz.

Bei Richtlinienaktionen wird vor der Aktion das folgende Präfix CodeArtifact verwendet:

```
codeartifact
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "codeartifact:action1",
      "codeartifact:action2"
         ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "codeartifact:Describe*"
```

Beispiele für CodeArtifact identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact

Politische Ressourcen für CodeArtifact

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der CodeArtifact Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Resources defined by AWS CodeArtifact in der Service Authorization Reference. Informationen zu

den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von AWS definierte AktionenCodeArtifact. Beispiele für die Angabe von CodeArtifact Ressourcen ARNs in Richtlinien finden Sie unterAWS CodeArtifact Ressourcen und Operationen.

Bedingungsschlüssel für Richtlinien für CodeArtifact

Unterstützt dienstspezifische Richtlinien-Bedingungsschlüssel: Nein

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

> **Note**
>
> AWS CodeArtifact unterstützt die folgenden AWS globalen Bedingungskontextschlüssel nicht:
>
> - Referer
> - UserAgent

Eine Liste der CodeArtifact Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS CodeArtifact in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Aktionen definiert von AWS CodeArtifact.

Beispiele für CodeArtifact identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact

ACLs in CodeArtifact

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit CodeArtifact

Unterstützt ABAC (Tags in Richtlinien): Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Weitere Informationen zum Markieren von CodeArtifact Ressourcen, einschließlich beispielhafter identitätsbasierter Richtlinien zur Beschränkung des Zugriffs auf eine Ressource auf der Grundlage der Tags dieser Ressource, finden Sie unter. Verwenden von Tags zur Steuerung des Zugriffs auf CodeArtifact-Ressourcen

Verwenden temporärer Anmeldeinformationen mit CodeArtifact

Unterstützt temporäre Anmeldeinformationen: Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln von einer Benutzerrolle zu einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für CodeArtifact

Unterstützt Forward Access Sessions (FAS): Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Es gibt zwei CodeArtifact API-Aktionen, für die der aufrufende Principal über Berechtigungen für andere Dienste verfügen muss:

1. `GetAuthorizationToken`benötigt `sts:GetServiceBearerToken` zusammen mit`codeartifact:GetAuthorizationToken`.
2. `CreateDomain`, erfordert bei der Bereitstellung eines nicht standardmäßigen Verschlüsselungsschlüssels `kms:DescribeKey` sowohl als auch `kms:CreateGrant` den KMS-Schlüssel zusammen mit`codeartifact:CreateDomain`.

Weitere Informationen zu den erforderlichen Berechtigungen und Ressourcen für Aktionen in finden Sie CodeArtifact unterAWS CodeArtifact Referenz zu Berechtigungen.

Servicerollen für CodeArtifact

Unterstützt Servicerollen: Nein

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann zu CodeArtifact Funktionseinschränkungen führen. Bearbeiten Sie Servicerollen nur, CodeArtifact wenn Sie dazu eine Anleitung erhalten.

Dienstbezogene Rollen für CodeArtifact

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der

Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für AWS CodeArtifact

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, CodeArtifact-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von definiert wurden CodeArtifact, einschließlich des Formats von ARNs für jeden der Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS CodeArtifact in der Service Authorization Reference.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der CodeArtifact-Konsole
- AWS-verwaltete (vordefinierte) Richtlinien für AWS CodeArtifact
- Erlauben Sie einem Benutzer, seine eigenen Berechtigungen einzusehen
- Erlaubt einem Benutzer, Informationen über Repositorys und Domänen abzurufen
- Erlauben Sie einem Benutzer, Informationen zu bestimmten Domänen abzurufen
- Erlaubt einem Benutzer, Informationen über bestimmte Repositorys abzurufen
- Die Dauer des Autorisierungstokens

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand CodeArtifact Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der CodeArtifact-Konsole

Um auf die AWS CodeArtifact Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den CodeArtifact Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die CodeArtifact Konsole weiterhin verwenden können, fügen Sie den Entitäten auch die `AWSCodeArtifactAdminAccess` oder die `AWSCodeArtifactReadOnlyAccess` AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

AWS-verwaltete (vordefinierte) Richtlinien für AWS CodeArtifact

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM-Richtlinien, die von erstellt und verwaltet AWS werden. Diese AWS verwalteten Richtlinien gewähren die erforderlichen Berechtigungen für allgemeine Anwendungsfälle, sodass Sie nicht erst untersuchen müssen, welche Berechtigungen benötigt werden. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien im IAM Benutzerhandbuch.

Die folgenden AWS verwalteten Richtlinien, die Sie Benutzern in Ihrem Konto zuordnen können, sind spezifisch für AWS CodeArtifact.

- `AWSCodeArtifactAdminAccess`— Bietet vollen Zugriff, CodeArtifact einschließlich der Berechtigungen zur Verwaltung von CodeArtifact Domänen.

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

```
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

- `AWSCodeArtifactReadOnlyAccess`— Bietet schreibgeschützten Zugriff auf. CodeArtifact

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

Um CodeArtifact Servicerollen zu erstellen und zu verwalten, müssen Sie auch die AWS verwaltete Richtlinie mit dem Namen anhängen. `IAMFullAccess`

Sie können auch Ihre eigenen, benutzerdefinierten IAM-Richtlinien erstellen, um Berechtigungen für CodeArtifact-Aktionen und -Ressourcen zu gewähren. Die benutzerdefinierten Richtlinien können Sie dann den IAM-Benutzern oder -Gruppen zuweisen, die diese Berechtigungen benötigen.

Erlauben Sie einem Benutzer, seine eigenen Berechtigungen einzusehen

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI AWS OR-API.

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

Erlaubt einem Benutzer, Informationen über Repositorys und Domänen abzurufen

Die folgende Richtlinie ermöglicht es einem IAM-Benutzer oder einer IAM-Rolle, jede Art von CodeArtifact Ressource aufzulisten und zu beschreiben, einschließlich Domänen, Repositorys, Pakete und Ressourcen. Die Richtlinie beinhaltet auch die `codeArtifact:ReadFromRepository` Berechtigung, die es dem Prinzipal ermöglicht, Pakete aus einem Repository abzurufen. CodeArtifact Es erlaubt nicht, neue Domänen oder Repositorys zu erstellen und neue Pakete zu veröffentlichen.

Die `sts:GetServiceBearerToken` Berechtigungen `codeartifact:GetAuthorizationToken` und sind erforderlich, um die `GetAuthorizationToken` API aufzurufen.

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

Erlauben Sie einem Benutzer, Informationen zu bestimmten Domänen abzurufen

Im Folgenden finden Sie ein Beispiel für eine Berechtigungsrichtlinie, die es einem Benutzer ermöglicht, nur Domänen in der `us-east-2` Region für Konten `123456789012` für jede Domäne aufzulisten, die mit dem Namen beginnen`my`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codeartifact:ListDomains",
         "Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/my*"
      }
   ]
}
```

Erlaubt einem Benutzer, Informationen über bestimmte Repositorys abzurufen

Im Folgenden finden Sie ein Beispiel für eine Berechtigungsrichtlinie, die es einem Benutzer ermöglicht, Informationen über Repositorys abzurufen, die mit enden`test`, einschließlich Informationen über die darin enthaltenen Pakete. Der Benutzer kann keine Ressourcen veröffentlichen, erstellen oder löschen.

Die `sts:GetServiceBearerToken` Berechtigungen `codeartifact:GetAuthorizationToken` und sind erforderlich, um die `GetAuthorizationToken` API aufzurufen.

```
{
   "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*",
               "codeartifact:Describe*"
            ],
            "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": "*"
         }
      ]
}
```

Die Dauer des Autorisierungstokens

Benutzer müssen sich CodeArtifact mit Autorisierungstoken authentifizieren, um Paketversionen zu veröffentlichen oder zu nutzen. Autorisierungstoken sind nur während ihrer konfigurierten Lebensdauer gültig. Token haben eine Standardlebensdauer von 12 Stunden. Weitere Informationen zu Autorisierungstoken finden Sie unterAWS CodeArtifact Authentifizierung und Tokens.

Beim Abrufen eines Tokens können Benutzer die Lebensdauer des Tokens konfigurieren. Gültige Werte für die Lebensdauer eines Autorisierungstokens sind `0` und eine beliebige Zahl zwischen `900` (15 Minuten) und `43200` (12 Stunden). Mit dem Wert von `0` wird ein Token mit einer Dauer erstellt, die den temporären Anmeldeinformationen der Benutzerrolle entspricht.

Administratoren können die gültigen Werte für die Gültigkeitsdauer eines Autorisierungstokens einschränken, indem sie den `sts:DurationSeconds` Bedingungsschlüssel in der Berechtigungsrichtlinie verwenden, die dem Benutzer oder der Gruppe zugewiesen ist. Wenn der Benutzer versucht, ein Autorisierungstoken mit einer Gültigkeitsdauer außerhalb der gültigen Werte zu erstellen, schlägt die Tokenerstellung fehl.

Die folgenden Beispielrichtlinien begrenzen die mögliche Dauer eines von CodeArtifact Benutzern erstellten Autorisierungstokens.

Beispielrichtlinie: Beschränken Sie die Lebensdauer eines Tokens auf genau 12 Stunden (43200 Sekunden)

Mit dieser Richtlinie können Benutzer nur Autorisierungstoken mit einer Lebensdauer von 12 Stunden erstellen.

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

Beispielrichtlinie: Beschränken Sie die Gültigkeitsdauer von Token auf 15 Minuten bis 1 Stunde oder entspricht dem Zeitraum der temporären Anmeldeinformationen des Benutzers

Mit dieser Richtlinie können Benutzer Token erstellen, die zwischen 15 Minuten und 1 Stunde gültig sind. Benutzer können auch ein Token erstellen, das für die Dauer der temporären Anmeldeinformationen ihrer Rolle gültig ist, indem sie `0` für angeben`--durationSeconds`.

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

Verwenden von Tags zur Steuerung des Zugriffs auf CodeArtifact-Ressourcen

Bedingungen in IAM-Benutzerrichtlinienanweisungen sind Teil der Syntax, mit der Sie Berechtigungen für Ressourcen angeben, die für Aktionen erforderlich sind. CodeArtifact Das Verwenden von Tags in Bedingungen ist eine Möglichkeit zur Kontrolle des Zugriffs auf Ressourcen und Anfragen. Informationen zum Markieren von CodeArtifact Ressourcen finden Sie unter. Taggen von -Ressourcen Dieses Thema behandelt die Tag-basierte Zugriffskontrolle.

Wenn Sie IAM-Richtlinien entwerfen, können Sie präzise abgestufte Berechtigungen festlegen, indem Sie Zugriff auf bestimmte Ressourcen gewähren. Mit zunehmender Anzahl der Ressourcen, die Sie verwalten, wird diese Aufgabe erschwert. Durch Markieren von Ressourcen und Verwenden von Tags in Richtlinienanweisungsbedingungen lässt sich diese Aufgabe vereinfachen. Sie erteilen

Massenzugriff auf eine beliebige Ressource mit einem bestimmten Tag. Anschließend wenden Sie dieses Tag während der Erstellung oder zu einem späteren Zeitpunkt wiederholt auf relevante Ressourcen an.

Markierungen können an die Ressource angehängt oder in der Anfrage an Services übergeben werden, die das Markieren unterstützen. In CodeArtifact können Ressourcen Tags haben, und einige Aktionen können Tags enthalten. Wenn Sie eine IAM-Richtlinie erstellen, können Sie Tag-Bedingungsschlüssel verwenden, um Folgendes zu kontrollieren:

- Welche Benutzer auf der Grundlage von bereits vorhandenen Tags Aktionen für eine Domain oder Repository-Ressource ausführen können.
- Welche Tags in der Anforderung einer Aktion übergeben werden können.
- Ob bestimmte Tag-Schlüssel in einer Anforderung verwendet werden können.

Die vollständige Syntax und Semantik der Tag-Bedingungsschlüssel finden Sie unter Steuern des Zugriffs mit Tags im IAM-Benutzerhandbuch.

> **Important**
>
> Wenn Sie Tags für Ressourcen verwenden, um Aktionen einzuschränken, müssen sich die Tags auf der Ressource befinden, auf der die Aktion ausgeführt wird. Um beispielsweise `DescribeRepository` Berechtigungen mit Tags zu verweigern, müssen sich die Tags in jedem Repository und nicht in der Domain befinden. Eine Liste der Aktionen CodeArtifact und der Ressourcen, auf denen sie ausgeführt werden, finden AWS CodeArtifact Referenz zu Berechtigungen Sie unter.

Beispiele für Tag-basierte Zugriffskontrolle

Die folgenden Beispiele zeigen, wie Sie Tag-Bedingungen in Richtlinien für CodeArtifact -Benutzer festlegen.

Example 1: Einschränken von Aktionen basierend auf Tags in der Anforderung

Die Richtlinie für `AWSCodeArtifactAdminAccess` verwaltete Benutzer gibt Benutzern uneingeschränkte Rechte, jede CodeArtifact Aktion auf einer beliebigen Ressource auszuführen.

Die folgende Richtlinie schränkt diese Befugnis ein und verweigert nicht autorisierten Benutzern die Erlaubnis, Repositorys zu erstellen, sofern die Anfrage nicht bestimmte Tags enthält. Zu diesem

Zweck wird die `CreateRepository` Aktion verweigert, wenn in der Anfrage kein Tag angegeben ist, das `costcenter` mit einem der Werte oder benannt ist. 1 2 Der Administrator eines Kunden muss diese IAM-Richtlinie nicht autorisierten IAM-Benutzern hinzufügen, zusätzlich zu der verwalteten Benutzerrichtlinie.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codeartifact:CreateRepository",
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/costcenter": "true"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": "codeartifact:CreateRepository",
      "Resource": "*",
      "Condition": {
          "ForAnyValue:StringNotEquals": {
              "aws:RequestTag/costcenter": [
                  "1",
                  "2"
              ]
          }
      }
  }
  ]
}
```

Example 2: Einschränken von Aktionen basierend auf Ressourcen-Tags

Die Richtlinie für `AWSCodeArtifactAdminAccess` verwaltete Benutzer gibt Benutzern uneingeschränkte Rechte, jede CodeArtifact Aktion auf einer beliebigen Ressource auszuführen.

Die folgende Richtlinie schränkt diese Befugnis ein und verweigert nicht autorisierten Benutzern die Erlaubnis, Aktionen an Repositorys in bestimmten Domänen durchzuführen. Hierfür verweigert sie einige Aktionen, wenn die Ressource über ein Tag mit dem Namen `Key1` und einem der Werte

`Value1` oder `Value2` verfügt. (Der Bedingungsschlüssel `aws:ResourceTag` wird verwendet, um den Zugriff auf die Ressourcen auf der Grundlage der Tags dieser Ressourcen zu steuern.) Der Administrator eines Kunden muss diese IAM-Richtlinie nicht autorisierten IAM-Benutzern hinzufügen, zusätzlich zu der verwalteten Benutzerrichtlinie.

```
{
  "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: Erlaube Aktionen, die auf Ressourcen-Tags basieren

Die folgende Richtlinie gewährt Benutzern die Erlaubnis, Aktionen an Repositorys und Paketen durchzuführen und Informationen darüber abzurufen. CodeArtifact

Zu diesem Zweck sind bestimmte Aktionen zulässig, wenn das Repository über ein Tag verfügt, das `Key1` mit diesem Wert `Value1` benannt ist. (Der Bedingungsschlüssel `aws:RequestTag` wird verwendet, um zu steuern, welche Tags in einer IAM-Anforderung übergeben werden können.) Die Bedingung `aws:TagKeys` stellt sicher, dass bei Tag-Schlüsseln die Groß-

und Kleinschreibung beachtet wird. Diese Richtlinie ist nützlich für IAM-Benutzer, denen die `AWSCodeArtifactAdminAccess`-Richtlinie für verwaltete Benutzer nicht angefügt ist. Die verwaltete Richtlinie gibt Benutzern uneingeschränkte Rechte, jede CodeArtifact Aktion auf einer beliebigen Ressource auszuführen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact:ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

Example 4: Erlaube Aktionen, die auf Tags in der Anfrage basieren

Die folgende Richtlinie gewährt Benutzern die Erlaubnis, Repositorys in bestimmten Domänen in CodeArtifact zu erstellen.

Zu diesem Zweck sind die `TagResource` Aktionen `CreateRepository` und zulässig, wenn die Create-Resource-API in der Anfrage ein Tag `Key1` mit dem Namen des Werts `Value1` angibt. (Der Bedingungsschlüssel `aws:RequestTag` wird verwendet, um zu steuern, welche Tags in einer IAM-Anforderung übergeben werden können.) Die Bedingung `aws:TagKeys` stellt sicher, dass bei Tag-Schlüsseln die Groß- und Kleinschreibung beachtet wird. Diese Richtlinie ist nützlich für IAM-Benutzer, denen die `AWSCodeArtifactAdminAccess`-Richtlinie für verwaltete Benutzer nicht angefügt ist. Die verwaltete Richtlinie gibt Benutzern uneingeschränkte Rechte, jede CodeArtifact Aktion auf einer beliebigen Ressource auszuführen.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:CreateRepository",
        "codeartifact:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

AWS CodeArtifact Referenz zu Berechtigungen

AWS CodeArtifact Ressourcen und Operationen

In AWS CodeArtifact, die primäre Ressource ist eine Domain. In einer Richtlinie identifizieren Sie die Ressource, für welche die Richtlinie gilt, mithilfe eines Amazon-Ressourcennamens (ARN). Repositorien sind ebenfalls Ressourcen und wurden mit ihnen ARNs verknüpft. Weitere Informationen finden Sie unter Amazon Resource Names (ARNs) in der Allgemeine Amazon Web Services-Referenz.

Ressourcentyp	ARN-Format
Domain	`arn:aws:codeartifact:` *region-ID* `:`*account-ID*`:domain/`*my_domain*
Repository	`arn:aws:codeartifact:` *region-ID* `:`*account-ID*`:repository/` *my_domain* `/`*my_repo*
Paketgruppe	`arn:aws:codeartifact:` *region-ID* `:`*account-ID*`:package-group/` *my_domain* `/`*encoded_package_group_pattern*

Ressourcentyp	ARN-Format
Package mit einem Namespace	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:package/` *my_domain* `/`*my_repo*`/`*package-format* `/`*namespace* `/`*my_package*
Package ohne Namespace	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:package/` *my_domain* `/`*my_repo*`/`*package-format* `//`*my_package*
Alle Ressourcen CodeArtifact	`arn:aws:codeartifact:*`
Alle CodeArtifact Ressourcen, die dem angegebenen Konto in der angegebenen AWS-Region gehören	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:*`

Welchen Ressourcen-ARN Sie angeben, hängt davon ab, auf welche Aktion oder Aktionen Sie den Zugriff kontrollieren möchten.

Sie können in Ihrer Anweisung eine bestimmte Domain (*myDomain*) mit ihrem ARN wie folgt angeben.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

Sie können in Ihrer Anweisung ein bestimmtes Repository (*myRepo*) mit seinem ARN wie folgt angeben.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie sie ARNs durch Kommas. Die folgende Aussage gilt für alle Pakete und Repositorys in einer bestimmten Domäne.

```
"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**
>
> Viele AWS Dienste behandeln einen Doppelpunkt (:) oder einen Schrägstrich (/) als dasselbe Zeichen in. ARNs CodeArtifact Verwendet jedoch eine exakte Übereinstimmung in den Ressourcenmustern und Regeln. Verwenden Sie also die richtigen Zeichen zum Erstellen von Ereignismustern, sodass sie mit der ARN-Syntax in der Ressource übereinstimmen.

AWS CodeArtifact API-Operationen und Berechtigungen

Sie können die folgende Tabelle als Referenz verwenden, wenn Sie Zugriffskontrolle einrichten und Berechtigungsrichtlinien schreiben, die Sie einer IAM-Identität zuordnen können (identitätsbasierte Richtlinien).

Sie können in Ihren AWS CodeArtifact Richtlinien AWS allgemeine Bedingungsschlüssel verwenden, um Bedingungen auszudrücken. Eine Liste finden Sie unter IAM JSON Policy Elements Reference im IAM-Benutzerhandbuch.

Sie geben die Aktionen im Feld `Action` der Richtlinie an. Um eine Aktion anzugeben, verwenden Sie das Präfix `codeartifact:` gefolgt vom Namen der API-Operation (z. B. `codeartifact:CreateDomain` und `codeartifact:AssociateExternalConnection`). Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Komma (z. B. `"Action": [ "codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection" ]`).

Verwenden von Platzhalterzeichen

Sie geben einen ARN mit oder ohne Platzhalterzeichen (*) als Ressourcenwert im Feld `Resource` der Richtlinie an. Sie können das Platzhalterzeichen verwenden, um mehrere Aktionen oder Ressourcen anzugeben. `codeartifact:*`Gibt beispielsweise alle Aktionen an und `codeartifact:Describe*` gibt alle CodeArtifact CodeArtifact Aktionen an, die mit dem Wort `Describe` beginnen.

Paketgruppe ARNs

> **Note**
>
> Dieser Abschnitt darüber, wie Paketgruppen ARNs und Muster kodiert werden, ist informativ. Es wird empfohlen, ARNs von der Konsole zu kopieren oder ARNs über die `DescribePackageGroup` API abzurufen, anstatt Muster zu kodieren und zu konstruieren. ARNs

IAM-Richtlinien verwenden das Platzhalterzeichen,`*`, um mehreren IAM-Aktionen oder mehreren Ressourcen zuzuordnen. Paketgruppenmuster verwenden das `*` Zeichen ebenfalls. Um IAM-Richtlinien, die einer einzelnen Paketgruppe entsprechen, einfacher schreiben zu können, verwendet das ARN-Format für Paketgruppen eine kodierte Version des Paketgruppenmusters.

Insbesondere lautet das ARN-Format der Paketgruppe wie folgt:

```
arn:aws:codeartifact:region:account-ID:package-
group/my_domain/encoded_package_group_pattern
```

Wobei das kodierte Paketgruppenmuster das Paketgruppenmuster ist, wobei bestimmte Sonderzeichen durch ihre prozentkodierten Werte ersetzt werden. Die folgende Liste enthält die Zeichen und ihre entsprechenden prozentkodierten Werte:

- `*` : `%2a`
- `$` : `%24`
- `%` : `%25`

Der ARN für eine Stammpaketgruppe einer Domain, (`/*`), wäre beispielsweise:

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

Beachten Sie, dass Zeichen, die nicht in der Liste enthalten ARNs sind, nicht kodiert werden können und dass Groß- und Kleinschreibung beachtet wird. Sie `*` müssen also als `%2a` und nicht codiert werden. `%2A`

Problembehandlung bei AWS CodeArtifact Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit CodeArtifact und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in CodeArtifact
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine CodeArtifact Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in CodeArtifact

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `codeartifact:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codeartifact:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `codeartifact:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine CodeArtifact Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen CodeArtifact unterstützt werden, finden Sie unter. Wie AWS CodeArtifact funktioniert mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Arbeiten mit Amazon VPC-Endpunkten

Sie können CodeArtifact die Verwendung eines VPC-Endpunkts (Virtual Private Cloud) mit Schnittstelle konfigurieren, um die Sicherheit Ihrer VPC zu verbessern.

VPC-Endpunkte verwenden AWS PrivateLink einen Dienst, der Ihnen den Zugriff CodeArtifact APIs über private IP-Adressen ermöglicht. AWS PrivateLink schränkt den gesamten Netzwerkverkehr zwischen Ihrer VPC und CodeArtifact dem AWS-Netzwerk ein. Wenn Sie einen VPC-Endpunkt mit Schnittstelle verwenden, benötigen Sie kein Internet-Gateway, kein NAT-Gerät oder kein Virtual Private Gateway. Weitere Informationen finden Sie unter VPC-Endpunkte im Amazon Virtual Private Cloud-Benutzerhandbuch.

Important

- VPC-Endpunkte unterstützen keine regionsübergreifenden Anfragen AWS . Stellen Sie sicher, dass Sie Ihren Endpunkt in derselben AWS Region erstellen, an die Sie Ihre API-Aufrufe richten möchten. CodeArtifact
- VPC-Endpunkte unterstützen nur von Amazon bereitgestellten DNS über Amazon Route 53. Wenn Sie Ihre eigene DNS verwenden möchten, können Sie die bedingte DNS-Weiterleitung nutzen. Weitere Informationen finden Sie unter DHCP-Optionssätze im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.
- Die Sicherheitsgruppe für den VPC-Endpunkt müssen eingehende Verbindungen auf Port 443 aus dem privaten Subnetz der VPC zulassen.

Themen

- VPC-Endpoints erstellen für CodeArtifact
- Erstellen Sie den Amazon S3-Gateway-Endpunkt
- CodeArtifact Von einer VPC aus verwenden
- Erstellen einer VPC-Endpunktrichtlinie für CodeArtifact

VPC-Endpoints erstellen für CodeArtifact

Verwenden Sie den EC2 `create-vpc-endpoint` AWS CLI Amazon-Befehl, um Virtual Private Cloud (VPC) -Endpunkte für CodeArtifact zu erstellen. Weitere Informationen finden

Sie unter Interface VPC Endpoints (AWS PrivateLink) im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Zwei VPC-Endpunkte sind erforderlich, damit sich alle Anfragen im CodeArtifact AWS Netzwerk befinden. Der erste Endpunkt wird zum Aufrufen verwendet CodeArtifact APIs (z. B. `GetAuthorizationToken` und`CreateRepository`).

```
com.amazonaws.region.codeartifact.api
```

Der zweite Endpunkt wird verwendet, um mithilfe von Paketmanagern und Build-Tools (z. B. npm und Gradle) auf CodeArtifact Repositorys zuzugreifen.

```
com.amazonaws.region.codeartifact.repositories
```

Der folgende Befehl erstellt einen Endpunkt für den Zugriff auf Repositorys. CodeArtifact

```
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
```

Der folgende Befehl erstellt einen Endpunkt für den Zugriff auf Paketmanager und Build-Tools.

```
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**
>
> Wenn Sie einen `codeartifact.repositories` Endpunkt erstellen, müssen Sie mithilfe der `--private-dns-enabled` Option einen privaten DNS-Hostnamen erstellen. Wenn Sie beim Erstellen des `codeartifact.repositories` Endpunkts keinen privaten DNS-Hostnamen erstellen können oder möchten, müssen Sie einen zusätzlichen Konfigurationsschritt ausführen, um Ihren Paketmanager CodeArtifact von einer VPC aus zu verwenden. Weitere Informationen finden Sie unter Verwenden Sie den `codeartifact.repositories` Endpunkt ohne privates DNS.

Nachdem Sie VPC-Endpoints erstellt haben, müssen Sie möglicherweise weitere Konfigurationen mit Sicherheitsgruppenregeln vornehmen, mit denen Sie die Endpoints verwenden können. CodeArtifact Weitere Informationen zu Sicherheitsgruppen in Amazon VPC finden Sie unter Sicherheitsgruppen.

Wenn Sie Probleme mit der Verbindung haben CodeArtifact, können Sie das VPC Reachability Analyzer-Tool verwenden, um das Problem zu debuggen. Weitere Informationen finden Sie unter Was ist VPC Reachability Analyzer?

Erstellen Sie den Amazon S3-Gateway-Endpunkt

CodeArtifact verwendet Amazon Simple Storage Service (Amazon S3) zum Speichern von Paketressourcen. Um Pakete abzurufen CodeArtifact, müssen Sie einen Gateway-Endpunkt für Amazon S3 erstellen. Wenn Ihr Build- oder Bereitstellungsprozess Pakete herunterlädt CodeArtifact, muss er CodeArtifact auf Paketmetadaten zugreifen und auf Amazon S3 zugreifen, um Paketressourcen (z. B. `.jar` Maven-Dateien) herunterzuladen.

> **Note**
>
> Ein Amazon S3 S3-Endpunkt ist nicht erforderlich, wenn Python- oder Swift-Paketformate verwendet werden.

Verwenden Sie den EC2 `create-vpc-endpoint` AWS CLI Amazon-Befehl CodeArtifact, um den Amazon S3-Gateway-Endpunkt für zu erstellen. Wenn Sie den Endpunkt erstellen, müssen Sie die Routing-Tabellen für Ihre VPC auswählen. Weitere Informationen finden Sie unter Gateway VPC Endpoints im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Der folgende Befehl erstellt einen Amazon S3 S3-Endpunkt.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
  --route-table-ids routetableid
```

Mindestberechtigungen für Amazon S3 S3-Buckets für AWS CodeArtifact

Der Amazon S3-Gateway-Endpunkt verwendet ein IAM-Richtliniendokument, um den Zugriff auf den Dienst zu beschränken. Um nur die Mindestberechtigungen für Amazon S3 S3-Bucket zuzulassen CodeArtifact, beschränken Sie den Zugriff auf den Amazon S3 S3-Bucket, der bei der Erstellung des IAM-Richtliniendokuments für den Endpunkt CodeArtifact verwendet wird.

In der folgenden Tabelle werden die Amazon S3 S3-Buckets beschrieben, auf die Sie in Ihren Richtlinien verweisen sollten, um den Zugriff CodeArtifact in jeder Region zu ermöglichen.

Region	Amazon S3 S3-Bucket ARN
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

Sie können den `aws codeartifact describe-domain` Befehl verwenden, um den Amazon S3 S3-Bucket abzurufen, der von einer CodeArtifact Domain verwendet wird.

```
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"
  }
}
```

Beispiel

Das folgende Beispiel zeigt, wie Sie Zugriff auf die Amazon S3 S3-Buckets gewähren, die für den CodeArtifact Betrieb in der `us-east-1` Region erforderlich sind. Für andere Regionen aktualisieren Sie den `Resource` Eintrag anhand der obigen Tabelle mit dem richtigen Berechtigungs-ARN für Ihre Region.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]
    }
  ]
}
```

CodeArtifact Von einer VPC aus verwenden

Wenn Sie privates DNS auf Ihrem `com.amazonaws.region.codeartifact.repositories` VPC-Endpunkt, in dem Sie ihn erstellt haben, nicht aktivieren können oder möchtenVPC-Endpoints

erstellen für CodeArtifact, müssen Sie eine andere Konfiguration für den Repository-Endpunkt verwenden, um ihn CodeArtifact von einer VPC aus zu verwenden. Folgen Sie den Anweisungen unterVerwenden Sie den `codeartifact.repositories` Endpunkt ohne privates DNS, um zu konfigurieren, CodeArtifact ob für den `com.amazonaws.region.codeartifact.repositories` Endpunkt privates DNS nicht aktiviert ist.

Verwenden Sie den **`codeartifact.repositories`** Endpunkt ohne privates DNS

Wenn Sie privates DNS auf Ihrem `com.amazonaws.region.codeartifact.repositories` VPC-Endpunkt, in dem Sie ihn erstellt haben, nicht aktivieren können oder wollenVPC-Endpoints erstellen für CodeArtifact, müssen Sie diese Anweisungen befolgen, um Ihren Paketmanager mit der richtigen CodeArtifact URL zu konfigurieren.

1. Führen Sie den folgenden Befehl aus, um einen VPC-Endpunkt zu finden, der zum Überschreiben des Hostnamens verwendet werden soll.

   ```
   $ aws ec2 describe-vpc-endpoints --filters Name=service-
   name,Values=com.amazonaws.region.codeartifact.repositories \
     --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
   ```

   Die Ausgabe sieht wie folgt aus.

   ```
   [
     [
       "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"
     ]
   ]
   ```

2. Aktualisieren Sie den VPC-Endpunktpfad so, dass er das Paketformat, Ihren CodeArtifact Domainnamen und den CodeArtifact Repository-Namen enthält. Sehen Sie sich das folgende Beispiel an.

   ```
   https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-
   west-2.vpce.amazonaws.com/format/d/domain_name-domain_owner/repo_name
   ```

   Ersetzen Sie die folgenden Felder aus dem Beispielendpunkt.

   - *`format`*: Ersetzen Sie es durch ein gültiges CodeArtifact Paketformat, z. B. `npm` oder`pypi`.

- *domain_name*: Ersetze es durch die CodeArtifact Domain, die das CodeArtifact Repository enthält, das deine Pakete hostet.
- *domain_owner*: Ersetze es durch die ID des Besitzers der CodeArtifact Domain, zum Beispiel111122223333.
- *repo_name*: Ersetze es durch das CodeArtifact Repository, das deine Pakete hostet.

Die folgende URL ist ein Beispiel für einen NPM-Repository-Endpunkt.

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. Konfigurieren Sie Ihren Paketmanager so, dass er den aktualisierten VPC-Endpunkt aus dem vorherigen Schritt verwendet. Sie müssen den Paketmanager konfigurieren, ohne den CodeArtifact `login` Befehl zu verwenden. Anweisungen zur Konfiguration der einzelnen Paketformate finden Sie in der folgenden Dokumentation.

   - npm: Konfiguration von npm ohne Verwendung des Login-Befehls
   - nuget: Konfiguriere Nuget oder Dotnet ohne den Login-Befehl
   - pip: Konfigurieren Sie pip ohne den Login-Befehl
   - Bindfaden: Konfigurieren und verwenden Sie Twine mit CodeArtifact
   - Gradel: CodeArtifact Mit Gradle verwenden
   - mvn: CodeArtifact Mit MVN verwenden

Erstellen einer VPC-Endpunktrichtlinie für CodeArtifact

Um eine VPC-Endpunktrichtlinie für zu erstellen CodeArtifact, geben Sie Folgendes an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Die folgende Beispielrichtlinie legt fest, dass Prinzipale im Konto 123456789012 die `GetAuthorizationToken` API aufrufen und Pakete aus einem Repository abrufen können. CodeArtifact

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
         "AWS": "arn:aws:iam::123456789012:root"
       }
    }
  ]
}
```

CodeArtifact Ressourcen erstellen mit AWS CloudFormation

CodeArtifact ist integriert in AWS CloudFormation, ein Service, der Ihnen hilft, Ihre AWS Ressourcen zu modellieren und einzurichten, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle AWS Ressourcen beschreibt, die Sie benötigen, und AWS CloudFormation kümmert sich um die Bereitstellung und Konfiguration dieser Ressourcen für Sie.

Wenn Sie sie verwenden AWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre CodeArtifact Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einfach einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren Konten und AWS Regionen bereit.

CodeArtifact und AWS CloudFormation Vorlagen

Um Ressourcen für und zugehörige Dienste bereitzustellen CodeArtifact und zu konfigurieren, müssen Sie sich mit AWS CloudFormation Vorlagen auskennen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern. AWS CloudFormation Weitere Informationen finden Sie unter Was ist AWS CloudFormation Designer? im AWS CloudFormation Benutzerhandbuch.

CodeArtifact unterstützt das Erstellen von Domänen, Repositorys und Paketgruppen in AWS CloudFormation. Weitere Informationen, einschließlich Beispielen für JSON- und YAML-Vorlagen, finden Sie in den folgenden Themen im AWS CloudFormation Benutzerhandbuch:

- AWS::CodeArtifact::Domain
- AWS::CodeArtifact::Repository
- AWS::CodeArtifact::PackageGroup

Das Löschen von CodeArtifact Ressourcen verhindern

CodeArtifact Repositorys enthalten wichtige Anwendungsabhängigkeiten, die möglicherweise nicht einfach wiederhergestellt werden können, wenn sie verloren gehen. Um CodeArtifact Ressourcen bei der Verwaltung von CodeArtifact Ressourcen vor versehentlichem Löschen zu schützen

CloudFormation, fügen Sie die `UpdateRetainPolicy` Attribute `DeletionPolicy` und mit dem Wert für alle Domänen und `Retain` Repositorys hinzu. Dadurch wird ein Löschen verhindert, wenn die Ressource aus der Stack-Vorlage entfernt wird oder der gesamte Stapel versehentlich gelöscht wird. Das folgende YAML-Snippet zeigt eine grundlegende Domain und ein Repository mit diesen Attributen:

```
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
```

Weitere Informationen zu diesen Attributen finden Sie unter DeletionPolicyund UpdateReplacePolicyim AWS CloudFormation Benutzerhandbuch.

Erfahren Sie mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- AWS CloudFormation
- AWS CloudFormation Benutzerhandbuch
- AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle

Problembehebung AWS CodeArtifact

Die folgenden Informationen können Ihnen helfen, häufig auftretende Probleme mit zu beheben CodeArtifact.

Informationen zur Behebung formatspezifischer Probleme finden Sie in den folgenden Themen:

- Fehlerbehebung in Maven
- Schnelle Problembehebung

Ich kann keine Benachrichtigungen sehen

Problem: Wenn Sie in der Entwicklertools-Konsole unter Notifications (Einstellungen) die Option Notifications (Benachrichtigungen) wählen, wird ein Berechtigungsfehler angezeigt.

Mögliche Korrekturen: Benachrichtigungen sind zwar ein Feature der Developer Tools-Konsole, CodeArtifact unterstützt aber derzeit keine Benachrichtigungen. Keine der verwalteten Richtlinien für CodeArtifact beinhaltet Berechtigungen, die es Benutzern ermöglichen, Benachrichtigungen einzusehen oder zu verwalten. Wenn Sie andere Dienste in der Developer Tools-Konsole verwenden und diese Dienste Benachrichtigungen unterstützen, enthalten die verwalteten Richtlinien für diese Dienste die Berechtigungen, die zum Anzeigen und Verwalten von Benachrichtigungen für diese Dienste erforderlich sind.

Taggen von -Ressourcen

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie einer AWS Ressource AWS zuweisen oder zuweisen. Jedes AWS Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment`, `Project` oder `Secret`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- einem optionalen Feld, das als Tag-Wert bezeichnet wird (z. B. `111122223333`, `Production` oder ein Team-Name). Ein nicht angegebener Tag-Wert entspricht einer leeren Zeichenfolge. Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß- und Kleinschreibung unterschieden.

Zusammen werden sie als Schlüssel-Wert-Paare bezeichnet.

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Viele AWS-Services unterstützen das Markieren mit Tags (kurz: Tagging). So können Ressourcen aus verschiedenen Services denselben Tag zuweisen, um anzugeben, dass die Ressourcen verbunden sind. Sie können beispielsweise einem Repository dasselbe Tag zuweisen, das Sie einem AWS CodeBuild Projekt zuweisen.

Tipps und bewährte Methoden zur Verwendung von Tags finden Sie im Whitepaper Best Practices for Tagging AWS Resources.

Sie können in CodeArtifact die folgenden Ressourcentypen mit Tags markieren:

- Kennzeichnen Sie ein Repository in CodeArtifact
- Kennzeichnen Sie eine Domain in CodeArtifact

Sie können die Konsole,, oder AWS SDKs verwenden AWS CLI CodeArtifact APIs, um:

- Fügen Sie einer Domain oder einem Repository Tags hinzu, wenn Sie es erstellen*.
- Fügen Sie Tags für eine Domain oder ein Repository hinzu, verwalten und entfernen Sie sie.

* Sie können einer Domain oder einem Repository keine Tags hinzufügen, wenn Sie es in der Konsole erstellen.

Neben der Identifizierung, Organisation und Nachverfolgung Ihrer Ressource mithilfe von Tags können Sie mithilfe von Tags in IAM-Richtlinien steuern, wer Ihre Ressource aufrufen und mit ihr

interagieren kann. Beispiele für Tag-basierte Zugriffsrichtlinien finden Sie unter Verwenden von Tags zur Steuerung des Zugriffs auf CodeArtifact-Ressourcen.

CodeArtifact Kostenzuweisung mit Tags

Sie können Tags verwenden, um sowohl Speicher- als auch Anforderungskosten zuzuweisen. CodeArtifact

Zuweisung der Datenspeicherkosten in CodeArtifact

Die Kosten für die Datenspeicherung sind an Domains gebunden. Um Ihre CodeArtifact Speicherkosten zuzuweisen, können Sie daher alle Tags verwenden, die auf Ihre Domains angewendet wurden. Informationen zum Hinzufügen von Tags zu Domains finden Sie unterKennzeichnen Sie eine Domain in CodeArtifact.

Zuordnung der Anforderungskosten in CodeArtifact

Der Großteil der Nutzung von Anfragen ist an Repositorys gebunden. Um die Kosten für Ihre CodeArtifact Anfragen zuzuordnen, können Sie daher alle Tags verwenden, die auf Ihre Repositorys angewendet wurden. Hinweise zum Hinzufügen von Tags zu Repositorys finden Sie unter. Kennzeichnen Sie ein Repository in CodeArtifact

Einige Anforderungstypen sind eher mit Domains als mit Repositorys verknüpft, sodass die Nutzung der Anfrage und die mit den Anfragen verbundenen Kosten den Tags in der Domain zugewiesen werden. Die beste Methode, um festzustellen, ob ein Anforderungstyp einer Domäne oder einem Repository zugeordnet ist, besteht darin, die in der AWS CodeArtifact Tabelle in der Service Authorization Reference definierten Aktionen zu verwenden. Suchen Sie den Anforderungstyp in der Spalte Aktionen und sehen Sie sich den Wert in der entsprechenden Spalte Ressourcentypen an. Wenn es sich bei dem Ressourcentyp um eine Domain handelt, werden Anfragen dieses Typs der Domain in Rechnung gestellt. Wenn es sich bei dem Ressourcentyp um ein Repository oder ein Paket handelt, werden Anfragen dieses Typs dem Repository in Rechnung gestellt. Bei einigen Aktionen werden beide Ressourcentypen angezeigt. Bei diesen Aktionen hängt die abgerechnete Ressource davon ab, welcher Wert in der Anfrage übergeben wird.

Kontingente in AWS CodeArtifact

In der folgenden Tabelle werden Ressourcenkontingente in beschrieben CodeArtifact. Die Ressourcenkontingente zusammen mit der Liste der Dienstendpunkte für CodeArtifact finden Sie unter AWS Dienstkontingente in der Allgemeine Amazon Web Services-Referenz.

Sie können eine Erhöhung der Servicekontingenten für die folgenden CodeArtifact Ressourcenkontingente beantragen. Weitere Informationen zur Beantragung einer Erhöhung der Service Quotas finden Sie unter AWS Servicekontingenten.

Name	Standard	Anpas	Beschreibung
Größe der Asset-Datei	Jede unterstützte Region: 5 Gigabyte	Ja	Die maximale Dateigröße pro Asset.
Ressourcen pro Paketversion	Jede unterstützte Region: 150	Nein	Die maximale Anzahl von Assets pro Paketversion.
CopyPackageVersions Anfragen pro Sekunde	Jede unterstützte Region: 5	Yes (Ja)	Die maximale Anzahl von Anrufen, die CopyPackageVersions pro Sekunde getätigt werden können.
Direkte Upstreams pro Repository	Jede unterstützte Region: 10	Nein	Die maximale Anzahl direkter Upstream-Repositorys pro Repository.
Domains pro Konto AWS	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von Domains, die pro AWS Konto erstellt werden können.
GetAuthorizationToken Anfragen pro Sekunde	Jede unterstützte Region: 40	Ja	Die maximale Anzahl der pro Sekunde abgerufenen Autorisierungstoken.

Name	Standard	Anpas	Beschreibung
GetPackageVersionAsset Anfragen pro Sekunde	Jede unterstützte Region: 50	Ja	Die maximale Anzahl von Anrufen, die GetPackageVersionAsset pro Sekunde getätigt werden können.
ListPackageVersionAssets Anfragen pro Sekunde	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Anrufen, die ListPackageVersionAssets pro Sekunde getätigt werden können.
ListPackageVersions Anfragen pro Sekunde	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Anrufen, die ListPackageVersions pro Sekunde getätigt werden können.
ListPackages Anfragen pro Sekunde	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Anrufen, die ListPackages pro Sekunde getätigt werden können.
PublishPackageVersion Anfragen pro Sekunde	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von Anrufen, die PublishPackageVersion pro Sekunde getätigt werden können.
Lesen Sie Anfragen pro Sekunde von einem einzigen AWS Konto aus	Jede unterstützte Region: 800	Ja	Die maximale Anzahl von Leseanfragen von einem AWS Konto pro Sekunde.
Repositorien pro Domain	Jede unterstützte Region: 1 000	Ja	Die maximale Anzahl von Repositorys, die pro Domäne erstellt werden können.

Name	Standard	Anpas	Beschreibung
Anfragen pro Sekunde mit einem einzigen Authentifizierungstoken	Jede unterstützte Region: 1 200	Nein	Die maximale Anzahl von Anforderungen pro Sekunde über einen einzigen Authentif izierungstoken.
Anfragen ohne Authentifizierungstoken pro IP-Adresse	Jede unterstützte Region: 600	Nein	Die maximale Anzahl der Anforderungen pro Sekunde ohne Authentif izierungstoken von einer einzelnen IP-Adresse.
Durchsuchte Upstream-Repositorys	Jede unterstützte Region: 25	Nein	Die maximale Anzahl von Upstream-Repositor ys, die beim Auflösen eines Pakets durchsucht wurden.
Schreiben Sie Anfragen pro Sekunde von einem einzigen AWS Konto aus	Jede unterstützte Region: 100	Yes (Ja)	Die maximale Anzahl von Schreibanforderungen von einem AWS Konto pro Sekunde.

Note

Im Allgemeinen wird jede Leseanforderung an CodeArtifact als eine Anforderung gezählt, die auf ein Kontingent angerechnet wird. Beim Ruby-Paketformat kann eine einzelne Leseanforderung für den `/api/v1/dependencies` Vorgang jedoch Daten über mehrere Pakete anfordern.
Die Anfrage kann beispielsweise so aussehen`https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1,gem2.gem3`. In diesem Beispiel zählt die Anfrage als drei Anfragen im Verhältnis zum Kontingent.

Beachten Sie, dass die Mehrfachanfragen nur für Servicekontingenten gelten, nicht für die Abrechnung. In diesem Beispiel wird Ihnen nur eine Anfrage in Rechnung gestellt, obwohl diese als drei Anfragen auf das Servicekontingent angerechnet wird.

AWS CodeArtifact Benutzerleitfaden, Dokumentenverlauf

In der folgenden Tabelle werden wichtige Änderungen an der Dokumentation für beschrieben CodeArtifact.

Änderung	Beschreibung	Datum
Dokumentation für die Konfiguration und Verwendung von Cargo mit hinzugefügt CodeArtifact	CodeArtifact unterstützt jetzt Frachtkisten. Es wurde eine Dokumentation mit Anleitungen zur Konfiguration von Cargo für die Verwendung von CodeArtifact Repositorys hinzugefügt. Weitere Informationen finden Sie unter Verwendung CodeArtifact mit Cargo.	20. Juni 2024
Dokumentation zur Konfiguration und Verwendung von Ruby mit hinzugefügt CodeArtifact	CodeArtifact unterstützt jetzt Ruby Gems. Es wurde eine Dokumentation mit Anleitungen zur Konfiguration von Ruby-Paketmanagern für die Verwendung von CodeArtifact Repositorys hinzugefügt. Weitere Informationen finden Sie unter CodeArtifact Mit Ruby verwenden.	30. April 2024
Es wurde ein Beispiel für eine Schlüsselrichtlinie für die Erstellung von Domains mit einem vom Kunden verwalteten AWS KMS Schlüssel hinzugefügt	Es wurde ein Beispiel für eine Schlüsselrichtlinie hinzugefügt, mit der ein vom Kunden verwalteter KMS-Schlüssel zum Verschlüsseln von Ressourcen in CodeArtifact Domänen erstellt werden kann. Weitere Informationen	18. April 2024

	finden Sie unter Beispiel für eine AWS KMS Schlüssel richtlinie.	
Dokumentation zur Unterstützung des Starts von Paketgruppen hinzugefügt.	Dokumentation zur Verwaltung und Verwendung von Paketgruppen in hinzugefügt CodeArtifact. Weitere Informationen finden Sie unter Arbeiten mit Paketgruppen in CodeArtifact.	21. März 2024
Der Dokumentation zum aws-Codeartifact-Login-Befehl wurden weitere gültige Paketmanager hinzugefügt.	`dotnetnuget`, und `swift` zur Liste der gültigen Paketmanager hinzugefügt, die mit dem `aws codeartifact login` Befehl verwendet werden können. Weitere Informationen finden Sie unter AWS CodeArtifact Authentifizierung und Tokens.	18. Februar 2024
Der Swift-Dokumentation zur Fehlerbehebung wurde ein Eintrag hinzugefügt, in dem es darum geht, dass Xcode auf CI-Maschinen hängenbleibt	Es wurden Informationen, einschließlich einer Lösung, zu einem Problem hinzugefügt, das dazu führen kann, dass Xcode auf CI-Computern hängen bleibt, weil der Schlüsselbund zur Eingabe eines Kennworts aufgefordert wird. Weitere Informationen finden Sie unter Xcode hängt auf dem CI-Computer, da der Schlüsselbund zur Eingabe des Passworts aufgefordert wird.	6. Februar 2024

Es wurden Informationen zur Behebung langsamer Installationszeiten von NPM-Paketen mit npm 8.x oder höher hinzugefügt	Es wurden Informationen zur Umgehung langsamer Installationszeiten von npm-Paketen hinzugefügt CodeArtifact, die zu langsamen Build-Zeiten führen könnten. Weitere Informationen finden Sie unter Fehlerbehebung bei langsamen Installationen mit npm 8.x oder höher.	29. Dezember 2023
Aktualisierte Informationen zum Verhalten von Python-Paketen und Metadaten in CodeArtifact	Aktualisierte Informationen darüber, wie CodeArtifact Repositorys Ressourcen und Metadaten der Python-Paketversion speichern und aktualisieren. Weitere Informationen finden Sie unter Python-Pakete von Upstreams und externen Verbindungen anfordern.	14. Dezember 2023
Die Dokumentation zur Überwachung wurde neu organisiert CodeArtifact	Die Informationen zur Überwachung von CodeArtifact Ereignissen wurden neu organisiert und Informationen zur Anzeige von CodeArtifact Anfragen mit CloudWatch Amazon-Metriken hinzugefügt. Weitere Informationen finden Sie unter Überwachung CodeArtifact.	14. Dezember 2023

Es wurden weitere Informationen zur Verwaltung von CodeArtifact Ressourcen hinzugefügt mit AWS CloudFormation	Es wurden Verweise und Links zur Dokumentation zur Verwaltung von CodeArtifact Ressourcen mit hinzugefügt CloudFormation, einschließlich eines Abschnitts zur Verhinderung des Löschens von CodeArtifact Ressourcen, die mit verwaltet werden CloudFormation. Weitere Informationen finden Sie unter Das Löschen von CodeArtifact Ressourcen verhindern.	07. Dezember 2023
Es wurde eine Dokumentation hinzugefügt, in der die Unterstützung CodeArtifact von AWS KMS External Key Stores (XKS) detailliert beschrieben wird	Es wurde ein Abschnitt mit Informationen zur Unterstützung CodeArtifact von KMS-Schlüsseln hinzugefügt, einschließlich der Verwendung von XKS-Schlüsseln mit. CodeArtifact Weitere Informationen finden Sie unter Arten von AWS KMS Schlüsseln, die unterstützt werden in CodeArtifact.	31. Oktober 2023
Die bestehende Dokumentation zur Fehlerbehebung wurde aktualisiert und neue hinzugefügt	Es wurde ein Thema zur Fehlerbehebung in Maven hinzugefügt und Links zur Dokumentation zur Fehlerbehebung bei Swift und Maven in das allgemeine Thema zur Fehlerbehebung aufgenommen. Weitere Informationen finden Sie unter Problembehebung AWS CodeArtifact.	28. September 2023

Die Dokumentation wurde um den Swift Package Manager-Befehl publish aktualisiert	Swift 5.9 führte einen `swift package-registry publish` Befehl zum Erstellen und Veröffentlichen eines Swift-Pakets in einem Paket-Repository ein. Die Swift-Dokumentation wurde aktualisiert und enthält nun Anweisungen zur Verwendung dieses Befehls. Weitere Informationen finden Sie unter Verwendung CodeArtifact mit Swift.	25. September 2023
Dokumentation für die Konfiguration CodeArtifact mit Swift hinzugefügt	CodeArtifact unterstützt jetzt Swift-Pakete. Es wurde eine Dokumentation mit Anleitungen zur Konfiguration von Swift für die Verwendung von CodeArtifact Repositorys hinzugefügt. Weitere Informationen finden Sie unter Verwendung CodeArtifact mit Swift.	20. September 2023
Es wurde eine Anleitung zum CodeArtifact Umgang mit gezerrten Python-Paketversionen hinzugefügt	Es wurde eine Dokumentation mit Informationen darüber hinzugefügt, wie man erkennt, ob eine Python-Paketversion entfernt wurde, wie mit CodeArtifact gezogenen Paketversionen umgegangen wird, und Antworten auf häufig gestellte Fragen. Weitere Informationen finden Sie unter Paketversionen wurden entfernt.	02. August 2023

Ein falscher Befehlsze ilenbefehl in der Yarn-Doku mentation wurde behoben	Ein falscher Befehlsze ilenbefehl, der ein CodeArtif act Autorisierungstoken abruft und es in einer Umgebungs variablen in der Yarn-Doku mentation speichert, wurde behoben.	20. Juli 2023
Kleinere Ergänzungen und kleine Bugfixes an der Python-Dokumentation	Informationen zu Pip und Twine wurden in ihrer jeweilige n Dokumentation hinzugefügt und korrigiert, was passiert, wenn der `codeartifact login` Befehl mit Twine verwendet wird. Weitere Informationen erhalten Sie unter Konfigurieren und verwenden Sie Pip mit CodeArtifact und Konfigurieren und verwenden Sie Twine mit CodeArtifact.	14. Juli 2023
Falsche Dotnet-Befehle in der Dokumentation wurden behoben CodeBuild	Die `dotnet add package` Befehle in der Verwenden von Paketen NuGet in CodeBuild Dokumentation wurden korrigiert.	13. Juli 2023
Aktualisierung AWS CodeArtif act und AWS Identity and Access Management Dokumentation	Die CodeArtifact IAM-Dokum entation wurde überarbei tet, um für mehr Klarheit und Konsistenz mit der Dokumenta tion für andere Dienste zu sorgen. AWS Siehe Identity and Access Management für AWS CodeArtifact.	24. Mai 2023

Es wurden Informationen über gepanzerte Python-Paketversionen hinzugefügt	Es wurden Informationen darüber hinzugefügt, wie Metadaten der gelöschten Python-Paketversion CodeArtifact beibehalten werden. Weitere Informationen finden Sie unterPaketversionen wurden entfernt.	11. April 2023
Es wurden Informationen zur Clojure-Unterstützung hinzugefügt	Es wurden Informationen zur Clojure-Unterstützung hinzugefügt, einschließlich der Verwaltung von Abhängigkeiten für Clojure-Projekte. Weitere Informationen finden Sie unter CodeArtifact Mit deps.edn verwenden.	21. März 2023
Es wurden Informationen zur Veröffentlichung generischer Pakete hinzugefügt	Es wurden Informationen zu generischen Paketen und zum Veröffentlichen und Herunterladen von Paketinhalten mit dem hinzugefügt AWS CLI. Weitere Informationen finden Sie unter Verwendung CodeArtifact mit generischen Paketen, Generische Pakete veröffentlichen und konsumieren und Unterstützte Befehle für generische Pakete.	10. März 2023
Es wurden Informationen zu Größenbeschränkungen für Assets beim Veröffentlichen hinzugefügt	Der Paketveröffentlichung wurde ein Abschnitt hinzugefügt, in dem die Größenbeschränkungen für die Veröffentlichung von Assets erläutert werden.	21. Juni 2022

Die Dokumentation zu externen Verbindungen wurde überarbeitet	Die Dokumentation für externe Verbindungen wurde verschoben und neu organisiert, sodass sie sich auf das Endziel des Benutzers konzentriert, nämlich sein Projektarchiv mit öffentlichen CodeArtifact Paket-Repositorys zu verbinden. Außerdem wurden weitere Anleitungen und Informationen zu den verschiedenen Methoden zur Erreichung dieses Ziels hinzugefügt. Weitere Informationen finden Sie unter Ein CodeArtifact Repository mit einem öffentlichen Repository Connect.	9. Mai 2022
Die CodeArtifact Veranstaltungsinformationen für Amazon CloudWatch Events wurden aktualisiert	Dem `account` Feld wurden weitere Informationen hinzugefügt und das `repositoryAdministrator` Feld hinzugefügt. Weitere Informationen finden Sie unter CodeArtifact Veranstaltungsformat und Beispiel.	7. März 2022

Es wurden Konfigurationsanweisungen für die Verwendung CodeArtifact von einer VPC ohne privates DNS hinzugefügt	Wenn Sie privates DNS auf Ihrem `codeartifact.repositories` VPC-Endpunkt nicht aktivieren können oder wollen, müssen Sie eine andere Konfiguration für den Repository-Endpunkt verwenden, um ihn CodeArtifact von einer VPC aus zu verwenden. Weitere Informationen finden Sie unter Verwenden Sie den `codeartifact.repositories` Endpunkt ohne privates DNS.	8. Februar 2022
Eine ausführliche Dokumentation zur Aktualisierung des Status von Paketversionen wurde hinzugefügt	Die Dokumentation zum Versionsstatus des Aktualisierungspakets wurde um ein eigenes Thema erweitert. Es wurde eine Dokumentation zum Aktualisieren des Status einer Paketversion hinzugefügt, einschließlich erforderlicher IAM-Berechtigungen, AWS CLI Beispielbefehlen für verschiedene Szenarien und möglicher Fehler. Weitere Informationen finden Sie unter Aktualisiere den Status der Paketversion.	1. September 2021

Die Dokumentation zum Kopieren von Paketversionen wurde mit detaillierteren Informationen zu den Berechtigungen aktualisiert	Es wurden weitere Informationen zu den erforderlichen IAM- und ressourcenbasierten Richtlinienberechtigungen für den Aufruf des `aws codeartifact copy-package-versions` Befehls zum Kopieren von Paketversionen von einem Repository in ein anderes innerhalb derselben Domäne hinzugefügt. CodeArtifact Neben weiteren Informationen gibt es jetzt Beispiele für die erforderlichen ressourcenbasierten Richtlinien für das Quell- und Ziel-Repository. Weitere Informationen finden Sie unter Erforderliche IAM-Berechtigungen zum Kopieren von Paketen.	25. August 2021

Aktualisierte Dokumentation zum Ausführen eines Gradle-Builds in IntelliJ IDEA	Die Dokumentation zum Ausführen eines Gradle-Builds in IntelliJ IDEA wurde mit Schritten zur Konfiguration von Gradle zum Abrufen von Plugins aktualisiert. CodeArtifact Außerdem wurde eine Option hinzugefügt, um für jeden neuen Lauf ein neues CodeArtifact Autorisierungstoken mit einem Inline-Aufruf von zu erstellen. `aws codeartifact get-authorization-token` Weitere Informationen finden Sie unter Führen Sie einen Gradle-Build in IntelliJ IDEA aus.	23. August 2021
Dokumentation zur Konfiguration und Verwendung von Yarn mit hinzugefügt AWS CodeArtifact	Dokumentation zur Konfiguration und Verwendung von Yarn 1.X und Yarn 2.X zur Verwaltung von NPM-Paketen hinzugefügt. CodeArtifact Weitere Informationen finden Sie unter Konfiguriere und verwende Yarn mit CodeArtifact.	30. Juli 2021

AWS CodeArtifact unterstützt jetzt Pakete NuGet	CodeArtifact Benutzer können jetzt NuGet Pakete veröffentlichen und konsumieren. Dokumentation für die Konfiguration und Verwendung von Visual Studio und NuGet Befehlszeilentools wie `nuget` und `dotnet` mit CodeArtifact Repositorys hinzugefügt. Weitere Informationen finden Sie unter Verwenden CodeArtifact mit NuGet.	19. November 2020
Ressourcen taggen in AWS CodeArtifact	Dokumentation zum Taggen von Repositorys und Domains in hinzugefügt. AWS CodeArtifact Siehe Taggen von -Ressourcen.	30. Oktober 2020
CodeArtifact unterstützt jetzt AWS CloudFormation	CodeArtifact Benutzer können jetzt AWS CloudFormation Vorlagen verwenden, um CodeArtifact Repositorys und Domains zu erstellen. CodeArtifact Ressourcen erstellen mit AWS CloudFormationWeitere Informationen und Informationen zu den ersten Schritten finden Sie unter.	8. Oktober 2020

Fügen Sie Informationen zur Erstellung von Amazon S3 S3-Gateway-Endpunkten zur Verwendung CodeArtifact mit Amazon VPC hinzu	Es wurden Informationen zum Erstellen von Amazon S3 S3-Gateway-Endpunkten mit dem EC2 AWS CLI Amazon-Befehl hinzugefügt. Diese Dokumentation enthält auch Informationen zu den spezifischen Berechtigungen, die für die Verwendung in Amazon VPC-Umgebungen CodeArtifact erforderlich sind. Siehe Erstellen Sie den Amazon S3-Gateway-Endpunkt.	12. August 2020
Veröffentlichen von Maven-Artefakten mit Curl und Veröffentlichen von Maven-Artefakten von Drittanbietern	Anleitung für und hinzugefügt. Publizieren mit curl Veröffentlichen Sie Artefakte von Drittanbietern	10. August 2020
Version für allgemeine Verfügbarkeit (GA)	Erste Version des CodeArtifact Benutzerhandbuchs.	10. Juni 2020

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.