CodeArtifact Guida per l'utente

CodeArtifact

CodeArtifact: CodeArtifact Guida per l'utente

Table of Contents

Che cos'è AWS CodeArtifact? 1
Come CodeArtifact funziona? 1
Concetti 2
Asset 2
Domain 2
Repository 3
Pacchetto 3
Gruppo di pacchetti 3
Namespace del pacchetto 4
Versione del pacchetto 4
Revisione della versione del pacchetto 4
Repository upstream 4
Come posso iniziare CodeArtifact? 5
Configurazione 6
Registrati per AWS 6
Installa o aggiorna e quindi configura il AWS CLI 7
Fornisci un utente IAM 8
Installa il tuo gestore di pacchetti o lo strumento di compilazione 10
Fasi successive 10
Nozioni di base 11
Prerequisiti 11
Nozioni di base utilizzando la console 12
Iniziare a usare il AWS CLI 14
Lavorare con i repository 21
Creazione di un repository 21
Crea un repository (console) 22
Crea un repository ()AWS CLI 23
Crea un repository con un repository upstream 24
Connessione a un repository 25
Usa un client di gestione dei pacchetti 25
Eliminare un repository 26
Eliminare un repository (console) 26
Elimina un repository ()AWS CLI 26
Proteggi i repository dall'eliminazione 27

Elenca gli archivi ... 29
Elenca i repository in un account AWS ... 29
Elenca i repository nel dominio ... 30
Visualizza o modifica la configurazione di un repository ... 32
Visualizza o modifica la configurazione di un repository (console) ... 32
Visualizza o modifica la configurazione di un repository ()AWS CLI ... 33
Policy del repository ... 35
Crea una politica delle risorse per concedere l'accesso in lettura ... 36
Imposta una politica ... 38
Leggi una politica ... 39
Eliminazione di una policy ... 40
Concedi l'accesso in lettura ai principali ... 40
Concedi l'accesso in scrittura ai pacchetti ... 41
Concedi l'accesso in scrittura a un repository ... 42
Interazione tra le politiche del repository e del dominio ... 43
Aggiungi un tag a un repository ... 44
Archivi di tag (CLI) ... 44
Archivi di tag (console) ... 48
Lavorare con i repository upstream ... 52
Qual è la differenza tra gli archivi upstream e le connessioni esterne? ... 52
Aggiungere o rimuovere i repository upstream ... 53
Aggiungi o rimuovi repository upstream (console) ... 53
Aggiungi o rimuovi i repository upstream ()AWS CLI ... 54
Connect un CodeArtifact repository a un repository pubblico ... 57
Connect a un repository esterno (console) ... 57
Connect a un repository esterno (CLI) ... 58
Archivi di connessioni esterne supportati ... 60
Rimuovere una connessione esterna (CLI) ... 61
Richiesta di una versione del pacchetto con repository upstream ... 61
Package retention dai repository upstream ... 62
Recupera i pacchetti tramite una relazione a monte ... 62
Conservazione dei pacchetti in repository intermedi ... 64
Richiesta di pacchetti da connessioni esterne ... 65
Recupera i pacchetti da una connessione esterna ... 66
Latenza della connessione esterna ... 67
CodeArtifact comportamento quando un repository esterno non è disponibile ... 68

Disponibilità di nuove versioni del pacchetto ... 69
Importazione di versioni di pacchetti con più di una risorsa ... 69
Ordine di priorità del repository upstream ... 70
Semplice esempio di ordine di priorità ... 71
Esempio di ordine di priorità complesso ... 71
Comportamento delle API con i repository upstream ... 73
Utilizzo dei pacchetti ... 75
Panoramica dei pacchetti ... 75
Formati di pacchetto supportati ... 76
Pubblicazione di pacchetti ... 76
Stato della versione del pacchetto ... 79
Nome del pacchetto, versione del pacchetto e normalizzazione del nome dell'asset ... 80
Elenca i nomi dei pacchetti ... 81
Elenca i nomi dei pacchetti npm ... 82
Elenca i nomi dei pacchetti Maven ... 84
Elenca i nomi dei pacchetti Python ... 85
Filtra per prefisso del nome del pacchetto ... 85
Combinazioni di opzioni di ricerca supportate ... 86
Formatta l'output ... 86
Impostazioni predefinite e altre opzioni ... 87
Elenca le versioni dei pacchetti ... 87
Elenca le versioni del pacchetto npm ... 89
Elenca le versioni del pacchetto Maven ... 90
Ordina le versioni ... 90
Versione di visualizzazione predefinita ... 91
Formatta l'output ... 91
Elenca le risorse della versione del pacchetto ... 92
Elenca gli asset di un pacchetto npm ... 93
Elenca le risorse di un pacchetto Maven ... 93
Scarica gli asset della versione del pacchetto ... 94
Copia i pacchetti tra i repository ... 95
Autorizzazioni IAM richieste per copiare i pacchetti ... 95
Copia le versioni dei pacchetti ... 97
Copia un pacchetto dai repository upstream ... 97
Copia un pacchetto npm con ambito ... 98
Copia le versioni del pacchetto Maven ... 98

Versioni che non esistono nel repository di origine 99
Versioni già esistenti nel repository di destinazione 99
Specificare una revisione della versione del pacchetto 101
Copia i pacchetti npm 102
Eliminare un pacchetto o una versione del pacchetto 102
Eliminazione di un pacchetto (AWS CLI) 103
Eliminazione di un pacchetto (console) 104
Eliminazione di una versione del pacchetto ()AWS CLI 104
Aggiunta di una versione del pacchetto (console) 105
Eliminazione di un pacchetto npm o di una versione del pacchetto 105
Eliminazione di un pacchetto Maven o di una versione del pacchetto 106
Procedure consigliate per l'eliminazione di pacchetti o versioni di pacchetti 106
Visualizza e aggiorna i dettagli e le dipendenze della versione del pacchetto 107
Visualizza i dettagli della versione del pacchetto 107
Visualizza i dettagli della versione del pacchetto npm 108
Visualizza i dettagli della versione del pacchetto Maven 109
Visualizza le dipendenze delle versioni del pacchetto 110
Visualizza il file readme della versione del pacchetto 111
Aggiorna lo stato della versione del pacchetto 112
Aggiornamento dello stato della versione del pacchetto 112
Autorizzazioni IAM richieste per aggiornare lo stato della versione di un pacchetto 114
Aggiornamento dello stato di un pacchetto npm con ambito 114
Aggiornamento dello stato di un pacchetto Maven 115
Specificare una revisione della versione del pacchetto 115
Utilizzo del parametro di stato previsto 116
Errori con le singole versioni del pacchetto 117
Eliminazione delle versioni dei pacchetti 118
Modifica dei controlli di origine dei pacchetti 120
Scenari comuni di controllo dell'accesso ai pacchetti 120
Impostazioni di controllo dell'origine del pacchetto 122
Impostazioni predefinite per il controllo dell'origine dei pacchetti 123
In che modo i controlli di origine dei pacchetti interagiscono con i controlli di origine dei gruppi di pacchetti 124
Modifica dei controlli di origine dei pacchetti 125
Repository editoriali e upstream 126
Lavorare con i gruppi di pacchetti 128

Crea un gruppo di pacchetti 129
Creare un gruppo di pacchetti (console) 129
Crea un gruppo di pacchetti (AWS CLI) 130
Visualizza o modifica un gruppo di pacchetti 131
Visualizza o modifica un gruppo di pacchetti (console) 131
Visualizza o modifica un gruppo di pacchetti ()AWS CLI 132
Eliminare un gruppo di pacchetti 133
Eliminare un gruppo di pacchetti (console) 134
Eliminare un gruppo di pacchetti (AWS CLI) 134
Controlli dell'origine dei gruppi di pacchetti 134
Impostazioni di restrizione 135
Elenchi di repository consentiti 136
Modifica delle impostazioni di controllo dell'origine dei gruppi di pacchetti 137
Esempi di configurazione del controllo dell'origine dei gruppi di pacchetti 138
In che modo le impostazioni di controllo dell'origine del gruppo di pacchetti interagiscono con le impostazioni di controllo dell'origine dei pacchetti 140
Sintassi della definizione del gruppo di pacchetti e comportamento di abbinamento 141
Sintassi ed esempi per la definizione dei gruppi di pacchetti 141
Gerarchia dei gruppi di pacchetti e specificità del modello 143
Parole, confini delle parole e corrispondenza dei prefissi 143
Distinzione tra lettere maiuscole e minuscole 144
Partita forte e debole 145
Varianti aggiuntive 145
Assegna un tag a un gruppo di pacchetti 146
Gruppi di pacchetti di tag (CLI) 146
Utilizzo dei domini 150
Panoramica del dominio 150
Domini con più account 151
Tipi di AWS KMS chiavi supportati in CodeArtifact 152
Creare un dominio 152
Creare un dominio (console) 153
Crea un dominio (AWS CLI) 154
Esempio di politica AWS KMS chiave 155
Eliminazione di un dominio 156
Restrizioni all'eliminazione del dominio 157
Eliminare un dominio (console) 157

Elimina un dominio ()AWS CLI 157
Politiche di dominio 158
Abilita l'accesso tra più account a un dominio 158
Esempio di policy di dominio 161
Esempio di politica di dominio con AWS Organizations 162
Imposta una politica di dominio 163
Leggi una politica di dominio 164
Eliminare una politica di dominio 164
Aggiungi un tag a un dominio 165
Domini di tag (CLI) 165
Domini di tag (console) 168
Usare Cargo 172
Configura e usa Cargo 172
Configura Cargo con CodeArtifact 172
Installazione delle casse Cargo 177
Pubblicazione di casse Cargo 178
Supporto al comando Cargo 178
Comandi supportati che richiedono l'accesso al registro 178
Comandi non supportati 179
Usare Maven 180
Uso CodeArtifact con Gradle 180
Recupera le dipendenze 181
Recupera i plugin 182
Pubblica artefatti 183
Esegui una build Gradle in IntelliJ IDEA 185
Usare CodeArtifact con mvn 189
Recupera le dipendenze 181
Pubblica artefatti 183
Pubblica artefatti di terze parti 194
Limita i download delle dipendenze di Maven a un repository CodeArtifact 195
Informazioni sul progetto Apache Maven 197
Utilizzare CodeArtifact con deps.edn 197
Recupera le dipendenze 197
Pubblica artefatti 199
Pubblicazione con curl 199
Usa i checksum Maven 201

Archiviazione con checksum 202
I checksum non corrispondono durante la pubblicazione 203
Recupero in seguito a mancate corrispondenze nei checksum 204
Usa le istantanee di Maven 205
Pubblicazione di istantanee in CodeArtifact 205
Consumo di versioni istantanee 208
Eliminazione delle versioni istantanee 208
Pubblicazione di istantanee con curl 208
Istantanee e connessioni esterne 211
Istantanee e repository upstream 211
Richiesta di pacchetti Maven da upstream e connessioni esterne 212
Importazione di nomi di risorse standard 212
Importazione di nomi di asset non standard 212
Verifica delle origini delle risorse 213
Importazione di nuove risorse e dello stato delle versioni dei pacchetti nei repository upstream 214
Risoluzione dei problemi con Maven 214
Disabilita i put paralleli per correggere l'errore 429: Troppe richieste 214
Utilizzo di npm 216
Configura e usa npm 216
Configurazione di npm con il comando login 216
Configurazione di npm senza utilizzare il comando login 217
Esecuzione dei comandi npm 220
Verifica dell'autenticazione e dell'autorizzazione di npm 220
Tornare al registro npm predefinito 221
Risoluzione dei problemi di installazioni lente con npm 8.x o versioni successive 221
Configura e usa Yarn 221
Configura Yarn 1.X con il comando `aws codeartifact login` 222
Configura Yarn 2.X con il comando `yarn config set` 223
supporto per i comandi npm 226
Comandi supportati che interagiscono con un repository 226
Comandi lato client supportati 227
Comandi non supportati 179
gestione dei tag npm 232
Modifica i tag con il client npm 232
tag npm e API CopyPackageVersions 232

tag npm e repository upstream 233
Support per gestori di pacchetti compatibili con npm 235
Usando NuGet 236
Utilizzare CodeArtifact con Visual Studio 236
Configurare Visual Studio con il CodeArtifact Credential Provider 237
Utilizzare la console Visual Studio Package Manager 238
Utilizzare con nuget o dotnet CodeArtifact 238
Configurare la CLI nuget o dotnet 239
Consumare NuGet pacchetti 244
Pubblica NuGet pacchetti 245
CodeArtifact NuGet Riferimento al provider di credenziali 246
CodeArtifact NuGet Versioni di Credential Provider 247
NuGet normalizzazione del nome del pacchetto, della versione e del nome dell'asset 248
NuGet compatibilità 249
NuGet Compatibilità generale 249
NuGet supporto da riga di comando 249
Uso di Python 250
Configura e usa pip con CodeArtifact 250
Configura pip con il comando `login` 250
Configura pip senza il comando login 251
Esegui pip 252
Configura e usa twine con CodeArtifact 253
Configura twine con il comando `login` 253
Configura twine senza il comando `login` 253
Esegui twine 254
Normalizzazione dei nomi dei pacchetti in Python 255
Compatibilità con Python 255
supporto per i comandi pip 256
Richiesta di pacchetti Python da upstream e connessioni esterne 257
Versioni del pacchetto modificate 258
Perché CodeArtifact non sta recuperando gli ultimi metadati o risorse eliminati per una versione del pacchetto? 258
Usare Ruby 260
Configura e usa RubyGems e Bundler 260
Configura RubyGems (`gem`) e Bundler () con `bundle` CodeArtifact 260
Installazione di Ruby gems 266

Pubblicazione di gemme di Ruby 267
RubyGems supporto ai comandi 268
Compatibilità con i bundler 268
Compatibilità con Bundler 269
Usare Swift 270
Configura Swift con CodeArtifact 270
Configura Swift con il comando login 270
Configura Swift senza il comando di login 272
Consumo e pubblicazione di pacchetti Swift 276
Consumo di pacchetti Swift 276
Consumo di pacchetti Swift in Xcode 277
Pubblicazione di pacchetti Swift 278
Recupero di pacchetti Swift GitHub e ripubblicazione su CodeArtifact 281
Normalizzazione del nome e dello spazio dei nomi del pacchetto Swift 283
Risoluzione dei problemi Swift 283
Ricevo un errore 401 in Xcode anche dopo aver configurato Swift Package Manager 283
Xcode si blocca sulla macchina CI a causa della richiesta di password del portachiavi 284
Utilizzo di pacchetti generici 287
Panoramica dei pacchetti generici 287
Vincoli relativi ai pacchetti generici 287
Comandi supportati 288
Pubblicazione e utilizzo di pacchetti generici 289
Pubblicazione di un pacchetto generico 289
Elenco delle risorse del pacchetto generico 291
Scaricamento delle risorse del pacchetto generico 293
Utilizzo CodeArtifact con CodeBuild 294
Utilizzo dei pacchetti npm in CodeBuild 294
Configura le autorizzazioni con i ruoli IAM 294
Accedi e usa npm 295
Usare i pacchetti Python in CodeBuild 296
Configura le autorizzazioni con i ruoli IAM 297
Accedi e usa pip o twine 298
Usare i pacchetti Maven in CodeBuild 300
Configura le autorizzazioni con i ruoli IAM 300
Usa gradle o mvn 301
Utilizzo di pacchetti in NuGet CodeBuild 302

Configura le autorizzazioni con i ruoli IAM 302
Consumare pacchetti NuGet 303
Compila con NuGet pacchetti 305
Pubblica NuGet pacchetti 307
Memorizzazione nella cache delle dipendenze 308
Monitoraggio CodeArtifact 310
Monitoraggio degli eventi CodeArtifact 310
CodeArtifact formato ed esempio dell'evento 312
Usa un evento per avviare un'esecuzione CodePipeline 316
Configura le autorizzazioni EventBridge 317
Crea la regola EventBridge 317
Crea l'obiettivo della EventBridge regola 317
Utilizzare un evento per eseguire una funzione Lambda 317
Crea la EventBridge regola 318
Crea l'obiettivo della regola EventBridge 318
Configura le autorizzazioni EventBridge 318
Sicurezza 319
Protezione dei dati 320
Crittografia dei dati 321
Privacy del traffico 321
Monitoraggio 321
Registrazione delle chiamate API con CodeArtifact AWS CloudTrail 322
Convalida della conformità 326
Autenticazione e token 326
Token creati con il comando `login` 328
Autorizzazioni necessarie per chiamare l'API `GetAuthorizationToken` 329
Token creati con l'API `GetAuthorizationToken` 329
Passa un token di autenticazione utilizzando una variabile di ambiente 331
Revoca CodeArtifact dei token di autorizzazione 332
Resilienza 332
Sicurezza dell’infrastruttura 332
Attacchi di sostituzione delle dipendenze 333
Identity and Access Management 334
Destinatari 334
Autenticazione con identità 335
Gestione dell’accesso tramite policy 336

Come AWS CodeArtifact funziona con IAM 338
Esempi di policy basate su identità 344
Utilizzo dei tag per controllare l'accesso alle risorse CodeArtifact 354
AWS CodeArtifact riferimento alle autorizzazioni 358
risoluzione dei problemi 362
Uso di endpoint VPC 364
Creazione di endpoint VPC 364
Creare l'endpoint gateway Amazon S3 366
Autorizzazioni minime per i bucket Amazon S3 per AWS CodeArtifact 366
Utilizzo CodeArtifact da un VPC 368
Usa l'`codeartifact.repositories`endpoint senza DNS privato 369
Creazione di una policy di endpoint VPC 370
AWS CloudFormation risorse 372
CodeArtifact e CloudFormation modelli 372
Prevenzione dell'eliminazione delle risorse CodeArtifact 372
Scopri di più su CloudFormation 373
Risoluzione dei problemi 374
Non riesco a visualizzare le notifiche 374
Applicazione di tag alle risorse 375
CodeArtifact allocazione dei costi con tag 376
Allocazione dei costi di archiviazione dei dati in CodeArtifact 376
Allocazione dei costi delle richieste in CodeArtifact 376
Quote in AWS CodeArtifact 377
Cronologia dei documenti 380
..... cccxciii

Che cos'è AWS CodeArtifact?

AWS CodeArtifact è un servizio di repository di artefatti gestito sicuro, altamente scalabile e che aiuta le organizzazioni a archiviare e condividere pacchetti software per lo sviluppo di applicazioni. Puoi utilizzarlo CodeArtifact con i più diffusi strumenti di compilazione e gestori di pacchetti come NuGet CLI, Maven, Gradle, npm, yarn, pip e twine. CodeArtifact aiuta a ridurre la necessità di gestire il proprio sistema di storage degli artefatti o di preoccuparsi della scalabilità della sua infrastruttura. Non ci sono limiti al numero o alla dimensione totale dei pacchetti che è possibile archiviare in un repository. CodeArtifact

Puoi creare una connessione tra il tuo CodeArtifact repository privato e un archivio pubblico esterno, come npmjs.com o Maven Central. CodeArtifact recupererà e archivierà quindi i pacchetti su richiesta dall'archivio pubblico quando vengono richiesti da un gestore di pacchetti. Ciò semplifica l'utilizzo delle dipendenze open source utilizzate dall'applicazione e contribuisce a garantire che siano sempre disponibili per le build e lo sviluppo. Puoi anche pubblicare pacchetti privati in un repository. CodeArtifact Ciò consente di condividere componenti software proprietari tra più applicazioni e team di sviluppo dell'organizzazione.

Per ulteriori informazioni, consulta AWS CodeArtifact.

Come CodeArtifact funziona?

CodeArtifact archivia i pacchetti software nei repository. I repository sono poliglotti: un singolo repository può contenere pacchetti di qualsiasi tipo supportato. Ogni CodeArtifact repository è membro di un singolo dominio. CodeArtifact Ti consigliamo di utilizzare un dominio di produzione per la tua organizzazione con uno o più repository. Ad esempio, potresti utilizzare ogni repository per un team di sviluppo diverso. I pacchetti presenti nei repository possono quindi essere scoperti e condivisi tra i team di sviluppo.

Per aggiungere pacchetti a un repository, configura un gestore di pacchetti come npm o Maven per utilizzare l'endpoint (URL) del repository. È quindi possibile utilizzare il gestore di pacchetti per pubblicare i pacchetti nel repository. Puoi anche importare pacchetti open source in un repository configurandolo con una connessione esterna a un archivio pubblico come npmjs, Gallery NuGet , Maven Central o PyPI. Per ulteriori informazioni, consulta Connect un CodeArtifact repository a un repository pubblico.

È possibile rendere i pacchetti in un repository disponibili in un altro repository nello stesso dominio. A tale scopo, configura un repository come upstream dell'altro. Tutte le versioni del pacchetto disponibili

per il repository upstream sono disponibili anche per il repository downstream. Inoltre, tutti i pacchetti disponibili nell'archivio upstream tramite una connessione esterna a un archivio pubblico sono disponibili nel repository downstream. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.

CodeArtifact richiede agli utenti di autenticarsi con il servizio per pubblicare o utilizzare le versioni dei pacchetti. È necessario autenticarsi al CodeArtifact servizio creando un token di autorizzazione utilizzando le proprie AWS credenziali. I pacchetti nei CodeArtifact repository non possono essere resi disponibili pubblicamente. Per ulteriori informazioni sull'autenticazione e l'accesso in CodeArtifact, vedereAWS CodeArtifact autenticazione e token.

Concetti AWS CodeArtifact

Di seguito sono riportati alcuni concetti e termini da conoscere quando si utilizza. CodeArtifact

Argomenti

- Asset
- Domain
- Repository
- Pacchetto
- Gruppo di pacchetti
- Namespace del pacchetto
- Versione del pacchetto
- Revisione della versione del pacchetto
- Repository upstream

Asset

Una risorsa è un singolo file archiviato in CodeArtifact associato a una versione del pacchetto, ad esempio un file npm o `.tgz` file Maven POM e JAR.

Domain

I repository vengono aggregati in un'entità di livello superiore nota come dominio. Tutte le risorse e i metadati del pacchetto vengono archiviati nel dominio, ma vengono utilizzati tramite i repository. Una determinata risorsa del pacchetto, ad esempio un file JAR Maven, viene archiviata una volta per

dominio, indipendentemente dal numero di repository in cui è presente. Tutte le risorse e i metadati di un dominio sono crittografati con la stessa AWS KMS key (chiave KMS) archiviata in (). AWS Key Management Service AWS KMS

Ogni repository è membro di un singolo dominio e non può essere spostato in un dominio diverso.

Utilizzando un dominio, puoi applicare una politica organizzativa su più repository. Con questo approccio, è possibile determinare quali account possono accedere agli archivi del dominio e quali archivi pubblici possono essere utilizzati come fonti dei pacchetti.

Sebbene un'organizzazione possa avere più domini, consigliamo un unico dominio di produzione che contenga tutti gli elementi pubblicati. In questo modo, i team possono trovare e condividere pacchetti all'interno dell'organizzazione.

Repository

Un CodeArtifact repository contiene una serie di versioni del pacchetto, ognuna delle quali è associata a un insieme di risorse. I repository sono poliglotti: un singolo repository può contenere pacchetti di qualsiasi tipo supportato. Ogni repository espone gli endpoint per il recupero e la pubblicazione di pacchetti utilizzando strumenti come nuget CLI, npm CLI, Maven CLI (mvn) e pip. Puoi creare fino a 1.000 repository per dominio.

Pacchetto

Un pacchetto è un insieme di software e metadati necessari per risolvere le dipendenze e installare il software. In CodeArtifact, un pacchetto è costituito da un nome di pacchetto, uno spazio dei nomi opzionale come `@types` in`@types/node`, un set di versioni del pacchetto e metadati a livello di pacchetto come i tag npm.

AWS CodeArtifact supporta i formati di pacchetti Cargo, generic, Maven, npm, NuGetPyPI, Ruby, Swift.

Gruppo di pacchetti

I gruppi di pacchetti possono essere utilizzati per applicare la configurazione a più pacchetti che corrispondono a uno schema definito utilizzando il formato del pacchetto, lo spazio dei nomi del pacchetto e il nome del pacchetto. È possibile utilizzare i gruppi di pacchetti per configurare più comodamente i controlli di origine dei pacchetti per più pacchetti. I controlli di origine dei pacchetti vengono utilizzati per bloccare o consentire l'inserimento o la pubblicazione di nuove versioni

dei pacchetti, proteggendo gli utenti da azioni dannose note come attacchi di sostituzione delle dipendenze.

Namespace del pacchetto

Alcuni formati di pacchetti supportano nomi di pacchetto gerarchici per organizzare i pacchetti in gruppi logici e aiutare a evitare collisioni di nomi. Ad esempio, npm supporta gli ambiti. Per ulteriori informazioni, consulta la documentazione di npm scopes. Il pacchetto npm `@types/node` ha un ambito `@types` e un nome di. `node` Ci sono molti altri nomi di pacchetti nell'`@types`ambito. In CodeArtifact, l'ambito («tipi») viene definito spazio dei nomi del pacchetto e il nome («nodo») viene indicato come nome del pacchetto. Per i pacchetti Maven, lo spazio dei nomi del pacchetto corrisponde a Maven GroupID. Il pacchetto Maven `org.apache.logging.log4j:log4j` ha un groupID (spazio dei nomi del pacchetto) e un artifactID (nome del pacchetto). `org.apache.logging.log4j` `log4j` Per i pacchetti generici, è richiesto uno spazio dei nomi. Alcuni formati di pacchetti come PyPI non supportano nomi gerarchici con un concetto simile a npm scope o Maven GroupID. Senza un modo per raggruppare i nomi dei pacchetti, può essere più difficile evitare le collisioni di nomi.

Versione del pacchetto

Una versione del pacchetto identifica la versione specifica di un pacchetto, ad esempio. `@types/node` `12.6.9` Il formato e la semantica del numero di versione variano a seconda dei diversi formati di pacchetto. Ad esempio, le versioni del pacchetto npm devono essere conformi alla specifica Semantic Versioning. In CodeArtifact, una versione del pacchetto è composta dall'identificatore di versione, dai metadati a livello di versione del pacchetto e da un set di risorse.

Revisione della versione del pacchetto

Una revisione della versione del pacchetto è una stringa che identifica un insieme specifico di risorse e metadati per una versione del pacchetto. Ogni volta che una versione del pacchetto viene aggiornata, viene creata una nuova revisione della versione del pacchetto. Ad esempio, potresti pubblicare un archivio di distribuzione dei sorgenti (sdist) per una versione del pacchetto Python e successivamente aggiungere una ruota Python che contiene codice compilato nella stessa versione. Quando pubblicate la ruota, viene creata una nuova revisione della versione del pacchetto.

Repository upstream

Un repository è a monte di un altro quando è possibile accedere alle versioni dei pacchetti in esso contenute dall'endpoint del repository downstream. Questo approccio unisce efficacemente i

contenuti dei due repository dal punto di vista di un client. Utilizzando CodeArtifact, è possibile creare una relazione a monte tra due repository.

Come posso iniziare CodeArtifact?

È consigliabile completare la procedura seguente:

1. Scopri di più CodeArtifact leggendoConcetti AWS CodeArtifact.
2. Configura il tuo Account AWS AWS CLI, il e un utente IAM seguendo la procedura riportata di seguitoConfigurazione con AWS CodeArtifact.
3. CodeArtifact Usalo seguendo le istruzioni riportate inGuida introduttiva con CodeArtifact.

Configurazione con AWS CodeArtifact

Se ti sei già registrato ad Amazon Web Services (AWS), puoi iniziare a utilizzare AWS CodeArtifact immediatamente. Puoi aprire la CodeArtifact console, scegliere Crea un dominio e un repository e seguire i passaggi della procedura guidata di avvio per creare il tuo primo dominio e repository.

Se non ti sei AWS ancora registrato o hai bisogno di assistenza per creare il tuo primo dominio e il tuo primo repository, completa le seguenti attività per iniziare a utilizzare: CodeArtifact

Argomenti

- Registrati per AWS
- Installa o aggiorna e quindi configura il AWS CLI
- Fornisci un utente IAM
- Installa il tuo gestore di pacchetti o lo strumento di compilazione

Registrati per AWS

Quando ti iscrivi ad Amazon Web Services (AWS), ti vengono addebitati solo i servizi e le risorse che utilizzi, tra cui AWS CodeArtifact.

Se ne hai già uno Account AWS, passa all'attività successiva,Installa o aggiorna e quindi configura il AWS CLI. Se non ne hai uno Account AWS, usa la procedura seguente per crearne uno.

Per creare un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire le attività che richiedono l'accesso di un utente root.

Installa o aggiorna e quindi configura il AWS CLI

Per richiamare CodeArtifact i comandi da AWS Command Line Interface (AWS CLI) su una macchina di sviluppo locale, è necessario installare AWS CLI.

Se è installata una versione precedente di quella AWS CLI installata, è necessario aggiornarla in modo che i CodeArtifact comandi siano disponibili. CodeArtifact i comandi sono disponibili nelle seguenti AWS CLI versioni:

1. AWS CLI 1: 1.18.77 e versioni successive
2. AWS CLI 2:2.0.21 e versioni successive

Per verificare la versione, usa il comando. `aws --version`

Per installare e configurare AWS CLI

1. Installare o aggiornare il file seguendo AWS CLI le istruzioni riportate in Installazione di AWS Command Line Interface.
2. Configurare AWS CLI, con il comando configure, come segue.

   ```
   aws configure
   ```

   Quando richiesto, specifica la chiave di AWS accesso e la chiave di accesso AWS segreta dell'utente IAM con CodeArtifact cui utilizzerai. Quando viene richiesto il Regione AWS nome predefinito, specifica la regione in cui creerai la pipeline, ad esempio. `us-east-2` Quando viene richiesto il formato di output predefinito, specificare `json`.

   > **Important**
   >
   > Quando configurate il AWS CLI, vi viene richiesto di specificare un. Regione AWS Scegli una delle regioni supportate elencate in Regione ed endpoint in. Riferimenti generali di AWS

   Per ulteriori informazioni, consulta Configurazione AWS Command Line Interface e gestione delle chiavi di accesso per gli utenti IAM.
3. Per verificare l'installazione o l'aggiornamento, chiama il seguente comando da. AWS CLI

```
aws codeartifact help
```

In caso di successo, questo comando visualizza un elenco di CodeArtifact comandi disponibili.

Successivamente, puoi creare un utente IAM e concedere a quell'utente l'accesso a CodeArtifact. Per ulteriori informazioni, consulta Fornisci un utente IAM.

Fornisci un utente IAM

Segui queste istruzioni per preparare un utente IAM all'uso CodeArtifact.

Per effettuare il provisioning di un utente IAM

1. Crea un utente IAM o usane uno associato al tuo Account AWS. Per ulteriori informazioni, consulta Creazione di un utente IAM e Panoramica delle politiche AWS IAM nella Guida per l'utente IAM.
2. Concedi all'utente IAM l'accesso a CodeArtifact.

   - Opzione 1: crea una policy IAM personalizzata. Con una policy IAM personalizzata, puoi fornire le autorizzazioni minime richieste e modificare la durata dei token di autenticazione. Per ulteriori informazioni e policy di esempio, consulta Esempi di policy basate sull'identità per AWS CodeArtifact.
   - Opzione 2: utilizza la policy `AWSCodeArtifactAdminAccess` AWS gestita. Il seguente frammento mostra il contenuto di questa politica.

     > **Important**
     >
     > Questa politica garantisce l'accesso a tutti. CodeArtifact APIs È consigliabile utilizzare sempre le autorizzazioni minime necessarie per eseguire l'attività. Per ulteriori informazioni, consultare la sezione best practice IAM nella Guida per l'utente IAM.

     JSON

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

```
      {
         "Action": [
            "codeartifact:*"
         ],
         "Effect": "Allow",
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": "sts:GetServiceBearerToken",
         "Resource": "*",
            "Condition": {
               "StringEquals": {
                  "sts:AWSServiceName": "codeartifact.amazonaws.com"
               }
            }
      }
   ]
}
```

> **Note**
>
> L'`sts:GetServiceBearerToken`autorizzazione deve essere aggiunta alla policy relativa agli utenti o ai ruoli IAM. Sebbene possa essere aggiunta a una politica delle risorse del CodeArtifact dominio o del repository, l'autorizzazione non avrà alcun effetto sulle politiche delle risorse.

L'`sts:GetServiceBearerToken`autorizzazione è necessaria per chiamare l' CodeArtifact`GetAuthorizationToken`API. Questa API restituisce un token che deve essere utilizzato quando si utilizza un gestore di pacchetti come `npm` o `pip` con CodeArtifact. Per utilizzare un gestore di pacchetti con un CodeArtifact repository, l'utente o il ruolo IAM deve consentire, `sts:GetServiceBearerToken` come mostrato nell'esempio di policy precedente.

Se non hai installato il gestore di pacchetti o lo strumento di compilazione che intendi utilizzare CodeArtifact, consulta. Installa il tuo gestore di pacchetti o lo strumento di compilazione

Installa il tuo gestore di pacchetti o lo strumento di compilazione

Per pubblicare o utilizzare pacchetti da CodeArtifact, è necessario utilizzare un gestore di pacchetti. Esistono diversi gestori di pacchetti per ogni tipo di pacchetto. L'elenco seguente contiene alcuni gestori di pacchetti con cui è possibile utilizzare CodeArtifact. Se non l'hai già fatto, installa i gestori di pacchetti per il tipo di pacchetto che desideri utilizzare.

- Per npm, usa npm CLI o pnpm.
- Per Maven, usa Apache Maven () o Gradle. `mvn`
- Per Python, usa pip per installare i pacchetti e twine per pubblicare i pacchetti.
- Per NuGet, usa il Toolkit for Visual Studio in Visual Studio o nuget o dotnet. CLIs
- Per i pacchetti generici, usa AWS CLIo SDK per pubblicare e scaricare il contenuto del pacchetto.

Fasi successive

I passaggi successivi dipenderanno dal tipo o dai tipi di pacchetto con CodeArtifact cui stai utilizzando e dallo stato delle tue CodeArtifact risorse.

Se stai iniziando CodeArtifact per la prima volta per te stesso, il tuo team o la tua organizzazione, consulta la seguente documentazione per informazioni generali su come iniziare e aiutarti a creare le risorse di cui avrai bisogno.

- Nozioni di base utilizzando la console
- Iniziare a usare il AWS CLI

Se le tue risorse sono già state create e sei pronto a configurare il tuo gestore di pacchetti per inviare pacchetti o installarli da un CodeArtifact repository, consulta la documentazione corrispondente al tipo di pacchetto o al gestore di pacchetti in uso.

- Usare CodeArtifact con npm
- Usare CodeArtifact con Python
- Utilizzo CodeArtifact con Maven
- Utilizzo CodeArtifact con NuGet
- Utilizzo CodeArtifact con pacchetti generici

Guida introduttiva con CodeArtifact

In questo tutorial introduttivo, creerai quanto segue: CodeArtifact

- Un dominio chiamato`my-domain`.
- Un repository chiamato `my-repo` così è contenuto in`my-domain`.
- Un repository chiamato `npm-store` così è contenuto in. `my-domain` `npm-store`Ha una connessione esterna al repository pubblico di npm. Questa connessione viene utilizzata per inserire un pacchetto npm nel repository. `my-repo`

Prima di iniziare questo tutorial, ti consigliamo di esaminarlo. CodeArtifact Concetti AWS CodeArtifact

> **Note**
>
> Questo tutorial richiede la creazione di risorse che potrebbero comportare addebiti sull'account AWS . Per ulteriori informazioni, consulta Prezzi di CodeArtifact .

Argomenti

- Prerequisiti
- Nozioni di base utilizzando la console
- Iniziare a usare il AWS CLI

Prerequisiti

Puoi completare questo tutorial usando il Console di gestione AWS o il AWS Command Line Interface (AWS CLI). Per seguire il tutorial, devi prima completare i seguenti prerequisiti:

- Completa le fasi descritte in Configurazione con AWS CodeArtifact.
- Installa la CLI di npm. Per ulteriori informazioni, vedere Download e installazione di Node.js e npm nella documentazione di npm.

Nozioni di base utilizzando la console

Esegui i passaggi seguenti per iniziare a CodeArtifact utilizzare. Console di gestione AWS Questa guida utilizza il gestore di `npm` pacchetti, se utilizzi un gestore di pacchetti diverso, dovrai modificare alcuni dei seguenti passaggi.

1. Accedi Console di gestione AWS e apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/start. Per ulteriori informazioni, consulta Configurazione con AWS CodeArtifact.
2. Scegli Create repository (Crea repository).
3. Nel nome del repository, inserisci. **`my-repo`**
4. (Facoltativo) In Descrizione del repository, inserite una descrizione facoltativa per il repository.
5. Nei repository upstream pubblici, seleziona npm-store per creare un repository connesso a npmjs che sia a monte del tuo repository. `my-repo`

   CodeArtifact `npm-store`assegna il nome a questo repository per te. Tutti i pacchetti disponibili nel repository upstream `npm-store` sono disponibili anche nel repository downstream,. `my-repo`
6. Scegli Next (Successivo).
7. In Account AWS, scegli Questo account AWS.
8. In Nome di dominio, inserisci**`my-domain`**.
9. Espandere Additional configuration (Configurazione aggiuntiva).
10. È necessario utilizzare una AWS KMS key (chiave KMS) per crittografare tutte le risorse del dominio. Puoi utilizzare una Chiave gestita da AWS o una chiave KMS che gestisci:

    - Scegli la chiave gestita AWS se desideri utilizzare la chiave predefinita Chiave gestita da AWS.
    - Scegli la chiave gestita dal cliente se desideri utilizzare una chiave KMS da te gestita. Per utilizzare una chiave KMS che gestisci, in ARN della chiave gestita dal cliente, cerca e scegli la chiave KMS.

    Per ulteriori informazioni, consulta la sezione Chiave gestita da AWSCustomer managed key nella Developer Guide.AWS Key Management Service
11. Scegli Next (Successivo).
12. In Rivedi e crea, esamina ciò CodeArtifact che stai creando per te.

- Il Package Flow mostra come `my-domain`my-repo, e `npm-store` sono correlati.
- Passaggio 1: Crea un repository mostra i dettagli su `my-repo` en`npm-store`.
- Passaggio 2: Seleziona il dominio che mostra i dettagli su`my-domain`.

Quando sei pronto, scegli Crea repository.

13. Nella pagina my-repo, scegli Visualizza istruzioni di connessione, quindi scegli npm.
14. Usa AWS CLI per eseguire il `login` comando mostrato in Configura il tuo client npm usando questo comando. AWS CLI CodeArtifact

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

Dovresti ricevere un output di conferma che il tuo accesso è riuscito.

```
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
```

Se ricevi l'erroreCould not connect to the endpoint URL, assicurati che AWS CLI sia configurato e che il nome della regione predefinito sia impostato sulla stessa regione in cui hai creato il repository, vedi Configurazione dell'interfaccia a riga di comando AWS.

Per ulteriori informazioni, consulta Configura e usa npm con CodeArtifact

15. Usa la CLI npm per installare un pacchetto npm. Ad esempio, per installare il popolare pacchetto npm`lodash`, usa il seguente comando.

```
npm install lodash
```

16. Ritorna alla CodeArtifact console. Se il tuo repository my-repo è aperto, aggiorna la pagina. Altrimenti, nel pannello di navigazione, scegli Repositories, quindi scegli my-repo.

In Pacchetti, dovresti vedere la libreria o il pacchetto npm che hai installato. Puoi scegliere il nome del pacchetto per visualizzarne la versione e lo stato. Puoi scegliere la versione più recente per visualizzare i dettagli del pacchetto come dipendenze, risorse e altro.

> **Note**
>
> Potrebbe esserci un ritardo tra il momento in cui installi il pacchetto e il momento in cui viene inserito nel tuo repository.

17. Per evitare ulteriori AWS addebiti, elimina le risorse che hai utilizzato durante questo tutorial:

> **Note**
>
> Non è possibile eliminare un dominio che contiene repository, quindi è necessario eliminare `my-repo` e `npm-store` prima di eliminare`my-domain`.

    a. Dal riquadro di navigazione, scegli Repository.
    b. Scegli npm-store, scegli Elimina, quindi segui i passaggi per eliminare il repository.
    c. Scegli my-repo, scegli Elimina, quindi segui i passaggi per eliminare il repository.
    d. Dal pannello di navigazione, scegli Domini.
    e. Scegli il mio dominio, scegli Elimina, quindi segui i passaggi per eliminare il dominio.

Iniziare a usare il AWS CLI

Esegui i passaggi seguenti per iniziare a CodeArtifact utilizzare AWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta Installa o aggiorna e quindi configura il AWS CLI. Questa guida utilizza il gestore di `npm` pacchetti, se utilizzi un gestore di pacchetti diverso, dovrai modificare alcuni dei seguenti passaggi.

1. Usa il AWS CLI per eseguire il create-domain comando.

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul nuovo dominio.

```
{
    "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
    }
}
```

Se ricevi l'erroreCould not connect to the endpoint URL, assicurati che AWS CLI sia configurato e che il nome della regione predefinito sia impostato sulla stessa regione in cui hai creato il repository, vedi Configurazione dell'interfaccia a riga di comando AWS.

2. Usa il create-repository comando per creare un repository nel tuo dominio.

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul nuovo repository.

```
{
    "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. Usa il create-repository comando per creare un repository upstream per il tuo repository. `my-repo`

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul nuovo repository.

```
{
    "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. Usa il associate-external-connection comando per aggiungere una connessione esterna al repository pubblico npm al tuo repository. `npm-store`

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul repository e sulla sua nuova connessione esterna.

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

Per ulteriori informazioni, consulta Connect un CodeArtifact repository a un repository pubblico.

5. Utilizzate il update-repository comando per associare il `npm-store` repository come repository upstream al repository. `my-repo`

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul repository aggiornato, incluso il nuovo repository upstream.

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

Per ulteriori informazioni, consulta Aggiungi o rimuovi i repository upstream ()AWS CLI.

6. Usa il login comando per configurare il tuo gestore di pacchetti npm con il tuo repository. `my-repo`

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

Dovresti ricevere un output che conferma che il tuo accesso è riuscito.

```
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
```

Per ulteriori informazioni, consulta Configura e usa npm con CodeArtifact.

7. Usa la CLI npm per installare un pacchetto npm. Ad esempio, per installare il popolare pacchetto npm`lodash`, usa il seguente comando.

```
npm install lodash
```

8. Usa il list-packages comando per visualizzare il pacchetto che hai appena installato nel tuo `my-repo` repository.

> **Note**
>
> Potrebbe verificarsi un ritardo tra il completamento del comando di `npm install` installazione e il momento in cui il pacchetto è visibile nel repository. Per i dettagli sulla latenza tipica durante il recupero di pacchetti da archivi pubblici, consulta. Latenza della connessione esterna

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

I dati in formato JSON vengono visualizzati nell'output con il formato e il nome del pacchetto installato.

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

Ora hai tre risorse: CodeArtifact

- Il dominio`my-domain`.
- Il repository `my-repo` contenuto in`my-domain`. Questo repository ha a disposizione un pacchetto npm.

- Il repository in `npm-store` cui è contenuto. `my-domain` Questo repository ha una connessione esterna al repository pubblico npm ed è associato come repository upstream al repository. `my-repo`

9. Per evitare ulteriori AWS addebiti, elimina le risorse che hai utilizzato durante questo tutorial:

> **Note**
>
> Non è possibile eliminare un dominio che contiene repository, quindi è necessario eliminare `my-repo` e `npm-store` prima di eliminare`my-domain`.

a. Usa il delete-repository comando per eliminare il `npm-store` repository.

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sull'archivio eliminato.

```
{
    "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. Usa il delete-repository comando per eliminare il repository. `npm-store`

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sull'archivio eliminato.

```
{
    "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. Usa il delete-domain comando per eliminare il repository. `my-domain`

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul dominio eliminato.

```
{
    "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
    }
}
```

Lavorare con i repository in CodeArtifact

Questi argomenti mostrano come utilizzare la CodeArtifact console e come creare AWS CLI, CodeArtifact APIs elencare, aggiornare ed eliminare i repository.

Argomenti

- Creazione di un repository
- Connessione a un repository
- Eliminare un repository
- Elenca gli archivi
- Visualizza o modifica la configurazione di un repository
- Policy del repository
- Aggiungi un tag a un repository CodeArtifact

Creazione di un repository

Poiché tutti i pacchetti CodeArtifact sono archiviati in repository, per CodeArtifact utilizzarli è necessario crearne uno. È possibile creare un repository utilizzando la CodeArtifact console, il AWS Command Line Interface (AWS CLI) o. CloudFormation Ogni repository è associato all' AWS account che usi al momento della creazione. È possibile disporre di più repository, che vengono creati e raggruppati in domini. Quando si crea un repository, questo non contiene alcun pacchetto. I repository sono poliglotti, il che significa che un singolo repository può contenere pacchetti di qualsiasi tipo supportato.

Per informazioni sui limiti CodeArtifact del servizio, ad esempio il numero massimo di repository consentiti in un singolo dominio, consulta. Quote in AWS CodeArtifact Se raggiungi il numero massimo di repository consentiti, puoi eliminare i repository per fare spazio ad altri.

A un repository possono essere associati uno o più CodeArtifact repository come repository upstream. Ciò consente a un client di gestione dei pacchetti di accedere ai pacchetti contenuti in più di un repository utilizzando un singolo endpoint URL. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.

Per ulteriori informazioni sulla gestione degli CodeArtifact archivi con CloudFormation, vedere. Creare CodeArtifact risorse con AWS CloudFormation

> **Note**
>
> Dopo aver creato un repository, non è possibile modificarne il nome, l' AWS account associato o il dominio.

Argomenti

- Crea un repository (console)
- Crea un repository ()AWS CLI
- Crea un repository con un repository upstream

Crea un repository (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Repository, quindi scegli Crea repository.
3. Per Nome archivio, inserisci un nome per il tuo repository.
4. (Facoltativo) In Descrizione del repository, inserisci una descrizione facoltativa per il tuo repository.
5. (Facoltativo) In Publish upstream repositories, aggiungi repository intermedi che collegano i tuoi repository alle autorità di gestione dei pacchetti come Maven Central o npmjs.com.
6. Scegli Next (Successivo).
7. In Account AWS, scegli Questo account AWS se hai effettuato l'accesso all'account proprietario del dominio. Scegli un altro account AWS se il dominio è di proprietà di un altro account AWS.
8. In Dominio, scegli il dominio in cui verrà creato il repository.

   Se non ci sono domini nell'account, devi crearne uno. Inserisci il nome per il nuovo dominio in Nome dominio.

   Espandere Additional configuration (Configurazione aggiuntiva).

   È necessario utilizzare una AWS KMS key (chiave KMS) per crittografare tutte le risorse del dominio. Puoi utilizzare una Chiave gestita da AWS o una chiave KMS che gestisci:

> **Important**
>
> CodeArtifact supporta solo chiavi KMS simmetriche. Non puoi utilizzare una chiave KMS asimmetrica per crittografare i tuoi domini. CodeArtifact Per informazioni su come determinare se una chiave KMS è simmetrica o asimmetrica, consulta Identificazione di chiavi KMS simmetriche e asimmetriche.

- Scegli la chiave gestita AWS se desideri utilizzare la chiave predefinita Chiave gestita da AWS.
- Scegli la chiave gestita dal cliente se desideri utilizzare una chiave KMS da te gestita. Per utilizzare una chiave KMS che gestisci, in ARN della chiave gestita dal cliente, cerca e scegli la chiave KMS.

Per ulteriori informazioni, consulta la sezione Chiavi gestite da AWSe la chiave gestita dal cliente nella Guida per gli AWS Key Management Service sviluppatori.

9. Scegli Next (Successivo).
10. In Rivedi e crea, esamina ciò CodeArtifact che stai creando per te.
    - Il flusso del pacchetto mostra come sono collegati il dominio e i repository.
    - Passaggio 1: Crea repository mostra i dettagli sul repository e sui repository upstream opzionali che verranno creati.
    - Passaggio 2: Seleziona il dominio e mostra i dettagli relativi a. `my_domain`

    Quando sei pronto, scegli Crea repository.

Crea un repository ()AWS CLI

Usa il `create-repository` comando per creare un repository nel tuo dominio.

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

Output di esempio:

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

Un nuovo repository non contiene pacchetti. Ogni repository è associato all' AWS account con cui sei autenticato al momento della creazione del repository.

Crea un repository con tag

Per creare un repository con tag, aggiungi il `--tags` parametro al tuo `create-domain` comando.

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

Crea un repository con un repository upstream

È possibile specificare uno o più repository upstream quando si crea un repository.

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

Output di esempio:

```
{
    "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**
>
> Per creare un repository con un repository upstream, è necessario disporre dell'autorizzazione per l'`AssociateWithDownstreamRepository`azione sul repository upstream.

Per aggiungere un upstream a un repository dopo che è stato creato, consulta e. Aggiungi o rimuovi repository upstream (console) Aggiungi o rimuovi i repository upstream ()AWS CLI

Connessione a un repository

Dopo aver configurato il profilo e le credenziali per l'autenticazione sul tuo AWS account, decidi in quale repository utilizzare. CodeArtifact Sono disponibili le seguenti opzioni:

- Creare un repository . Per ulteriori informazioni, consulta Creazione di un repository.
- Usa un repository già esistente nel tuo account. Puoi usare il `list-repositories` comando per trovare i repository creati nel tuo AWS account. Per ulteriori informazioni, consulta Elenca gli archivi.
- Usa un repository in un altro AWS account. Per ulteriori informazioni, consulta le politiche del repository.

Usa un client di gestione dei pacchetti

Dopo aver individuato il repository che desideri utilizzare, consulta uno dei seguenti argomenti.

- Utilizzo CodeArtifact con Maven

- Usare con npm CodeArtifact
- Usando con CodeArtifact NuGet
- Usare CodeArtifact con Python

Eliminare un repository

È possibile eliminare un repository utilizzando la CodeArtifact console o il. AWS CLI Dopo che un repository è stato eliminato, non è più possibile inviarvi pacchetti o estrarre pacchetti da esso. Tutti i pacchetti nel repository diventano definitivamente non disponibili e non possono essere ripristinati. È possibile creare un repository con lo stesso nome, ma il suo contenuto sarà vuoto.

> **Important**
>
> L'eliminazione di un repository non può essere annullata. Dopo aver eliminato un repository, non è più possibile recuperarlo e non può essere ripristinato.

Argomenti

- Eliminare un repository (console)
- Elimina un repository ()AWS CLI
- Proteggi i repository dall'eliminazione

Eliminare un repository (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel pannello di navigazione, scegli Repository, quindi scegli il repository che desideri eliminare.
3. Scegli Elimina e segui i passaggi per eliminare il dominio.

Elimina un repository ()AWS CLI

Usa il `delete-repository` comando per eliminare un repository.

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

Output di esempio:

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

Proteggi i repository dall'eliminazione

È possibile impedire l'eliminazione accidentale di un repository includendo una politica di dominio simile alla seguente:

JSON

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

Questa politica impedisce a tutti i responsabili di eliminare il repository, ma se in un secondo momento decidi di dover eliminare il repository, puoi farlo seguendo questi passaggi:

1. Nella politica del dominio, aggiorna la politica nel modo seguente:

   JSON

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

   Sostituisci *repository-arn* con l'ARN del repository che desideri eliminare.

2. Nella AWS CodeArtifact console, scegli Repository ed elimina il repository scelto.

3. Dopo aver eliminato il repository, puoi modificare nuovamente la politica per evitare eliminazioni accidentali.

   JSON

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

```
}
```

In alternativa, puoi includere la stessa dichiarazione di rifiuto in una policy del repository. Ciò consente di disporre di una maggiore flessibilità per proteggere i repository di alto valore dall'eliminazione.

Elenca gli archivi

Usa i comandi in questo argomento per elencare i repository in un AWS account o dominio.

Elenca i repository in un account AWS

Usa questo comando per elencare tutti i repository del tuo AWS account.

```
aws codeartifact list-repositories
```

Output di esempio:

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

È possibile impaginare la risposta `list-repositories` utilizzando i parametri `--max-results` and`--next-token`. Per`--max-results`, specificate un numero intero compreso tra 1 e 1000 per specificare il numero di risultati restituiti in una singola pagina. L'impostazione predefinita è 50. Per tornare alle pagine successive, esegui `list-repositories` nuovamente e passa il `nextToken` valore ricevuto nell'output del comando precedente a`--next-token`. Quando l'`--next-token`opzione non viene utilizzata, viene sempre restituita la prima pagina dei risultati.

Elenca i repository nel dominio

Usa `list-repositories-in-domain` per ottenere un elenco di tutti i repository in un dominio.

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

L'output mostra che alcuni repository sono amministrati da account diversi AWS .

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

È possibile impaginare la risposta `list-repositories-in-domain` utilizzando i parametri `--max-results` and`--next-token`. Per`--max-results`, specificate un numero intero compreso tra 1 e 1000 per specificare il numero di risultati restituiti in una singola pagina. L'impostazione predefinita è 50. Per tornare alle pagine successive, esegui `list-repositories-in-domain` nuovamente e passa il `nextToken` valore ricevuto nell'output del comando precedente a`--next-token`. Quando l'`--next-token`opzione non viene utilizzata, viene sempre restituita la prima pagina dei risultati.

Per visualizzare i nomi dei repository in un elenco più compatto, provate il seguente comando.

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

Output di esempio:

```
repo1
repo2
repo3
```

L'esempio seguente restituisce l'ID dell'account oltre al nome del repository.

```
aws codeartifact list-repositories-in-domain --domain my_domain --domain-
owner 111122223333  \
```

```
  --query 'repositories[*].[name,administratorAccount]' --output text
```

Output di esempio:

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

Per ulteriori informazioni sul `--query` parametro, consulta l'CodeArtifact API ListRepositoriesReference.

Visualizza o modifica la configurazione di un repository

È possibile visualizzare e aggiornare i dettagli del repository utilizzando la CodeArtifact console o il pulsante AWS Command Line Interface ()AWS CLI.

> **Note**
>
> Dopo aver creato un repository, non è possibile modificarne il nome, l' AWS account associato o il dominio.

Argomenti

- Visualizza o modifica la configurazione di un repository (console)
- Visualizza o modifica la configurazione di un repository ()AWS CLI

Visualizza o modifica la configurazione di un repository (console)

È possibile visualizzare i dettagli e aggiornare il repository utilizzando la CodeArtifact console.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Repository, quindi scegli il nome del repository che desideri visualizzare o modificare.
3. Espandi Dettagli per vedere quanto segue:

   - Il dominio del repository. Scegli il nome di dominio per saperne di più.

- La politica delle risorse del repository. Scegli Applica una politica del repository per aggiungerne una.
- L'Amazon Resource Name (ARN) del repository.
- Se il tuo repository dispone di una connessione esterna, puoi scegliere la connessione per saperne di più. Un repository può avere solo una connessione esterna. Per ulteriori informazioni, consulta Connect un CodeArtifact repository a un repository pubblico.
- Se il tuo repository dispone di repository upstream, puoi sceglierne uno per visualizzarne i dettagli. Un repository può avere fino a 10 repository diretti a monte. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.

> **Note**
>
> Un repository può avere una connessione esterna o repository upstream, ma non entrambi.

4. In Pacchetti, puoi vedere tutti i pacchetti disponibili per questo repository. Scegli un pacchetto per saperne di più.
5. Scegli Visualizza le istruzioni di connessione, quindi scegli un gestore di pacchetti con cui imparare a configurarlo CodeArtifact.
6. Scegli Applica policy di repository per aggiornare o aggiungere una policy sulle risorse al tuo repository. Per ulteriori informazioni, consulta Policy del repository.
7. Scegli Modifica per aggiungere o aggiornare quanto segue.

   - La descrizione del repository.
   - Tag associati al repository.
   - Se il repository dispone di una connessione esterna, è possibile modificare l'archivio pubblico a cui si connette. Altrimenti, puoi aggiungere uno o più repository esistenti come repository upstream. Disponili nell'ordine in cui desideri che venga data loro la priorità CodeArtifact quando viene richiesto un pacchetto. Per ulteriori informazioni, consulta Ordine di priorità del repository upstream.

Visualizza o modifica la configurazione di un repository ()AWS CLI

Per visualizzare la configurazione corrente di un repository in CodeArtifact, usa il `describe-repository` comando.

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

Output di esempio:

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

Modifica la configurazione upstream di un repository

Un repository upstream consente a un client di gestione dei pacchetti di accedere ai pacchetti contenuti in più di un repository utilizzando un singolo endpoint URL. Per aggiungere o modificare la relazione upstream di un repository, usa il comando. `update-repository`

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

Output di esempio:

```
{
    "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**
>
> Per aggiungere un repository upstream, è necessario disporre dell'autorizzazione per l'`AssociateWithDownstreamRepository`azione sul repository upstream.

Per rimuovere la relazione upstream di un repository, utilizzate un elenco vuoto come argomento dell'opzione. `--upstreams`

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

Output di esempio:

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

Policy del repository

CodeArtifact utilizza autorizzazioni basate sulle risorse per controllare l'accesso. Le autorizzazioni basate sulle risorse ti consentono di specificare chi ha accesso a un repository e quali operazioni può eseguire su di esso. Per impostazione predefinita, solo il proprietario ha accesso a un repository. Puoi applicare un documento di policy che consenta ad altri responsabili IAM di accedere al tuo repository.

Per ulteriori informazioni, consulta Politiche basate sulle risorse e Politiche basate sull'identità e Politiche basate sulle risorse.

Crea una politica delle risorse per concedere l'accesso in lettura

Una politica delle risorse è un file di testo in formato JSON. Il file deve specificare un principale (attore), una o più azioni e un effetto (`AllowoDeny`). Ad esempio, la seguente politica in materia di risorse concede all'account l'`123456789012`autorizzazione a scaricare i pacchetti dal repository.

JSON

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

Poiché la policy viene valutata solo per le operazioni sul repository a cui è collegata, non è necessario specificare una risorsa. Poiché la risorsa è implicita, è possibile impostarla su. `Resource` * Affinché un gestore di pacchetti possa scaricare un pacchetto da questo repository, sarà inoltre necessario creare una politica di dominio per l'accesso tra account diversi. La politica del dominio deve concedere almeno `codeartifact:GetAuthorizationToken` l'autorizzazione al principale. Per un esempio di una politica di dominio completa per la concessione dell'accesso a più account, consulta questo. Esempio di policy di dominio

> **Note**
>
> L'`codeartifact:ReadFromRepository`azione può essere utilizzata solo su una risorsa del repository. Non puoi inserire l'Amazon Resource Name (ARN) di un pacchetto come

risorsa con `codeartifact:ReadFromRepository` l'azione per consentire l'accesso in lettura a un sottoinsieme di pacchetti in un repository. Un determinato principale può leggere tutti i pacchetti in un repository o nessuno di essi.

Poiché l'unica azione specificata nel repository è `ReadFromRepository` che gli utenti e i ruoli dell'account `1234567890` possono scaricare i pacchetti dal repository. Tuttavia, non possono eseguire altre azioni su di essi (ad esempio, elencare i nomi e le versioni dei pacchetti). In genere, si concedono le autorizzazioni nella seguente politica, oltre al `ReadFromRepository` fatto che un utente che scarica pacchetti da un repository deve interagire con esso anche in altri modi.

JSON

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

Imposta una politica

Dopo aver creato un documento di policy, utilizzate il `put-repository-permissions-policy` comando per allegarlo a un repository:

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

Quando si chiama`put-repository-permissions-policy`, la politica delle risorse sul repository viene ignorata durante la valutazione delle autorizzazioni. Ciò garantisce che il proprietario di un dominio non possa bloccarsi dall'archivio, il che gli impedirebbe di aggiornare la politica delle risorse.

> **Note**
>
> Non è possibile concedere le autorizzazioni a un altro AWS account per aggiornare la politica delle risorse su un repository utilizzando una politica delle risorse, poiché la politica delle risorse viene ignorata durante la chiamata. put-repository-permissions-policy

Output di esempio:

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

L'output del comando contiene l'Amazon Resource Name (ARN) della risorsa del repository, il contenuto completo del documento di policy e un identificatore di revisione. Puoi passare l'identificatore di revisione all'utilizzo dell'opzione. `put-repository-permissions-policy --policy-revision` Ciò garantisce che una revisione nota del documento venga sovrascritta e non una versione più recente impostata da un altro autore.

Leggi una politica

Usa il `get-repository-permissions-policy` comando per leggere una versione esistente di un documento di policy. Per formattare l'output in modo da renderlo leggibile, usa `--output` and `--query policy.document` insieme al modulo `json.tool` Python.

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

Output di esempio:

JSON

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

Eliminazione di una policy

Utilizzate il `delete-repository-permissions-policy` comando per eliminare una policy da un repository.

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

Il formato dell'output è lo stesso del `get-repository-permissions-policy` comando.

Concedi l'accesso in lettura ai principali

Quando si specifica l'utente root di un account come principale in un documento di policy, si concede l'accesso a tutti gli utenti e i ruoli di quell'account. Per limitare l'accesso a utenti o ruoli selezionati, utilizza il relativo ARN nella `Principal` sezione della policy. Ad esempio, utilizza quanto segue per concedere l'accesso bob in lettura all'account `123456789012` utente IAM.

JSON

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

Concedi l'accesso in scrittura ai pacchetti

L'`codeartifact:PublishPackageVersion`azione viene utilizzata per controllare l'autorizzazione a pubblicare nuove versioni di un pacchetto. La risorsa utilizzata con questa azione deve essere un pacchetto. Il formato del CodeArtifact pacchetto ARNs è il seguente.

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

L'esempio seguente mostra l'ARN per un pacchetto npm con ambito `@parity` e nome `ui` nel `my_repo` repository in domain. `my_domain`

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

L'ARN per un pacchetto npm senza ambito ha la stringa vuota per il campo namespace. Ad esempio, quanto segue è l'ARN per un pacchetto senza ambito e con nome `react` nel `my_repo` repository nel dominio. `my_domain`

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

La seguente politica concede all'account `123456789012` l'autorizzazione a pubblicare versioni di `@parity/ui` nel repository. `my_repo`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:PublishPackageVersion"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "arn:aws:codeartifact:us-east-1:111122223333:package/my_domain/my_repo/npm/parity/ui"
        }
```

```
    ]
}
```

> **Important**
>
> Per concedere l'autorizzazione alla pubblicazione di Maven e delle versioni dei NuGet pacchetti, aggiungi le seguenti autorizzazioni oltre a. `codeartifact:PublishPackageVersion`
>
> 1. NuGet: `codeartifact:ReadFromRepository` e specifica la risorsa del repository
> 2. Maven: `codeartifact:PutPackageMetadata`

Poiché questa politica specifica un dominio e un repository come parte della risorsa, consente la pubblicazione solo se collegati a quel repository.

Concedi l'accesso in scrittura a un repository

È possibile utilizzare i caratteri jolly per concedere il permesso di scrittura per tutti i pacchetti in un repository. Ad esempio, utilizzate la seguente politica per concedere a un account il permesso di scrivere su tutti i pacchetti del `my_repo` repository.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codeartifact:PublishPackageVersion"
            ],
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "arn:aws:codeartifact:us-
east-1:111122223333:package/my_domain/my_repo/*"
        }
    ]
```

```
}
```

Interazione tra le politiche del repository e del dominio

CodeArtifact supporta le politiche delle risorse su domini e repository. Le politiche relative alle risorse sono facoltative. Ogni dominio può avere una politica e ogni repository del dominio può avere una propria politica di repository. Se sono presenti sia una policy di dominio che una policy di repository, entrambe vengono valutate per determinare se una richiesta a un CodeArtifact repository è consentita o rifiutata. Le politiche di dominio e di repository vengono valutate utilizzando le seguenti regole:

- Nessuna politica delle risorse viene valutata quando si eseguono operazioni a livello di account come o. ListDomainsListRepositories
- Non viene valutata alcuna politica di repository quando si eseguono operazioni a livello di dominio come o. DescribeDomainListRepositoriesInDomain
- La politica del dominio non viene valutata durante l'esecuzione. PutDomainPermissionsPolicy Tieni presente che questa regola impedisce i blocchi.
- La politica del dominio viene valutata durante l'esecuzione PutRepositoryPermissionsPolicy, ma la politica del repository non viene valutata.
- Un rifiuto esplicito in qualsiasi policy ha la precedenza su un permesso in un'altra policy.
- Un'autorizzazione esplicita è richiesta solo in una politica delle risorse. L'omissione di un'azione da una policy di repository non comporterà un rifiuto implicito se la policy del dominio consente l'azione.
- Quando nessuna policy relativa alle risorse consente un'azione, il risultato è un rifiuto implicito, a meno che l'account del principale chiamante non sia l'account del proprietario del dominio o dell'amministratore del repository e una politica basata sull'identità consenta l'azione.

Le politiche relative alle risorse sono facoltative se utilizzate per concedere l'accesso in uno scenario con account singolo, in cui l'account chiamante utilizzato per accedere a un repository è lo stesso del proprietario del dominio e dell'account dell'amministratore del repository. Le politiche relative alle risorse sono necessarie per concedere l'accesso in uno scenario con più account in cui l'account del chiamante non è lo stesso del proprietario del dominio o dell'account dell'amministratore del repository. L'accesso tra account in CodeArtifact segue le regole generali IAM per l'accesso tra account diversi, come descritto in Determinare se una richiesta tra account è consentita nella Guida per l'utente IAM.

- A un titolare dell'account del proprietario del dominio può essere concesso l'accesso a qualsiasi archivio del dominio tramite una politica basata sull'identità. Tieni presente che in questo caso, non è richiesta alcuna autorizzazione esplicita in una politica di dominio o di repository.
- A un titolare dell'account del proprietario del dominio può essere concesso l'accesso a qualsiasi repository tramite una politica di dominio o di repository. Tieni presente che in questo caso, non è richiesta alcuna autorizzazione esplicita in una politica basata sull'identità.
- A un responsabile dell'account dell'amministratore del repository può essere concesso l'accesso al repository tramite una politica basata sull'identità. Tieni presente che in questo caso, non è richiesta alcuna autorizzazione esplicita in una politica di dominio o di repository.
- A un principale di un altro account viene concesso l'accesso solo se consentito da almeno una politica delle risorse e da almeno una politica basata sull'identità, senza che alcuna politica neghi esplicitamente l'azione.

Aggiungi un tag a un repository CodeArtifact

I tag sono coppie chiave-valore associate a risorse AWS. Puoi applicare tag ai tuoi repository in. CodeArtifact Per informazioni sull'etichettatura CodeArtifact delle risorse, sui casi d'uso, sui vincoli di chiave e valore dei tag e sui tipi di risorse supportati, consulta. Applicazione di tag alle risorse

È possibile utilizzare la CLI per specificare i tag quando si crea un repository. È possibile utilizzare la console o la CLI per aggiungere o rimuovere tag e aggiornare i valori dei tag in un repository. Puoi aggiungere fino a 50 tag a ciascun repository.

Argomenti

- Archivi di tag (CLI)
- Archivi di tag (console)

Archivi di tag (CLI)

È possibile utilizzare la CLI per gestire i tag del repository.

Argomenti

- Aggiungere tag a un repository (CLI)
- Visualizza i tag per un repository (CLI)
- Modifica dei tag per un repository (CLI)

- Rimuovere i tag da un repository (CLI)

Aggiungere tag a un repository (CLI)

Puoi usare la console o AWS CLI etichettare i repository.

Per aggiungere un tag a un repository al momento della creazione, consulta Creazione di un repository .

In queste fasi, si assume che sia già installata una versione recente della AWS CLI o che sia aggiornata alla versione corrente. Per ulteriori informazioni, consultare Installing the AWS Command Line Interface.

Al terminale o alla riga di comando, eseguire il comando tag-resource, specificando l'ARN (Amazon Resource Name) del repository in cui aggiungere i tag e la chiave e il valore del tag che desideri aggiungere.

> **Note**
>
> Per ottenere l'ARN del repository, esegui il comando: `describe-repository`
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

È possibile aggiungere più di un tag a un repository. Ad esempio, per etichettare un archivio denominato *my_repo* in un dominio denominato *my_domain* con due tag, una chiave di tag denominata *key1* con il valore del tag e una chiave di *value1* tag denominata *key2* con il valore del tag di: *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
```

In caso di successo, questo comando non produce alcun risultato.

Visualizza i tag per un repository (CLI)

Segui questi passaggi per utilizzare per visualizzare AWS CLI i AWS tag di un repository. Se non sono stati aggiunti tag, l'elenco restituito è vuoto.

Dal terminale o dalla riga di comando, esegui il comando list-tags-for-resource.

> **Note**
>
> Per ottenere l'ARN del repository, esegui il comando: `describe-repository`
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Ad esempio, per visualizzare un elenco di chiavi e valori di tag per un repository denominato *my_repo* in un dominio denominato *my_domain* con il valore `arn:aws:codeartifact:`*us-west-2*`:`*111122223333*`:repository/`*my_domain*`/`*my_repo* ARN:

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

Se il comando viene eseguito correttamente, restituisce informazioni simili alle seguenti:

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

Modifica dei tag per un repository (CLI)

Segui questi passaggi per utilizzare per modificare un tag per un repository. AWS CLI È possibile modificare il valore di una chiave esistente o aggiungere un'altra chiave.

Nel terminale o nella riga di comando, esegui il tag-resource comando, specificando l'ARN del repository in cui desideri aggiornare un tag e specifica la chiave del tag e il valore del tag.

> **Note**
>
> Per ottenere l'ARN del repository, esegui il comando: `describe-repository`

```
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
```

In caso di successo, questo comando non produce alcun risultato.

Rimuovere i tag da un repository (CLI)

Segui questi passaggi per utilizzare per AWS CLI rimuovere un tag da un repository.

> **Note**
>
> Se si cancella un repository , tutte le associazioni di tag vengono rimosse dal repository cancellato. Non è necessario rimuovere i tag prima di eliminare un repository.

Nel terminale o nella riga di comando, esegui il untag-resource comando, specificando l'ARN del repository in cui desideri rimuovere i tag e la chiave del tag che desideri rimuovere.

> **Note**
>
> Per ottenere l'ARN del repository, esegui il comando: `describe-repository`
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Ad esempio, per rimuovere più tag su un repository denominato *my_repo* in un dominio denominato *my_domain* con le chiavi dei tag e: *key1 key2*

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

In caso di successo, questo comando non produce alcun risultato. Dopo aver rimosso i tag, è possibile visualizzare i tag rimanenti nel repository utilizzando il `list-tags-for-resource` comando.

Archivi di tag (console)

È possibile utilizzare la console o l'interfaccia a riga di comando per aggiungere tag alle risorse.

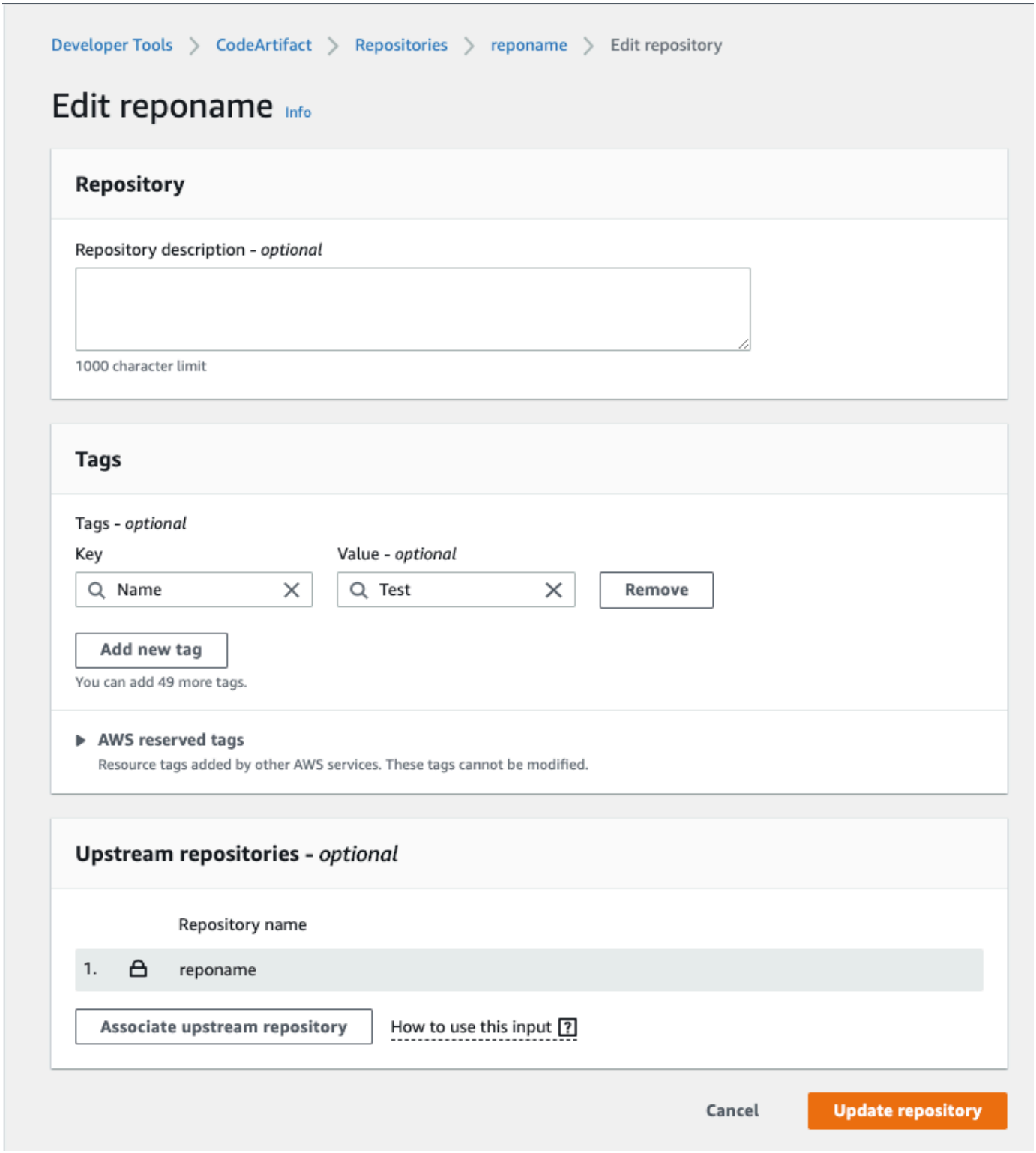
Argomenti

- Aggiungi tag a un repository (console)
- Visualizza i tag per un repository (console)
- Modifica i tag per un repository (console)
- Rimuovi i tag da un repository (console)

Aggiungi tag a un repository (console)

È possibile utilizzare la console per aggiungere tag a un repository esistente.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Repository, scegli il repository a cui vuoi aggiungere i tag.
3. Espandi la sezione Dettagli.
4. In Tag del repository, se non ci sono tag nel repository, scegli Aggiungi tag del repository. Se ci sono tag nel repository, scegli Visualizza e modifica i tag del repository.
5. Scegli Aggiungi nuovo tag.
6. Nei campi Chiave e Valore, inserisci il testo per ogni tag che desideri aggiungere. Il campo Value (Valore) è facoltativo. Ad esempio, in Key (Chiave), immettere **`Name`**. In Valore, immetti **`Test`**.



7. (Facoltativo) Scegliere Add tag (Aggiungi tag) per aggiungere ulteriori righe e inserire più tag.
8. Scegli Aggiorna repository.

Visualizza i tag per un repository (console)

È possibile utilizzare la console per elencare i tag dei repository esistenti.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Repository, scegli il repository in cui desideri visualizzare i tag.
3. Espandi la sezione Dettagli.
4. In Tag del repository, scegli Visualizza e modifica i tag del repository.

> **Note**
>
> Se non ci sono tag aggiunti a questo repository, la console leggerà Aggiungi tag del repository.

Modifica i tag per un repository (console)

È possibile utilizzare la console per modificare i tag che sono stati aggiunti al repository.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Repository, scegli il repository in cui desideri aggiornare i tag.
3. Espandi la sezione Dettagli.
4. In Tag del repository, scegli Visualizza e modifica i tag del repository.

> **Note**
>
> Se non ci sono tag aggiunti a questo repository, la console leggerà Aggiungi tag del repository.

5. Nei campi Key (Chiave) e Value (Valore), aggiornare i valori di ogni campo in base alle esigenze. Ad esempio, per la chiave **`Name`**, in Value (Valore), modificare **`Test`** in **`Prod`**.
6. Scegli Aggiorna repository.

Rimuovi i tag da un repository (console)

È possibile utilizzare la console per eliminare i tag dai repository.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Repository, scegli il repository in cui desideri rimuovere i tag.
3. Espandi la sezione Dettagli.
4. In Tag del repository, scegli Visualizza e modifica i tag del repository.

> **Note**
>
> Se non ci sono tag aggiunti a questo repository, la console leggerà Aggiungi tag del repository.

5. Accanto alla chiave e al valore per ogni tag che desideri eliminare, scegli Rimuovi.
6. Scegli Aggiorna repository.

Lavorare con i repository upstream in CodeArtifact

Un repository può avere altri repository come AWS CodeArtifact repository upstream. Ciò consente a un client di gestione dei pacchetti di accedere ai pacchetti contenuti in più di un repository utilizzando un singolo endpoint del repository.

È possibile aggiungere uno o più repository upstream a un AWS CodeArtifact repository utilizzando, o SDK. Console di gestione AWS AWS CLI Per associare un repository a un repository upstream, è necessario disporre dell'autorizzazione per l'azione sul repository upstream. `AssociateWithDownstreamRepository` Per ulteriori informazioni, consultare Crea un repository con un repository upstream e Aggiungere o rimuovere i repository upstream.

Se un repository upstream ha una connessione esterna a un repository pubblico, i repository che ne derivano possono estrarre i pacchetti da quel repository pubblico. Ad esempio, supponiamo che il repository abbia un repository upstream denominato e `my_repo` `upstream` abbia una connessione esterna a un repository `upstream` npm pubblico. In questo caso, un gestore di pacchetti a cui è collegato `my_repo` può estrarre pacchetti dal repository pubblico npm. Per ulteriori informazioni sulla richiesta di pacchetti da repository upstream o connessioni esterne, vedere o. Richiesta di una versione del pacchetto con repository upstream Richiesta di pacchetti da connessioni esterne

Argomenti

- Qual è la differenza tra gli archivi upstream e le connessioni esterne?
- Aggiungere o rimuovere i repository upstream
- Connect un CodeArtifact repository a un repository pubblico
- Richiesta di una versione del pacchetto con repository upstream
- Richiesta di pacchetti da connessioni esterne
- Ordine di priorità del repository upstream
- Comportamento delle API con i repository upstream

Qual è la differenza tra gli archivi upstream e le connessioni esterne?

In CodeArtifact, i repository upstream e le connessioni esterne si comportano per lo più allo stesso modo, ma ci sono alcune differenze importanti.

1. È possibile aggiungere fino a 10 repository upstream a un repository. CodeArtifact È possibile aggiungere solo una connessione esterna.
2. Esistono chiamate API separate per aggiungere un repository upstream o una connessione esterna.
3. Il comportamento di conservazione dei pacchetti è leggermente diverso, in quanto i pacchetti richiesti dagli archivi upstream vengono conservati in tali repository. Per ulteriori informazioni, consulta Conservazione dei pacchetti in repository intermedi.

Aggiungere o rimuovere i repository upstream

Segui i passaggi indicati nelle seguenti sezioni per aggiungere o rimuovere repository upstream da o verso un repository. CodeArtifact Per ulteriori informazioni sui repository upstream, consulta. Lavorare con i repository upstream in CodeArtifact

Questa guida contiene informazioni sulla configurazione di altri CodeArtifact repository come repository upstream. Per informazioni sulla configurazione di una connessione esterna a repository pubblici come npmjs.com, Nuget Gallery, Maven Central o PyPI, consulta Aggiungere una connessione esterna.

Aggiungi o rimuovi repository upstream (console)

Esegui i passaggi della procedura seguente per aggiungere un repository come repository upstream utilizzando la console. CodeArtifact Per informazioni sull'aggiunta di un repository upstream con, vedere. AWS CLIAggiungi o rimuovi i repository upstream ()AWS CLI

Per aggiungere un repository upstream utilizzando la console CodeArtifact

1. Apri la AWS CodeArtifact console su codeartifact/homehttps://console.aws.amazon.com/codesuite/.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il nome di dominio che contiene il tuo repository.
3. Scegli il nome del tuo repository.
4. Scegli Modifica.
5. Nei repository upstream, scegli Associa repository upstream e aggiungi il repository che desideri aggiungere come repository upstream. Puoi aggiungere repository solo nello stesso dominio dei repository upstream.

6. Scegli Aggiorna repository.

Per rimuovere un repository upstream utilizzando la console CodeArtifact

1. Apri la AWS CodeArtifact console su codeartifact/homehttps://console.aws.amazon.com/codesuite/.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il nome di dominio che contiene il tuo repository.
3. Scegli il nome del tuo repository.
4. Scegli Modifica.
5. Nei repository upstream, trova la voce dell'elenco del repository upstream che desideri rimuovere e scegli Dissocia.

> **Important**
>
> Una volta rimosso un repository upstream da un CodeArtifact repository, i gestori di pacchetti non avranno accesso ai pacchetti nel repository upstream o in nessuno dei suoi repository upstream.

6. Scegliete Aggiorna repository.

Aggiungi o rimuovi i repository upstream ()AWS CLI

È possibile aggiungere o rimuovere i CodeArtifact repository upstream di un repository utilizzando (). AWS Command Line Interface AWS CLI A tale scopo, utilizzate il `update-repository` comando e specificate i repository upstream utilizzando il parametro. `--upstreams`

È possibile aggiungere repository solo nello stesso dominio dei repository upstream.

Per aggiungere repository upstream ()AWS CLI

1. In caso contrario, segui i passaggi indicati Configurazione con AWS CodeArtifact per configurare e configurare il AWS CLI con. CodeArtifact
2. Usa il `aws codeartifact update-repository` comando con il `--upstreams` flag per aggiungere repository upstream.

> **Note**
>
> La chiamata al `update-repository` comando sostituisce i repository upstream configurati esistenti con l'elenco di repository fornito con il flag. `--upstreams` Se desideri aggiungere repository upstream e mantenere quelli esistenti, devi includere i repository upstream esistenti nella chiamata.

Il comando di esempio seguente aggiunge due repository upstream a un repository denominato che si trova in un dominio denominato`my_repo`. `my_domain` L'ordine dei repository upstream nel `--upstreams` parametro determina la priorità di ricerca quando CodeArtifact richiede un pacchetto dal repository. `my_repo` Per ulteriori informazioni, consulta Ordine di priorità del repository upstream.

Per informazioni sulla connessione a repository pubblici ed esterni come npmjs.com o Maven Central, consulta. Connect un CodeArtifact repository a un repository pubblico

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

L'output contiene i repository upstream, come segue.

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

Per rimuovere un repository upstream ()AWS CLI

1. In caso contrario, segui i passaggi indicati Configurazione con AWS CodeArtifact per configurare e configurare il AWS CLI con. CodeArtifact
2. Per rimuovere i repository upstream da un CodeArtifact repository, usa il `update-repository` comando con il flag. `--upstreams` L'elenco dei repository fornito al comando sarà il nuovo set di repository upstream per il repository. CodeArtifact Includi i repository upstream esistenti che desideri conservare e ometti i repository upstream che desideri rimuovere.

   Per rimuovere tutti i repository upstream da un repository, usa il comando e includi senza argomenti. `update-repository` `--upstreams` Quanto segue rimuove i repository upstream da un repository denominato contenuto in un `my_repo` dominio denominato. `my_domain`

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

   L'output mostra che l'elenco di `upstreams` è vuoto.

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

Connect un CodeArtifact repository a un repository pubblico

Puoi aggiungere una connessione esterna tra un repository e un CodeArtifact repository pubblico esterno come https://npmjs.como il repository Maven Central. Quindi, quando richiedi un pacchetto dal CodeArtifact repository che non è già presente nel repository, il pacchetto può essere recuperato dalla connessione esterna. In questo modo è possibile utilizzare le dipendenze open source utilizzate dall'applicazione.

Nel CodeArtifact, il modo previsto di utilizzare le connessioni esterne consiste nell'avere un repository per dominio con una connessione esterna a un determinato archivio pubblico. Ad esempio, se desideri connetterti a npmjs.com, configura un repository nel tuo dominio con una connessione esterna a npmjs.com e configura tutti gli altri repository con una connessione upstream. In questo modo, tutti i repository possono utilizzare i pacchetti che sono già stati recuperati da npmjs.com, anziché recuperarli e archiviarli nuovamente.

Argomenti

- Connect a un repository esterno (console)
- Connect a un repository esterno (CLI)
- Archivi di connessioni esterne supportati
- Rimuovere una connessione esterna (CLI)

Connect a un repository esterno (console)

Quando si utilizza la console per aggiungere una connessione a un repository esterno, si verifica quanto segue:

1. Un `-store` repository per il repository esterno verrà creato nel tuo CodeArtifact dominio se non ne esiste già uno. Questi `-store` repository si comportano come repository intermedi tra il tuo repository e il repository esterno e consentono di connetterti a più di un repository esterno.
2. Il `-store` repository appropriato viene aggiunto come repository a monte del repository.

L'elenco seguente contiene ogni `-store` repository presente CodeArtifact e il rispettivo repository esterno a cui si connettono.

1. `cargo-store`è collegato a crates.io.
2. `clojars-store`è collegato a Clojars Repository.

3. `commonsware-store`è connesso ad Android Repository. CommonsWare
4. `google-android-store`è connesso a Google Android.
5. `gradle-plugins-store`è collegato ai plugin Gradle.
6. `maven-central-store`è collegato a Maven Central Repository.
7. `npm-store`è collegato a npmjs.com.
8. `nuget-store`è collegato a nuget.org.
9. `pypi-store`è collegato alla Python Packaging Authority.
10. `rubygems-store`è connesso RubyGems a.org.

Per connettersi a un repository esterno (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il nome di dominio che contiene il tuo repository.
3. Scegli il nome del tuo repository.
4. Scegli Modifica.
5. Nei repository upstream, scegli Associa repository upstream e aggiungi il repository appropriato collegato come `-store` repository upstream.
6. Scegli Aggiorna repository.

Dopo che il `-store` repository è stato aggiunto come repository upstream, i gestori di pacchetti collegati al repository possono recuperare i pacchetti dal CodeArtifact rispettivo repository esterno.

Connect a un repository esterno (CLI)

È possibile utilizzare il AWS CLI per connettere il CodeArtifact repository a un repository esterno aggiungendo una connessione esterna direttamente al repository. Ciò consentirà agli utenti connessi al CodeArtifact repository, o a uno qualsiasi dei suoi repository a valle, di recuperare i pacchetti dal repository esterno configurato. Ogni CodeArtifact repository può avere solo una connessione esterna.

Si consiglia di avere un repository per dominio con una connessione esterna a un determinato repository pubblico. Per connettere altri repository all'archivio pubblico, aggiungi il repository con la connessione esterna come upstream. Se tu o qualcun altro nel tuo dominio avete già configurato connessioni esterne nella console, è probabile che il dominio disponga già di un `-store` repository

con una connessione esterna all'archivio pubblico a cui desideri connetterti. Per ulteriori informazioni sugli `-store` archivi e sulla connessione con la console, consulta. Connect a un repository esterno (console)

Per aggiungere una connessione esterna a un CodeArtifact repository (CLI)

- Utilizzare `associate-external-connection` per aggiungere una connessione esterna. L'esempio seguente collega un repository al registro pubblico npm, npmjs.com. Per un elenco dei repository esterni supportati, vedere. Archivi di connessioni esterne supportati

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

Output di esempio:

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

Dopo aver aggiunto una connessione esterna, consulta Richiesta di pacchetti da connessioni esterne per informazioni sulla richiesta di pacchetti da un repository esterno con una connessione esterna.

Archivi di connessioni esterne supportati

CodeArtifact supporta una connessione esterna ai seguenti archivi pubblici. Per utilizzare la CodeArtifact CLI per specificare una connessione esterna, utilizzate il valore nella colonna Nome per il `--external-connection` parametro quando eseguite il `associate-external-connection` comando.

Repository type (Tipo di repository)	Descrizione	Nome
Maven	Repository Clojars	`public:maven-clojars`
Maven	CommonsWare Archivio Android	`public:maven-commonsware`
Maven	Archivio Google Android	`public:maven-googleandroid`
Maven	Archivio di plugin Gradle	`public:maven-gradleplugins`
Maven	Maven Central	`public:maven-central`
npm	registro pubblico npm	`public:npmjs`
NuGet	NuGet Galleria	`public:nuget-org`
Python	Indice dei pacchetti Python	`public:pypi`
Ruby	RubyGems.org	`public:ruby-gems-org`
Rust	Crates.io	`public:crates-io`

Rimuovere una connessione esterna (CLI)

Per rimuovere una connessione esterna che è stata aggiunta utilizzando il `associate-external-connection` comando in AWS CLI, utilizzare`disassociate-external-connection`.

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

Output di esempio:

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

Richiesta di una versione del pacchetto con repository upstream

Quando un client (ad esempio, npm) richiede una versione del pacchetto da un CodeArtifact repository denominato `my_repo` che ha più repository upstream, può verificarsi quanto segue:

- Se `my_repo` contiene la versione del pacchetto richiesta, viene restituita al client.
- Se `my_repo` non contiene la versione del pacchetto richiesta, la CodeArtifact cerca nei `my_repo` repository upstream. Se viene trovata la versione del pacchetto, viene copiato un riferimento ad essa e la versione del pacchetto viene restituita al client. `my_repo`
- Se `my_repo` né i relativi repository upstream contengono la versione del pacchetto, al client viene restituita una `Not Found` risposta HTTP 404.

Quando si aggiungono repository upstream utilizzando il `update-repository` comando `create-repository` or, l'ordine in cui vengono passati al `--upstreams` parametro determina la loro

priorità quando viene richiesta una versione del pacchetto. Specificate i repository upstream con `--upstreams` l'ordine che desiderate CodeArtifact utilizzare quando viene richiesta una versione del pacchetto. Per ulteriori informazioni, consulta Ordine di priorità del repository upstream.

Il numero massimo di repository diretti upstream consentito per un repository è 10. Poiché i repository diretti a monte possono avere anche repository diretti a monte propri, è CodeArtifact possibile cercare le versioni dei pacchetti in più di 10 repository. Il numero massimo di repository che vengono cercati quando viene CodeArtifact richiesta una versione del pacchetto è 25.

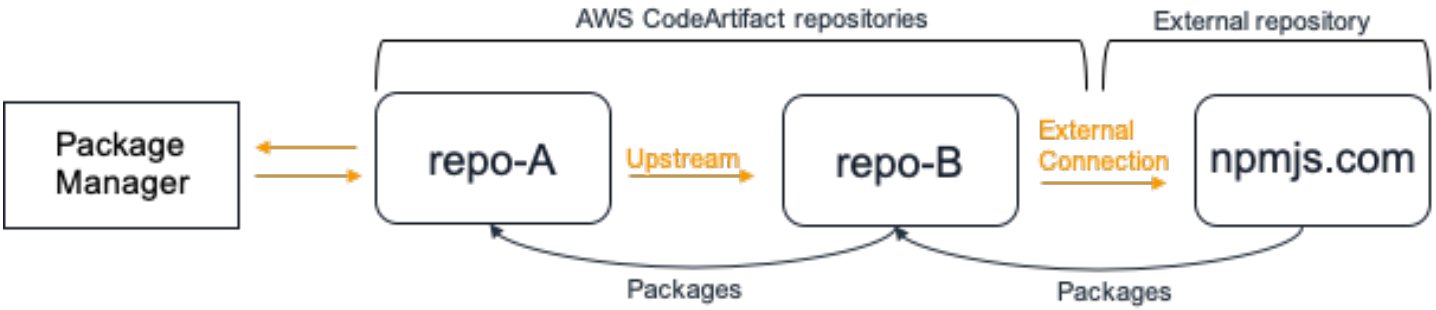
Package retention dai repository upstream

Se una versione del pacchetto richiesta viene trovata in un repository upstream, viene mantenuto un riferimento ad essa ed è sempre disponibile nell'archivio downstream. La versione del pacchetto conservata non è influenzata da nessuno dei seguenti fattori:

- Eliminazione del repository upstream.
- Disconnessione del repository upstream dal repository downstream.
- Eliminazione della versione del pacchetto dal repository upstream.
- Modifica della versione del pacchetto nell'archivio upstream (ad esempio, aggiungendovi una nuova risorsa).

Recupera i pacchetti tramite una relazione a monte

Se un CodeArtifact repository ha una relazione upstream con un repository che dispone di una connessione esterna, le richieste di pacchetti non presenti nel repository upstream vengono copiate dal repository esterno. Ad esempio, considera la seguente configurazione: un repository denominato ha un repository upstream denominato. `repo-A` `repo-B` `repo-B`ha una connessione esterna a. https://npmjs.com



Se npm è configurato per utilizzare il `repo-A` repository, l'esecuzione `npm install` attiva la copia dei pacchetti dall'interno. https://npmjs.com`repo-B` Vengono inoltre inserite le versioni installate. `repo-A` L'esempio seguente install`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
```

Dopo l'esecuzione`npm install`, `repo-A` contiene solo la versione più recente (`lodash 4.17.20`) perché è la versione che è stata recuperata dan`npm`. `repo-A`

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

Output di esempio:

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

Perché `repo-B` dispone di una connessione esterna a https://npmjs.com, tutte le versioni del pacchetto da cui vengono importate https://npmjs.comvengono archiviate in`repo-B`. Queste versioni del pacchetto avrebbero potuto essere recuperate da qualsiasi repository downstream con una relazione a monte con. `repo-B`

Il contenuto di `repo-B` fornisce un modo per vedere tutti i pacchetti e le versioni dei pacchetti importati nel tempo. https://npmjs.com Ad esempio, per vedere tutte le versioni del `lodash` pacchetto importate nel tempo, è possibile utilizzare `list-package-versions` quanto segue.

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

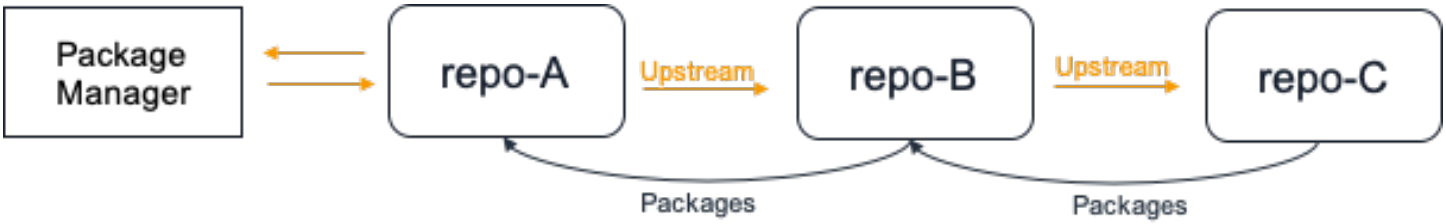
```
            --domain-owner 111122223333 --format npm --package lodash --max-results 5
```

Output di esempio:

```
{
    "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"
}
```

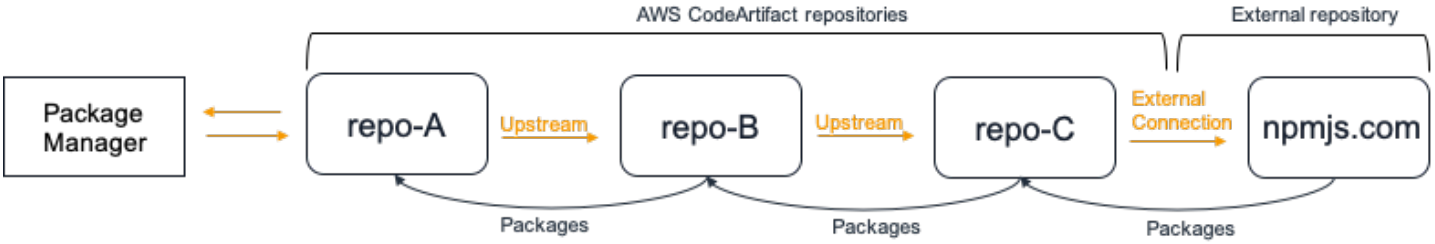
Conservazione dei pacchetti in repository intermedi

CodeArtifact consente il concatenamento di repository upstream. Ad esempio, `repo-A` può avere `repo-B` come upstream e `repo-B` può avere `repo-C` come upstream. Questa configurazione rende le versioni del pacchetto in `repo-B` e `repo-C` disponibili da. `repo-A`



Quando un gestore di pacchetti si connette al repository `repo-A` e recupera una versione del pacchetto dal repository`repo-C`, la versione del pacchetto non viene conservata nell'archivio. `repo-B` La versione del pacchetto verrà conservata solo nell'archivio più a valle, in questo esempio. `repo-A` Non verrà conservata in nessun archivio intermedio. Questo vale anche per le catene più lunghe; ad esempio, se ci fossero quattro repository`repo-A`, `repo-B`repo-C, e `repo-D` e un gestore di pacchetti collegato da cui è stata `repo-A` recuperata una versione del pacchetto`repo-D`, la versione del pacchetto verrebbe conservata in ma non in o. `repo-A` `repo-B` `repo-C`

Il comportamento di conservazione dei pacchetti è simile quando si estrae una versione del pacchetto da un repository esterno, tranne per il fatto che la versione del pacchetto viene sempre conservata nel repository a cui è collegata la connessione esterna. Ad esempio, `repo-A` ha `repo-B` come upstream. `repo-B`ha `repo-C` come upstream e ha `repo-C` anche npmjs.com configurato come connessione esterna; vedere lo schema seguente.



Se un gestore di pacchetti connesso **`repo-A`** richiede una versione del pacchetto, ad esempio lodash 4.17.20, e la versione del pacchetto non è presente in nessuno dei tre repository, verrà recuperata da npmjs.com. Quando lodash 4.17.20 viene recuperato, verrà mantenuto in quanto si tratta del repository più a valle e **`repo-A`** poiché ha la connessione esterna a npmjs.com allegata. **`repo-C`** lodash 4.17.20 non verrà conservato in quanto si tratta di un repository intermedio. `repo-B`

Richiesta di pacchetti da connessioni esterne

Le sezioni seguenti descrivono come richiedere un pacchetto da una connessione esterna e CodeArtifact il comportamento previsto quando si richiede un pacchetto.

Argomenti

- Recupera i pacchetti da una connessione esterna
- Latenza della connessione esterna
- CodeArtifact comportamento quando un repository esterno non è disponibile
- Disponibilità di nuove versioni del pacchetto
- Importazione di versioni di pacchetti con più di una risorsa

Recupera i pacchetti da una connessione esterna

Per recuperare i pacchetti da una connessione esterna dopo averli aggiunti al CodeArtifact repository come descritto inConnect un CodeArtifact repository a un repository pubblico, configurate il gestore di pacchetti in modo che utilizzi il repository e installi i pacchetti.

> **Note**
>
> Le seguenti istruzioni utilizzanonpm, per visualizzare le istruzioni di configurazione e utilizzo per altri tipi di pacchetti, vedereUtilizzo CodeArtifact con Maven, Utilizzo CodeArtifact con NuGet o. Usare CodeArtifact con Python

Per recuperare i pacchetti da una connessione esterna

1. Configura e autentica il tuo gestore di pacchetti con il tuo CodeArtifact repository. Per npm, utilizzare il seguente comando `aws codeartifact login`.

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

2. Richiedi il pacchetto dal repository pubblico. Pernpm, usa il seguente `npm install` comando, sostituendolo *`lodash`* con il pacchetto che desideri installare.

   ```
   npm install lodash
   ```

3. Dopo che il pacchetto è stato copiato nel tuo CodeArtifact repository, puoi usare i `list-package-versions` comandi `list-packages` e per visualizzarlo.

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

Output di esempio:

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

Il `list-package-versions` comando elenca tutte le versioni del pacchetto copiate nel repository. CodeArtifact

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

Output di esempio:

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

Latenza della connessione esterna

Quando si recupera un pacchetto da un archivio pubblico utilizzando una connessione esterna, si verifica un ritardo tra il momento in cui il pacchetto viene recuperato dall'archivio pubblico e il momento in cui viene archiviato nel repository dell'utente. CodeArtifact Ad esempio, supponiamo di aver installato la versione 1.2.5 del pacchetto npm «lodash» come descritto in. Recupera i

pacchetti da una connessione esterna Sebbene il comando `npm install lodash` lodash sia stato completato correttamente, la versione del pacchetto potrebbe non essere ancora presente nel tuo repository. CodeArtifact In genere occorrono circa 3 minuti prima che la versione del pacchetto appaia nel repository, anche se a volte può richiedere più tempo.

A causa di questa latenza, potresti aver recuperato con successo una versione del pacchetto, ma potresti non essere ancora in grado di visualizzare la versione nel tuo repository nella CodeArtifact console o durante la chiamata alle operazioni dell' ListPackages API. ListPackageVersions Una volta CodeArtifact mantenuta in modo asincrono la versione del pacchetto, sarà visibile nella console e tramite richieste API.

CodeArtifact comportamento quando un repository esterno non è disponibile

Occasionalmente, un repository esterno può subire un'interruzione, il che significa che CodeArtifact non può recuperare i pacchetti da esso o che il recupero dei pacchetti è molto più lento del normale. Quando ciò si verifica, le versioni dei pacchetti già estratte da un archivio esterno (ad esempio npmjs.com) e archiviate in un repository continueranno a essere disponibili per il download. CodeArtifact CodeArtifact Tuttavia, i pacchetti che non sono già archiviati in CodeArtifact potrebbero non essere disponibili, anche se è stata configurata una connessione esterna a tale repository. Ad esempio, il tuo CodeArtifact repository potrebbe contenere la versione del pacchetto npm `lodash 4.17.19` perché è quella che hai usato finora nella tua applicazione. Quando desideri eseguire l'aggiornamento a`4.17.20`, normalmente CodeArtifact recupererai la nuova versione da npmjs.com e la memorizzerai nel tuo repository. CodeArtifact Tuttavia, se npmjs.com presenta un'interruzione, questa nuova versione non sarà disponibile. L'unica soluzione è riprovare più tardi, una volta ripristinato npmjs.com.

Le interruzioni del repository esterno possono influire anche sulla pubblicazione di nuove versioni del pacchetto in. CodeArtifact In un repository con una connessione esterna configurata, non CodeArtifact consentirà la pubblicazione di una versione del pacchetto già presente nell'archivio esterno. Per ulteriori informazioni, consulta Panoramica dei pacchetti. Tuttavia, in rari casi, un'interruzione del repository esterno potrebbe significare che CodeArtifact non dispone di up-to-date informazioni su quali pacchetti e versioni dei pacchetti sono presenti in un repository esterno. In questo caso, CodeArtifact potrebbe consentire la pubblicazione di una versione del pacchetto che normalmente negherebbe.

Disponibilità di nuove versioni del pacchetto

Affinché una versione del pacchetto in un archivio pubblico come npmjs.com sia disponibile tramite un CodeArtifact repository, deve prima essere aggiunta a una cache dei metadati dei pacchetti regionali. Questa cache è gestita da CodeArtifact ogni AWS regione e contiene metadati che descrivono il contenuto degli archivi pubblici supportati. A causa di questa cache, c'è un ritardo tra la pubblicazione di una nuova versione del pacchetto in un archivio pubblico e il momento in cui è disponibile da. CodeArtifact Questo ritardo varia in base al tipo di pacchetto.

Per i pacchetti npm, Python e Nuget, potrebbe verificarsi un ritardo fino a 30 minuti dalla pubblicazione di una nuova versione del pacchetto su npmjs.com, pypi.org o nuget.org e dal momento in cui è disponibile per l'installazione da un repository. CodeArtifact CodeArtifact sincronizza automaticamente i metadati di questi due repository per garantire che la cache sia aggiornata.

Per i pacchetti Maven, potrebbe verificarsi un ritardo fino a 3 ore tra la pubblicazione di una nuova versione del pacchetto in un repository pubblico e il momento in cui è disponibile per l'installazione da un repository. CodeArtifact CodeArtifact controllerà la presenza di nuove versioni di un pacchetto al massimo una volta ogni 3 ore. La prima richiesta per un determinato nome di pacchetto dopo la scadenza della durata della cache di 3 ore farà sì che tutte le nuove versioni di quel pacchetto vengano importate nella cache regionale.

Per i pacchetti Maven di uso comune, le nuove versioni vengono in genere importate ogni 3 ore perché l'elevata frequenza di richieste significa che la cache viene spesso aggiornata non appena la durata della cache è scaduta. Per i pacchetti usati di rado, la cache non avrà la versione più recente finché non viene richiesta una versione del pacchetto da un repository. CodeArtifact Alla prima richiesta, saranno disponibili solo le versioni precedentemente importate da CodeArtifact, ma questa richiesta provocherà l'aggiornamento della cache. Nelle richieste successive, le nuove versioni del pacchetto verranno aggiunte alla cache e saranno disponibili per il download.

Importazione di versioni di pacchetti con più di una risorsa

Entrambi i pacchetti Maven e Python possono avere più risorse per versione del pacchetto. Ciò rende l'importazione di pacchetti di questi formati più complessa rispetto a npm e NuGet pacchetti, che hanno solo una risorsa per versione del pacchetto. Per le descrizioni di quali risorse vengono importate per questi tipi di pacchetti e di come vengono rese disponibili le nuove risorse aggiunte, consulta e. Richiesta di pacchetti Python da upstream e connessioni esterne Richiesta di pacchetti Maven da upstream e connessioni esterne

Ordine di priorità del repository upstream

Quando richiedi una versione del pacchetto da un repository con uno o più repository upstream, la loro priorità corrisponde all'ordine in cui erano elencati quando si chiamava il comando or. `create-repository` `update-repository` Quando viene trovata la versione del pacchetto richiesta, la ricerca si interrompe, anche se non è stata effettuata la ricerca in tutti gli archivi upstream. Per ulteriori informazioni, consulta Aggiungi o rimuovi i repository upstream ()AWS CLI.

Usa il `describe-repository` comando per vedere l'ordine di priorità.

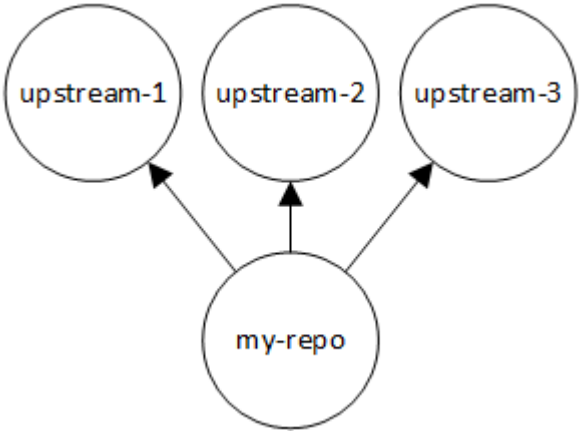
```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-
owner 111122223333
```

Il risultato potrebbe essere il seguente. Mostra che la priorità del repository upstream è la `upstream-1` prima, la `upstream-2` seconda e `upstream-3` la terza.

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

Semplice esempio di ordine di priorità

Nel diagramma seguente, il `my_repo` repository ha tre repository upstream. L'ordine di priorità dei repository upstream è,,. `upstream-1` `upstream-2` `upstream-3`



Una richiesta per una versione del pacchetto `my_repo` cerca nei repository nel seguente ordine finché non viene trovata o finché non viene restituita una risposta HTTP 404 al client: `Not Found`

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

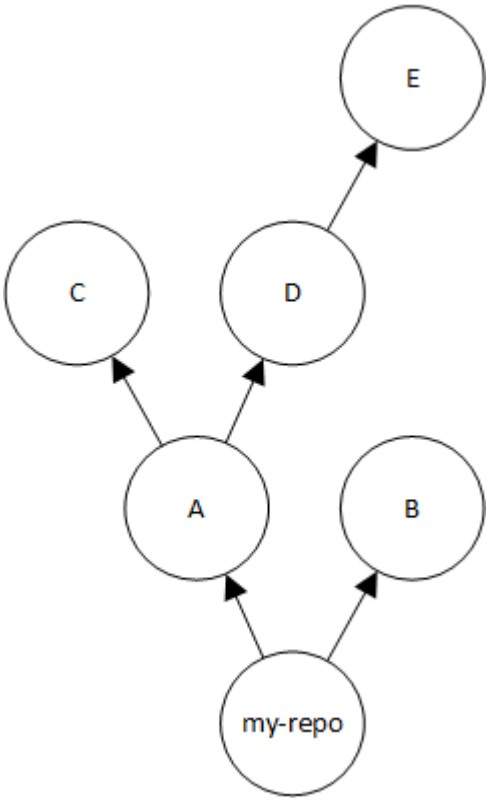
Se viene trovata la versione del pacchetto, la ricerca si interrompe, anche se non è stata cercata in tutti gli archivi upstream. Ad esempio, se la versione del pacchetto viene trovata in`upstream-1`, la ricerca si interrompe e CodeArtifact non viene eseguita la ricerca in `upstream-2` o. `upstream-3`

Quando si utilizza il AWS CLI comando `list-package-versions` per elencare le versioni del pacchetto`my_repo`, viene eseguita la ricerca solo in`my_repo`. Non elenca le versioni dei pacchetti negli archivi upstream.

Esempio di ordine di priorità complesso

Se un repository upstream ha i propri repository upstream, viene utilizzata la stessa logica per trovare una versione del pacchetto prima di passare al successivo repository upstream. Ad esempio, supponiamo che il repository abbia due `my_repo` repository upstream e. `A` `B` Se il repository `A` ha repository upstream, una richiesta per una versione del pacchetto viene `my_repo` dapprima inserita `my_repo`A, poi negli archivi upstream di e così via. `A`

Nel diagramma seguente, il repository contiene repository upstream. `my_repo` Il repository upstream `A` ha due repository upstream e un repository upstream. `D` I repository upstream allo stesso livello del diagramma vengono visualizzati nel loro ordine di priorità, da sinistra a destra (il repository ha un ordine di priorità più elevato rispetto al repository e il repository `A` ha un ordine di priorità più elevato rispetto al repositoryB). `C` `D`



In questo esempio, una richiesta per una versione del pacchetto viene visualizzata nei `my_repo` repository nell'ordine seguente finché non viene trovata o finché un gestore di pacchetti non restituisce una risposta HTTP 404 al client: `Not Found`

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

Comportamento delle API con i repository upstream

Quando richiami determinati CodeArtifact APIs repository collegati a repository upstream, il comportamento può essere diverso a seconda che i pacchetti o le versioni dei pacchetti siano archiviati nel repository di destinazione o nel repository upstream. Il loro comportamento è documentato qui. APIs

Per ulteriori informazioni su CodeArtifact APIs, consulta l'CodeArtifact API Reference.

La maggior parte dei APIs riferimenti a un pacchetto o a una versione del pacchetto restituirà un `ResourceNotFound` errore se la versione del pacchetto specificata non è presente nell'archivio di destinazione. Questo è vero anche se il pacchetto o la versione del pacchetto è presente in un repository upstream. In effetti, i repository upstream vengono ignorati quando li chiamano. APIs Questi sono: APIs

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

Per dimostrare questo comportamento, abbiamo due repository: `target-repo` e`upstream-repo`. `target-repo`è vuoto ed è `upstream-repo` configurato come repository upstream. `upstream-repo`contiene il pacchetto npm. `lodash`

Quando chiamiamo l'`DescribePackageVersion`API on`upstream-repo`, che contiene il `lodash` pacchetto, otteniamo il seguente risultato:

```
{
    "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"
    }
```

Quando chiamiamo la stessa API on`target-repo`, che è vuota ma è `upstream-repo` configurata come upstream, otteniamo il seguente risultato:

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

L'`CopyPackageVersions`API si comporta in modo diverso. Per impostazione predefinita, `CopyPackageVersions` l'API copia solo le versioni del pacchetto archiviate nel repository di destinazione. Se una versione del pacchetto è archiviata nel repository upstream ma non nel repository di destinazione, non verrà copiata. Per includere le versioni dei pacchetti archiviati solo nell'archivio upstream, imposta il valore di `includeFromUpstream` to nella richiesta API. `true`

Per ulteriori informazioni sull'`CopyPackageVersions`API, consulta. Copia i pacchetti tra i repository

Lavorare con i pacchetti in CodeArtifact

I seguenti argomenti mostrano come eseguire azioni sui pacchetti utilizzando la CodeArtifact CLI e l'API.

Argomenti

- Panoramica dei pacchetti
- Elenca i nomi dei pacchetti
- Elenca le versioni dei pacchetti
- Elenca le risorse della versione del pacchetto
- Scarica gli asset della versione del pacchetto
- Copia i pacchetti tra i repository
- Eliminare un pacchetto o una versione del pacchetto
- Visualizza e aggiorna i dettagli e le dipendenze della versione del pacchetto
- Aggiorna lo stato della versione del pacchetto
- Modifica dei controlli di origine dei pacchetti

Panoramica dei pacchetti

Un pacchetto è un insieme di software e metadati necessari per risolvere le dipendenze e installare il software. In CodeArtifact, un pacchetto è costituito da un nome di pacchetto, uno spazio dei nomi opzionale come `@types` in`@types/node`, un set di versioni del pacchetto e metadati a livello di pacchetto come i tag npm.

Indice

- Formati di pacchetto supportati
- Pubblicazione di pacchetti
  - Autorizzazioni di pubblicazione
  - Sovrascrivere le risorse del
  - Pacchetti privati e archivi pubblici
  - Pubblicazione di versioni di pacchetti con patch
  - Limiti di dimensione degli asset per la pubblicazione

  - Latenza di pubblicazione
- Stato della versione del pacchetto
- Nome del pacchetto, versione del pacchetto e normalizzazione del nome dell'asset

Formati di pacchetto supportati

AWS CodeArtifact supporta i formati di pacchetti Cargo, generic, Maven, npm, NuGetPyPI, Ruby, Swift.

Pubblicazione di pacchetti

È possibile pubblicare nuove versioni di qualsiasi formato di pacchetto supportato in un CodeArtifact repository utilizzando strumenti come `npm`,`twine`, `Maven`, `Gradle`, `nuget`, `dotnet`.

Autorizzazioni di pubblicazione

L'utente o il ruolo AWS Identity and Access Management (IAM) deve disporre delle autorizzazioni per la pubblicazione nell'archivio di destinazione. Per pubblicare i pacchetti sono necessarie le seguenti autorizzazioni:

- Carico: `codeartifact:PublishPackageVersion`
- generico: `codeartifact:PublishPackageVersion`
- Maven: e `codeartifact:PublishPackageVersion` `codeartifact:PutPackageMetadata`
- npm: `codeartifact:PublishPackageVersion`
- NuGet: `codeartifact:PublishPackageVersion` e `codeartifact:ReadFromRepository`
- Python: `codeartifact:PublishPackageVersion`
- Ruby: `codeartifact:PublishPackageVersion`
- Rapido: `codeartifact:PublishPackageVersion`

Nell'elenco precedente di autorizzazioni, la policy IAM deve specificare la `package` risorsa per le `codeartifact:PublishPackageVersion` autorizzazioni and. `codeartifact:PutPackageMetadata` Deve inoltre specificare la `repository` risorsa per l'autorizzazione. `codeartifact:ReadFromRepository`

Per ulteriori informazioni sulle autorizzazioni in CodeArtifact, vedereAWS CodeArtifact riferimento alle autorizzazioni.

Sovrascrivere le risorse del

Non è possibile ripubblicare una risorsa del pacchetto già esistente con contenuti diversi. Ad esempio, supponete di aver già pubblicato un pacchetto Maven con una risorsa JAR. `mypackage-1.0.jar` Puoi pubblicare nuovamente quella risorsa solo se il checksum delle risorse vecchie e nuove è identico. Per ripubblicare la stessa risorsa con nuovi contenuti, eliminate prima la versione del pacchetto utilizzando il delete-package-versions comando. Il tentativo di ripubblicare lo stesso nome di risorsa con contenuti diversi genererà un errore di conflitto HTTP 409.

Per i formati di pacchetto che supportano più risorse (generico, PyPI e Maven), puoi aggiungere nuove risorse con nomi diversi a una versione del pacchetto esistente, supponendo che tu disponga delle autorizzazioni richieste. Per i pacchetti generici, potete aggiungere nuove risorse purché la versione del pacchetto sia nello stato in cui si trova. `Unfinished` Poiché npm supporta solo una singola risorsa per versione del pacchetto, per modificare in qualsiasi modo una versione del pacchetto pubblicata, devi prima eliminarla utilizzandodelete-package-versions.

Se provate a ripubblicare una risorsa già esistente (ad esempio`mypackage-1.0.jar`) e il contenuto della risorsa pubblicata e della nuova risorsa sono identici, l'operazione avrà successo perché l'operazione è idempotente.

Pacchetti privati e archivi pubblici

CodeArtifact non pubblica i pacchetti archiviati negli CodeArtifact archivi in archivi pubblici come npmjs.com o Maven Central. CodeArtifact importa pacchetti da repository pubblici a un CodeArtifact repository, ma non sposta mai i pacchetti nella direzione opposta. I pacchetti pubblicati CodeArtifact negli archivi rimangono privati e sono disponibili solo per AWS gli account, i ruoli e gli utenti a cui hai concesso l'accesso.

Pubblicazione di versioni di pacchetti con patch

A volte potresti voler pubblicare una versione modificata del pacchetto, potenzialmente disponibile in un archivio pubblico. Ad esempio, potreste aver trovato un bug in una dipendenza critica dell'applicazione chiamata `mydep 1.1` e dovete correggerlo prima che il fornitore del pacchetto possa esaminare e accettare la modifica. Come descritto in precedenza, CodeArtifact impedisce la pubblicazione `mydep 1.1` nel CodeArtifact repository se l'archivio pubblico è raggiungibile dal repository tramite repository upstream e una CodeArtifact connessione esterna.

Per ovviare a questo problema, pubblica la versione del pacchetto in un CodeArtifact repository diverso in cui l'archivio pubblico non è raggiungibile. Quindi usa l'`copy-package-versions`API per copiare la versione con patch nel CodeArtifact repository `mydep 1.1` da cui la consumerai.

Limiti di dimensione degli asset per la pubblicazione

La dimensione massima di una risorsa del pacchetto che può essere pubblicata è limitata dalla quota massima per le dimensioni del file Asset mostrata inQuote in AWS CodeArtifact. Ad esempio, non potete pubblicare una ruota Maven JAR o Python più grande della dimensione massima del file di asset corrente. Se devi archiviare risorse più grandi in CodeArtifact, richiedi un aumento della quota.

Oltre alla quota massima per la dimensione del file di asset, la dimensione massima di una richiesta di pubblicazione per i pacchetti npm è di 2 GB. Questo limite è indipendente dalla quota massima per la dimensione del file di asset e non può essere aumentato con un aumento della quota. In una richiesta di pubblicazione npm (HTTP PUT), i metadati del pacchetto e il contenuto dell'archivio tar del pacchetto npm sono raggruppati insieme. Per questo motivo, la dimensione massima effettiva di un pacchetto npm che può essere pubblicato varia e dipende dalla dimensione dei metadati inclusi.

> **Note**
>
> I pacchetti npm pubblicati sono limitati a una dimensione massima inferiore a 2 GB.

Latenza di pubblicazione

Le versioni dei pacchetti pubblicate in un CodeArtifact repository sono spesso disponibili per il download in meno di un secondo. Ad esempio, se si pubblica una versione del pacchetto npm su CodeArtifact with`npm publish`, tale versione dovrebbe essere disponibile per un `npm install` comando in meno di un secondo. Tuttavia, la pubblicazione può essere incoerente e talvolta può richiedere più tempo. Se devi usare una versione del pacchetto subito dopo la pubblicazione, usa nuovi tentativi per assicurarti che il download sia affidabile. Ad esempio, dopo aver pubblicato la versione del pacchetto, ripeti il download fino a tre volte se la versione del pacchetto appena pubblicata non è inizialmente disponibile al primo tentativo di download.

> **Note**
>
> L'importazione di una versione del pacchetto da un archivio pubblico richiede in genere più tempo della pubblicazione. Per ulteriori informazioni, consulta Latenza della connessione esterna.

Stato della versione del pacchetto

Ogni versione del pacchetto in CodeArtifact ha uno stato che descrive lo stato corrente e la disponibilità della versione del pacchetto. È possibile modificare lo stato della versione del pacchetto in AWS CLI and SDK. Per ulteriori informazioni, consulta Aggiorna lo stato della versione del pacchetto.

Di seguito sono riportati i valori possibili per lo stato della versione del pacchetto:

- Pubblicato: la versione del pacchetto è stata pubblicata correttamente e può essere richiesta utilizzando un gestore di pacchetti. La versione del pacchetto verrà inclusa negli elenchi delle versioni dei pacchetti restituiti ai gestori di pacchetti, ad esempio, nell'output din`npm view <package-name> versions`. Tutte le risorse della versione del pacchetto sono disponibili nel repository.
- Incompiuto: il client ha caricato una o più risorse per una versione del pacchetto, ma non l'ha finalizzata spostandola nello stato. `Published` Attualmente solo le versioni generiche e del pacchetto Maven possono avere lo stato di. `Unfinished` Per i pacchetti Maven, ciò può verificarsi quando il client carica una o più risorse per una versione del pacchetto ma non pubblica un `maven-metadata.xml` file per il pacchetto che include quella versione. Quando una versione del pacchetto Maven è incompleta, non verrà inclusa negli elenchi di versioni restituiti ai client di questo tipo `mvn` o`gradle`, quindi non può essere utilizzata come parte di una build. I pacchetti generici possono essere mantenuti deliberatamente `Unfinished` nello stato fornendo il `unfinished` flag quando si chiama l'API. PublishPackageVersion Un pacchetto generico può essere modificato in `Published` stato omettendo il `unfinished` flag o chiamando l'UpdatePackageVersionsStatusAPI.
- Non in elenco: le risorse della versione del pacchetto possono essere scaricate dal repository, ma la versione del pacchetto non è inclusa nell'elenco delle versioni restituite ai gestori di pacchetti. Ad esempio, per un pacchetto npm, l'output di non `npm view <package-name> versions` includerà la versione del pacchetto. Ciò significa che la logica di risoluzione delle dipendenze di npm non selezionerà la versione del pacchetto perché la versione non appare nell'elenco delle versioni disponibili. Tuttavia, se la versione del pacchetto Unlisted è già referenziata in un `npm package-lock.json` file, può comunque essere scaricata e installata, ad esempio, durante l'esecuzione. `npm ci`
- Archiviata: le risorse della versione del pacchetto non possono più essere scaricate. La versione del pacchetto non verrà inclusa nell'elenco delle versioni restituite ai gestori di pacchetti. Poiché gli asset non sono disponibili, il consumo della versione del pacchetto da parte dei client è bloccato. Se la build dell'applicazione dipende da una versione aggiornata a Archived, la build si

interromperà, supponendo che la versione del pacchetto non sia stata memorizzata nella cache locale. Non è possibile utilizzare un gestore di pacchetti o uno strumento di compilazione per ripubblicare una versione del pacchetto archiviata perché è ancora presente nel repository, ma è possibile modificare lo stato della versione del pacchetto riportandola a Non elencata o Pubblicata con l'API. UpdatePackageVersionsStatus

- Eliminata: la versione del pacchetto non viene visualizzata negli elenchi e le risorse non possono essere scaricate dall'archivio. La differenza fondamentale tra Disposed e Archived è che con lo stato Disposed, le risorse della versione del pacchetto verranno eliminate definitivamente da. CodeArtifact Per questo motivo, non è possibile spostare una versione del pacchetto da Disposed a Archiviata, Non elencata o Pubblicata. La versione del pacchetto non può più essere utilizzata perché le risorse sono state eliminate. Dopo che una versione del pacchetto è stata contrassegnata come Disposta, non ti verrà più addebitato alcun costo per l'archiviazione delle risorse del pacchetto.

Le versioni del pacchetto di tutti gli stati verranno restituite per impostazione predefinita quando si chiama list-package-versions senza `--status` parametri.

Oltre agli stati elencati in precedenza, è possibile eliminare anche una versione del pacchetto con l'DeletePackageVersionsAPI. Dopo l'eliminazione, una versione del pacchetto non è più presente nell'archivio ed è possibile ripubblicare liberamente quella versione del pacchetto utilizzando un gestore di pacchetti o uno strumento di compilazione. Dopo l'eliminazione di una versione del pacchetto, non ti verrà più addebitato alcun costo per l'archiviazione delle risorse di quella versione del pacchetto.

Nome del pacchetto, versione del pacchetto e normalizzazione del nome dell'asset

CodeArtifact normalizza i nomi dei pacchetti, le versioni dei pacchetti e i nomi delle risorse prima di archiviarli, il che significa che i nomi o le versioni CodeArtifact possono essere diversi dal nome o dalla versione forniti al momento della pubblicazione del pacchetto. Per ulteriori informazioni su come vengono normalizzati i nomi e le versioni CodeArtifact per ogni tipo di pacchetto, consultate la seguente documentazione:

- Normalizzazione dei nomi dei pacchetti in Python
- NuGet normalizzazione del nome del pacchetto, della versione e del nome dell'asset

CodeArtifact non esegue la normalizzazione su altri formati di pacchetti.

Elenca i nomi dei pacchetti

Usa il `list-packages` comando in CodeArtifact per ottenere un elenco di tutti i nomi dei pacchetti in un repository. Questo comando restituisce solo i nomi dei pacchetti, non le versioni.

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

Output di esempio:

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

Elenca i nomi dei pacchetti npm

Per elencare solo i nomi dei pacchetti npm, imposta il valore dell'`--format`opzione su. `npm`

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

Per elencare i pacchetti npm in uno spazio dei nomi (ambito npm), usa le opzioni and. `--namespace` `--format`

Important

Il valore dell'`--namespace`opzione non dovrebbe includere l'iniziale. `@` Per cercare lo spazio dei nomi`@types`, imposta il valore su. *`types`*

Note

L'`--namespace`opzione filtra in base al prefisso del namespace. Qualsiasi pacchetto npm con un ambito che inizia con il valore passato all'`--namespace`opzione verrà restituito nella risposta. `list-packages`

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

Output di esempio:

```
{
    "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"

        }
```

```
    ]
}
```

Elenca i nomi dei pacchetti Maven

Per elencare solo i nomi dei pacchetti Maven, imposta il valore dell'opzione su. `--format` `maven` È inoltre necessario specificare l'ID del gruppo Maven nell'opzione. `--namespace`

> **Note**
>
> L'`--namespace`opzione filtra in base al prefisso dello spazio dei nomi. Qualsiasi pacchetto npm con un ambito che inizia con il valore passato all'`--namespace`opzione verrà restituito nella risposta. `list-packages`

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

Output di esempio:

```
{
    "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"
```

```
        }
    ]
}
```

Elenca i nomi dei pacchetti Python

Per elencare solo i nomi dei pacchetti Python, imposta il valore dell'`--format`opzione su. `pypi`

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

Filtra per prefisso del nome del pacchetto

Per restituire i pacchetti che iniziano con una stringa specificata, puoi usare l'`--package-prefix`opzione.

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

Output di esempio:

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

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

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

```
        }
    ]
}
```

Combinazioni di opzioni di ricerca supportate

È possibile utilizzare le `--package-prefix` opzioni `--format--namespace`, e in qualsiasi combinazione, ad eccezione di quelle che non `--namespace` possono essere utilizzate da sole. La ricerca di tutti i pacchetti npm con un ambito che inizia con `@types` richiede la specificazione dell'`--format`opzione. L'utilizzo `--namespace` da solo genera un errore.

L'utilizzo di nessuna delle tre opzioni è supportato anche da `list-packages` e restituirà tutti i pacchetti di tutti i formati presenti nel repository.

Formatta l'output

È possibile utilizzare parametri disponibili per tutti i AWS CLI comandi per rendere la `list-packages` risposta compatta e più leggibile. Utilizzate il `--query` parametro per specificare il formato di ogni versione del pacchetto restituito. Utilizzate il `--output` parametro per formattare la risposta come testo normale.

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

Output di esempio:

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

Per ulteriori informazioni, consulta Controllo dell'output del comando da AWS CLI nella Guida per l'utente di AWS Command Line Interface .

Impostazioni predefinite e altre opzioni

Per impostazione predefinita, il numero massimo di risultati restituiti da `list-packages` è 100. È possibile modificare questo limite di risultati utilizzando l'`--max-results`opzione.

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

Il valore massimo consentito di `--max-results` è 1.000. Per consentire l'inserimento di pacchetti in repository con più di 1.000 pacchetti, `list-packages` supporta l'impaginazione utilizzando il `nextToken` campo nella risposta. Se il numero di pacchetti nel repository è superiore al valore di`--max-results`, è possibile passare il valore di `nextToken` a un'altra chiamata di per `list-packages` ottenere la pagina successiva di risultati.

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

Elenca le versioni dei pacchetti

Usa il `list-package-versions` comando in AWS CodeArtifact per ottenere un elenco di tutte le versioni di un nome di pacchetto in un repository.

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

Output di esempio:

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

È possibile aggiungere il `--status` parametro alla `list-package-versions` chiamata per filtrare i risultati in base allo stato della versione del pacchetto. Per ulteriori informazioni sullo stato della versione del pacchetto, vedereStato della versione del pacchetto.

È possibile impaginare la risposta `list-package-versions` utilizzando i `--next-token` parametri `--max-results` and. Per`--max-results`, specificate un numero intero compreso tra 1 e 1000 per specificare il numero di risultati restituiti in una singola pagina. L'impostazione predefinita è 50. Per tornare alle pagine successive, esegui `list-package-versions` nuovamente e passa il `nextToken` valore ricevuto nell'output del comando precedente a`--next-token`. Quando l'`--next-token`opzione non viene utilizzata, viene sempre restituita la prima pagina dei risultati.

Il `list-package-versions` comando non elenca le versioni dei pacchetti negli archivi upstream. Tuttavia, vengono elencati i riferimenti alle versioni dei pacchetti in un repository upstream che sono state copiate nel repository durante una richiesta di versione del pacchetto. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.

Elenca le versioni del pacchetto npm

Per elencare tutte le versioni del pacchetto per un pacchetto npm, imposta il valore dell'`--format`opzione su. `npm`

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

Per elencare le versioni del pacchetto npm in uno spazio dei nomi specifico (ambito npm), usa l'opzione. `--namespace` Il valore dell'`--namespace`opzione non deve includere l'iniziale. `@` Per cercare lo spazio dei nomi`@types`, imposta il valore su. *`types`*

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

Elenca le versioni del pacchetto Maven

Per elencare tutte le versioni del pacchetto per un pacchetto Maven, imposta il valore dell'opzione su. `--format` `maven` È inoltre necessario specificare l'ID del gruppo Maven nell'opzione. `--namespace`

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

Ordina le versioni

`list-package-versions`può generare versioni ordinate in ordine decrescente in base alla data di pubblicazione (le versioni pubblicate più di recente vengono elencate per prime). Utilizzate il `--sort-by` parametro con un valore pari a, come segue. `PUBLISHED_TIME`

```
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
```

Output di esempio:

```
{

  "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...."
}
```

Versione di visualizzazione predefinita

Il valore restituito per `defaultDisplayVersion` dipende dal formato del pacchetto:

- Per i pacchetti generici, Maven e PyPI, è la versione del pacchetto pubblicata più di recente.
- Per i pacchetti npm, è la versione a cui fa riferimento il tag. `latest` Se il `latest` tag non è impostato, è la versione del pacchetto pubblicata più di recente.

Formatta l'output

È possibile utilizzare parametri disponibili per tutti i AWS CLI comandi per rendere la `list-package-versions` risposta compatta e più leggibile. Utilizzate il `--query` parametro per specificare il formato di ogni versione del pacchetto restituito. Utilizzate il `--output` parametro per formattare la risposta come testo semplice.

```
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]'
```

Output di esempio:

```
0.1.1
0.1.2
0.1.0
```

```
3.0.0
```

Per ulteriori informazioni, consultate Controllare l'output dei AWS CLI comandi dalla Guida AWS Command Line Interface per l'utente.

Elenca le risorse della versione del pacchetto

Una risorsa è un singolo file (ad esempio, un file npm o un `.tgz` file Maven POM o JAR) archiviato in CodeArtifact e associato a una versione del pacchetto. È possibile utilizzare il `list-package-version-assets` comando per elencare le risorse in ogni versione del pacchetto.

Esegui il `list-package-version-assets` comando per restituire le seguenti informazioni su ogni risorsa del tuo AWS account e della tua AWS regione attuale:

- Il suo nome.
- Le sue dimensioni, in byte.
- Un insieme di valori hash utilizzati per la convalida dei checksum.

Ad esempio, utilizzate il comando seguente per elencare gli asset del pacchetto Python`flatten-json`, version. `0.1.7`

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

Di seguito è riportato l'output.

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

Elenca gli asset di un pacchetto npm

Un pacchetto npm ha sempre una singola risorsa con un nome di. `package.tgz` Per elencare le risorse di un pacchetto npm con ambito, includi l'ambito nell'opzione. `--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
```

Elenca le risorse di un pacchetto Maven

Per elencare gli asset di un pacchetto Maven, includi il namespace del pacchetto nell'opzione. `--namespace` Per elencare gli asset del pacchetto Maven: `commons-cli:commons-cli`

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

```
  --namespace commons-cli --package-version 1.0
```

Scarica gli asset della versione del pacchetto

Una risorsa è un singolo file (ad esempio, un file npm o un `.tgz` file Maven POM o JAR) archiviato in CodeArtifact e associato a una versione del pacchetto. È possibile scaricare le risorse del pacchetto utilizzando. `get-package-version-assets command` Ciò consente di recuperare le risorse senza utilizzare un client di gestione dei pacchetti come npm o`pip`. Per scaricare una risorsa è necessario fornire il nome della risorsa che può essere ottenuto utilizzando il `list-package-version-assets` comando, per ulteriori informazioni, vedereElenca le risorse della versione del pacchetto. La risorsa verrà scaricata nella memoria locale con un nome di file specificato dall'utente.

L'esempio seguente scarica la *guava-27.1-jre.jar* risorsa dal pacchetto Maven *com.google.guava:guava* con versione. *27.1-jre*

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

In questo esempio, il nome del file è stato specificato come *guava-27.1-jre.jar* nell'ultimo argomento del comando precedente, quindi la risorsa scaricata verrà denominata. *guava-27.1-jre.jar*

L'output del comando sarà:

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

> **Note**
>
> Per scaricare risorse da un pacchetto npm con ambito, includi l'ambito nell'`--namespace`opzione. Il @ simbolo deve essere omesso durante l'utilizzo. `--namespace` Ad esempio, se l'ambito è`@types`, usa`--namespace types`.

Il download di risorse utilizzando `get-package-version-asset` richiede l'`codeartifact:GetPackageVersionAsset`autorizzazione sulla risorsa del pacchetto. Per ulteriori informazioni sulle politiche di autorizzazione basate sulle risorse, consultate Politiche basate sulle risorse nella Guida per l'utente.AWS Identity and Access Management

Copia i pacchetti tra i repository

È possibile copiare le versioni dei pacchetti da un repository a un altro in. CodeArtifact Ciò può essere utile per scenari come i flussi di lavoro per la promozione dei pacchetti o la condivisione delle versioni dei pacchetti tra team o progetti. I repository di origine e di destinazione devono trovarsi nello stesso dominio per copiare le versioni dei pacchetti.

Autorizzazioni IAM richieste per copiare i pacchetti

Per copiare le versioni dei pacchetti CodeArtifact, l'utente chiamante deve disporre delle autorizzazioni IAM richieste e la policy basata sulle risorse allegata ai repository di origine e destinazione deve disporre delle autorizzazioni richieste. Per ulteriori informazioni sulle politiche e sugli archivi di autorizzazioni basati sulle risorse, consulta. CodeArtifact Policy del repository

L'utente che chiama `copy-package-versions` deve avere l'`ReadFromRepository`autorizzazione sul repository di origine e l'autorizzazione sul repository di destinazione. `CopyPackageVersions`

Il repository di origine deve disporre dell'`ReadFromRepository`autorizzazione e il repository di destinazione deve avere l'`CopyPackageVersions`autorizzazione assegnata all'account IAM o all'utente che copia i pacchetti. Le seguenti politiche sono esempi di politiche di repository da aggiungere al repository di origine o al repository di destinazione con il comando. `put-repository-permissions-policy` Sostituisci *`111122223333`* con l'ID dell'account chiamante. `copy-package-versions`

> **Note**
>
> La chiamata `put-repository-permissions-policy` sostituirà l'attuale politica del repository, se ne esiste una. Puoi usare il `get-repository-permissions-policy` comando per vedere se esiste una politica, per maggiori informazioni vediLeggi una politica. Se una politica esiste, potresti voler aggiungere queste autorizzazioni ad essa invece di sostituirla.

Esempio di politica di autorizzazione del repository di origine

JSON

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

Esempio di politica di autorizzazione del repository di destinazione

JSON

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

Copia le versioni dei pacchetti

Usa il `copy-package-versions` comando in CodeArtifact per copiare una o più versioni del pacchetto da un repository di origine a un repository di destinazione nello stesso dominio. L'esempio seguente copierà le versioni 6.0.2 e 4.0.0 di un pacchetto npm denominato `my-package` dal repository al repository. `my_repo` `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
```

È possibile copiare più versioni dello stesso nome di pacchetto in un'unica operazione. Per copiare versioni di nomi di pacchetto diversi, è necessario chiamare ciascuno `copy-package-versions` di essi.

Il comando precedente produrrà l'output seguente, supponendo che entrambe le versioni possano essere copiate correttamente.

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

Copia un pacchetto dai repository upstream

Normalmente, cerca nel repository specificato dall'`--source-repository`opzione `copy-package-versions` solo le versioni da copiare. Tuttavia, è possibile copiare le versioni sia dal repository di origine che dai relativi repository upstream utilizzando l'opzione. `--include-from-upstream` Se utilizzi l' CodeArtifact SDK, chiama l'`CopyPackageVersions`API con il

`includeFromUpstream` parametro impostato su true. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.

Copia un pacchetto npm con ambito

Per copiare una versione del pacchetto npm in un ambito, usa l'`--namespace`opzione per specificare l'ambito. Ad esempio, per copiare il pacchetto`@types/react`, usa`--namespace types`. Il `@` simbolo deve essere omesso durante l'utilizzo`--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
```

Copia le versioni del pacchetto Maven

Per copiare le versioni dei pacchetti Maven tra i repository, specifica il pacchetto da copiare passando l'ID del gruppo Maven con l'`--namespace`opzione e il Maven ArtifactID con l'opzione. `--name` Ad esempio, per copiare una singola versione di: `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
```

Se la versione del pacchetto viene copiata correttamente, l'output sarà simile al seguente.

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

Versioni che non esistono nel repository di origine

Se si specifica una versione che non esiste nel repository di origine, la copia avrà esito negativo. Se alcune versioni esistono nel repository di origine e altre no, tutte le versioni non verranno copiate. Nell'esempio seguente, la versione 0.2.0 del pacchetto `array-unique` npm è presente nel repository dei sorgenti, ma la versione 5.6.7 no:

```
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
```

L'output in questo scenario sarà simile al seguente.

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

Il codice `SKIPPED` di errore viene utilizzato per indicare che la versione non è stata copiata nel repository di destinazione perché non è stato possibile copiare un'altra versione.

Versioni già esistenti nel repository di destinazione

Quando una versione del pacchetto viene copiata in un repository in cui esiste già, CodeArtifact confronta le risorse del pacchetto e i metadati a livello di versione del pacchetto nei due repository.

Se le risorse e i metadati della versione del pacchetto sono identici negli archivi di origine e di destinazione, non viene eseguita una copia ma l'operazione viene considerata riuscita. Ciò significa che `copy-package-versions` è idempotente. Quando ciò si verifica, la versione che era già presente nei repository di origine e di destinazione non verrà elencata nell'output di. `copy-package-versions`

Nell'esempio seguente, due versioni del pacchetto npm `array-unique` sono presenti nel repository dei sorgenti. `repo-1` La versione 0.2.1 è presente anche nel repository di destinazione `dest-repo` e la versione 0.2.0 no.

```
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
```

L'output in questo scenario sarà simile al seguente.

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

La versione 0.2.0 è elencata in `successfulVersions` quanto è stata copiata correttamente dal repository di origine a quello di destinazione. La versione 0.2.1 non viene mostrata nell'output in quanto era già presente nell'archivio di destinazione.

Se le risorse o i metadati della versione del pacchetto differiscono negli archivi di origine e di destinazione, l'operazione di copia avrà esito negativo. È possibile utilizzare il `--allow-overwrite` parametro per forzare una sovrascrittura.

Se alcune versioni esistono nel repository di destinazione e altre no, tutte le versioni non verranno copiate. Nell'esempio seguente, la versione 0.3.2 del pacchetto `array-unique` npm è presente sia nell'archivio di origine che in quello di destinazione, ma i contenuti della versione del pacchetto sono diversi. La versione 0.2.1 è presente nel repository di origine ma non in quello di destinazione.

```
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
```

L'output in questo scenario sarà simile al seguente.

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

La versione 0.2.1 è contrassegnata come `SKIPPED` se non fosse stata copiata nel repository di destinazione. Non è stato copiato perché la copia della versione 0.3.2 non è riuscita perché era già presente nel repository di destinazione, ma non era identica nei repository di origine e di destinazione.

Specificare una revisione della versione del pacchetto

Una revisione della versione del pacchetto è una stringa che specifica un insieme specifico di risorse e metadati per una versione del pacchetto. È possibile specificare una revisione della versione del pacchetto per copiare le versioni del pacchetto che si trovano in uno stato specifico. Per specificare una revisione della versione del pacchetto, utilizzate il `--version-revisions` parametro per passare una o più versioni del pacchetto separate da virgole e le coppie di revisione della versione del pacchetto al comando. `copy-package-versions`

> **Note**
>
> È necessario specificare il parametro o con`--versions`. `--version-revisions` `copy-package-versions` Non è possibile specificare entrambi.

L'esempio seguente copierà la versione 0.3.2 del pacchetto solo `my-package` se è presente nell'archivio dei sorgenti con la revisione della versione del pacchetto. `REVISION-1-SAMPLE-6C81EFF7DA55CC`

```
aws codeartifact copy-package-versions --domain my_domain --domain-owner 111122223333
 --source-repository repo-1 \
```

```
 --destination-repository repo-2 --format npm --namespace my-namespace \
 --package my-package --version-revisions 0.3.2=REVISION-1-SAMPLE-6C81EFF7DA55CC
```

L'esempio seguente copia due versioni del pacchettomy-package, 0.3.2 e 0.3.13. La copia avrà successo solo se nel repository di origine la versione 0.3.2 di my-package ha una revisione e la versione 0.3.13 ha una revisioneREVISION-1-SAMPLE-6C81EFF7DA55CC. 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
```

Per trovare le revisioni di una versione del pacchetto, usa il comando o. describe-package-version list-package-versions

Per ulteriori informazioni, consulta le pagine Revisione della versione del pacchetto e CopyPackageVersion nella Documentazione di riferimento dell'API CodeArtifact .

Copia i pacchetti npm

Per ulteriori informazioni sul copy-package-versions comportamento con i pacchetti npm, consulta i tag npm e l'API. CopyPackageVersions

Eliminare un pacchetto o una versione del pacchetto

È possibile eliminare una o più versioni del pacchetto alla volta utilizzando il delete-package-versions comando. Per rimuovere completamente un pacchetto da un repository, incluse tutte le versioni e le configurazioni associate, usa il delete-package comando. Un pacchetto può esistere in un repository senza alcuna versione del pacchetto. Ciò può accadere quando tutte le versioni vengono eliminate utilizzando il delete-package-versions comando o se il pacchetto è stato creato senza alcuna versione utilizzando l'operazione put-package-origin-configuration API (vediModifica dei controlli di origine dei pacchetti).

Argomenti

- Eliminazione di un pacchetto (AWS CLI)
- Eliminazione di un pacchetto (console)

- Eliminazione di una versione del pacchetto ()AWS CLI
- Aggiunta di una versione del pacchetto (console)
- Eliminazione di un pacchetto npm o di una versione del pacchetto
- Eliminazione di un pacchetto Maven o di una versione del pacchetto
- Procedure consigliate per l'eliminazione di pacchetti o versioni di pacchetti

Eliminazione di un pacchetto (AWS CLI)

È possibile eliminare un pacchetto, incluse tutte le versioni e la configurazione del pacchetto, utilizzando il `delete-package` comando. L'esempio seguente elimina il pacchetto PyPI `my-package` denominato nel `my_repo` repository nel dominio: `my_domain`

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

Output di esempio:

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

È possibile confermare che il pacchetto è stato eliminato eseguendo l'esecuzione con lo stesso `describe-package` nome di pacchetto:

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

Eliminazione di un pacchetto (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, selezionare Repositories (Repository).
3. Scegli il repository da cui desideri eliminare un pacchetto.
4. Scegli il Package che desideri eliminare.
5. Scegli Delete Package.

Eliminazione di una versione del pacchetto ()AWS CLI

È possibile eliminare una o più versioni del pacchetto alla volta utilizzando il `delete-package-versions` comando. L'esempio seguente elimina `4.0.0` le `4.0.1` versioni e e `5.0.0` del pacchetto PyPI `my-package` denominato `my_repo` nel `my_domain` dominio:

```
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
```

Output di esempio:

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

È possibile confermare che le versioni sono state eliminate eseguendo l'esecuzione `list-package-versions` con lo stesso nome di pacchetto:

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

Aggiunta di una versione del pacchetto (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, selezionare Repositories (Repository).
3. Scegli il repository da cui desideri eliminare le versioni del pacchetto.
4. Scegli il Package da cui desideri eliminare le versioni.
5. Seleziona la versione del pacchetto che desideri eliminare.
6. Scegli Elimina.

> **Note**
>
> Nella console, puoi eliminare solo una versione del pacchetto alla volta. Per eliminarne più di uno alla volta, usa la CLI.

Eliminazione di un pacchetto npm o di una versione del pacchetto

Per eliminare un pacchetto npm o singole versioni del pacchetto, imposta l'`--format`opzione su. `npm` Per eliminare una versione del pacchetto in un pacchetto npm con ambito, usa l'`--namespace`opzione per specificare l'ambito. Ad esempio, per eliminare il pacchetto`@types/react`, usa. `--namespace types` Omettete il `@` simbolo durante l'uso`--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
```

Per eliminare il pacchetto`@types/react`, incluse tutte le sue versioni:

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

Eliminazione di un pacchetto Maven o di una versione del pacchetto

Per eliminare un pacchetto Maven o singole versioni del pacchetto, imposta l'`--format`opzione su `maven` e specifica il pacchetto da eliminare passando l'ID del gruppo Maven con l'opzione e il Maven ArtifactID con l'`--namespace`opzione. `--name` Ad esempio, quanto segue mostra come eliminare una singola versione di: `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
```

L'esempio seguente mostra come eliminare il pacchetto`com.google.guava:guava`, incluse tutte le sue versioni:

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

Procedure consigliate per l'eliminazione di pacchetti o versioni di pacchetti

Se è necessario eliminare una versione del pacchetto, è consigliabile creare un repository per archiviare una copia di backup della versione del pacchetto che si desidera eliminare. Puoi farlo chiamando prima l'archivio `copy-package-versions` di backup:

```
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
```

Dopo aver copiato la versione del pacchetto, puoi richiamare `delete-package-versions` il pacchetto o la versione del pacchetto che desideri eliminare.

```
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
```

Visualizza e aggiorna i dettagli e le dipendenze della versione del pacchetto

È possibile visualizzare le informazioni sulla versione di un pacchetto, comprese le dipendenze, in. CodeArtifact È inoltre possibile aggiornare lo stato di una versione del pacchetto. Per ulteriori informazioni sullo stato della versione del pacchetto, vedereStato della versione del pacchetto.

Visualizza i dettagli della versione del pacchetto

Utilizzate il `describe-package-version` comando per visualizzare i dettagli sulle versioni dei pacchetti. I dettagli della versione del pacchetto vengono estratti da un pacchetto quando viene pubblicato su. CodeArtifact I dettagli nei diversi pacchetti variano e dipendono dai loro formati e dalla quantità di informazioni che gli autori hanno aggiunto.

La maggior parte delle informazioni nell'output del `describe-package-version` comando dipende dal formato del pacchetto. Ad esempio, `describe-package-version` estrae le informazioni di un pacchetto npm dal relativo `package.json` file. La revisione è stata creata da. CodeArtifact Per ulteriori informazioni, consulta Specificare una revisione della versione del pacchetto.

Due versioni di pacchetto con lo stesso nome possono trovarsi nello stesso repository se ognuna si trova in namespace diversi. Utilizzate il `--namespace` parametro opzionale per specificare uno spazio dei nomi. Per ulteriori informazioni, consulta Visualizza i dettagli della versione del pacchetto npm o Visualizza i dettagli della versione del pacchetto Maven.

L'esempio seguente restituisce dettagli sulla versione `1.9.0` di un pacchetto Python denominato `pyhamcrest` che si trova nel `my_repo` repository.

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

L'output potrebbe essere simile al seguente.

```
{
```

```
  "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 recupera i dettagli della versione del pacchetto, come la home page del pacchetto o le informazioni sulla licenza del pacchetto, dai metadati forniti dall'autore del pacchetto. Se una di queste informazioni supera i 400 KB, che è il limite di dimensione degli elementi di DynamoDB CodeArtifact , non sarà in grado di elaborare tali dati e potresti non visualizzare queste informazioni sulla console o nella risposta di. `describe-package-version` Ad esempio, un pacchetto python come pi. https://py org/project/rapyd-sdk/ ha un campo di licenza molto grande, quindi queste informazioni non verrebbero elaborate da. CodeArtifact

Visualizza i dettagli della versione del pacchetto npm

Per visualizzare i dettagli sulla versione di un pacchetto npm, imposta il valore dell'`--format`opzione su. **`npm`** Facoltativamente, includi lo spazio dei nomi della versione del pacchetto (npm scope) nell'opzione. `--namespace` Il valore dell'`--namespace`opzione non deve includere l'interlinea. `@` Per cercare lo spazio dei nomi`@types`, imposta il valore su. *`types`*

Quanto segue restituisce i dettagli sulla versione `4.41.5` di un pacchetto npm denominato `webpack` nell'ambito. `@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
```

L'output potrebbe essere simile al seguente.


```
{
  "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"
  }
}
```


Visualizza i dettagli della versione del pacchetto Maven

Per visualizzare i dettagli sulla versione di un pacchetto Maven, imposta il valore dell'`--format`opzione su `maven` e includi lo spazio dei nomi della versione del pacchetto nell'opzione. `--namespace`

L'esempio seguente restituisce i dettagli sulla versione `1.2` di un pacchetto Maven denominato `commons-rng-client-api` che si trova nello spazio dei nomi e nel repository. `org.apache.commons` `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
```

L'output potrebbe essere simile al seguente.

```
{
  "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 non estrae le informazioni dettagliate sulla versione del pacchetto dai file POM principali. I metadati per una determinata versione del pacchetto includeranno solo le informazioni nel POM relative a quella versione esatta del pacchetto, non al POM principale o a qualsiasi altro POM a cui si fa riferimento transitivamente utilizzando il tag POM. `parent` Ciò significa che l'output di `describe-package-version` ometterà i metadati (come le informazioni sulla licenza) per le versioni del pacchetto Maven che si basano su un riferimento per contenere questi metadati. `parent`

Visualizza le dipendenze delle versioni del pacchetto

Usa il `list-package-version-dependencies` comando per ottenere un elenco delle dipendenze della versione del pacchetto. Il comando seguente elenca le dipendenze di un pacchetto npm denominato`my-package`, version`4.41.5`, nel `my_repo` repository, nel dominio. `my_domain`

```
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
```

L'output potrebbe essere simile al seguente.

```
{
  "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"
}
```

Per l'intervallo di valori supportati per il campo DependencyType, consulta il tipo di PackageDependencydati nell'API. CodeArtifact

Visualizza il file readme della versione del pacchetto

Alcuni formati di pacchetto, come npm, includono un `README` file. Usa il `get-package-version-readme` per ottenere il `README` file di una versione del pacchetto. Il comando seguente restituisce il `README` file di un pacchetto npm denominato`my-package`, version`4.41.5`, nel `my_repo` repository, nel `my_domain` dominio.

> **Note**
>
> CodeArtifact non supporta la visualizzazione di file readme da pacchetti generici o Maven.

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

L'output potrebbe essere simile al seguente.

```
{
  "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"
}
```

Aggiorna lo stato della versione del pacchetto

Ogni versione del pacchetto in CodeArtifact ha uno stato che descrive lo stato corrente e la disponibilità della versione del pacchetto. È possibile modificare lo stato della versione del pacchetto utilizzando AWS CLI sia la console che la console.

> **Note**
>
> Per ulteriori informazioni sullo stato della versione del pacchetto, incluso un elenco degli stati disponibili, vedereStato della versione del pacchetto.

Aggiornamento dello stato della versione del pacchetto

L'impostazione dello stato di una versione del pacchetto consente di controllare come una versione del pacchetto può essere utilizzata senza eliminarla completamente dal repository. Ad esempio, quando la versione di un pacchetto ha lo stato di`Unlisted`, può comunque essere scaricata normalmente, ma non verrà visualizzata negli elenchi delle versioni del pacchetto restituiti a comandi come. `npm view` L'UpdatePackageVersionsStatus API consente di impostare lo stato della versione del pacchetto di più versioni dello stesso pacchetto in un'unica chiamata API. Per una descrizione dei diversi stati, vederePanoramica dei pacchetti.

Utilizzate il `update-package-versions-status` comando per modificare lo stato di una versione del pacchetto in `Published``Unlisted`, o`Archived`. Per visualizzare le autorizzazioni

IAM necessarie per utilizzare il comando, consultaAutorizzazioni IAM richieste per aggiornare lo stato della versione di un pacchetto. L'esempio seguente imposta lo stato della versione 4.1.0 del pacchetto npm su. `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
```

Output di esempio:

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

Questo esempio utilizza un pacchetto npm, ma il comando funziona in modo identico per altri formati. È possibile spostare più versioni allo stesso stato di destinazione utilizzando un singolo comando, vedi l'esempio seguente.

```
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
```

Output di esempio:

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

```
    },
    "failedVersions": {}
}
```

Nota che una volta pubblicata, una versione del pacchetto non può essere riportata allo `Unfinished` stato, quindi questo stato non è consentito come valore per il `--target-status` parametro. Per spostare la versione del pacchetto nello `Disposed` stato, utilizzate invece il `dispose-package-versions` comando come descritto di seguito.

Autorizzazioni IAM richieste per aggiornare lo stato della versione di un pacchetto

`update-package-versions-status`Per richiedere un pacchetto, è necessario disporre dell'`codeartifact:UpdatePackageVersionsStatus`autorizzazione sulla risorsa del pacchetto. Ciò significa che è possibile concedere l'autorizzazione alla chiamata `update-package-versions-status` in base al pacchetto. Ad esempio, una policy IAM che conceda il permesso di chiamare `update-package-versions-status` il pacchetto npm *`chalk`* includerebbe un'istruzione come la seguente.

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

Aggiornamento dello stato di un pacchetto npm con ambito

Per aggiornare lo stato della versione del pacchetto di una versione del pacchetto npm con un ambito, usa il parametro. `--namespace` Ad esempio, per rimuovere la versione 8.0.0 di`@nestjs/core`, utilizzate il comando seguente.

```
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
```

Aggiornamento dello stato di un pacchetto Maven

I pacchetti Maven hanno sempre un ID di gruppo, che viene chiamato namespace in. CodeArtifact Usa il `--namespace` parametro per specificare l'ID del gruppo Maven durante la chiamata. `update-package-versions-status` Ad esempio, per archiviare la versione 2.13.1 del pacchetto `org.apache.logging.log4j:log4j` Maven, utilizzare il comando seguente.

```
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
```

Specificare una revisione della versione del pacchetto

Una revisione della versione del pacchetto è una stringa che specifica un insieme specifico di risorse e metadati per una versione del pacchetto. È possibile specificare una revisione della versione del pacchetto per aggiornare lo stato delle versioni del pacchetto che si trovano in uno stato specifico. Per specificare una revisione della versione del pacchetto, utilizzate il `--version-revisions` parametro per passare una o più versioni del pacchetto separate da virgole e le coppie di revisione delle versioni del pacchetto. Lo stato di una versione del pacchetto verrà aggiornato solo se la revisione corrente della versione del pacchetto corrisponde al valore specificato.

> **Note**
>
> Il `—-versions` parametro deve essere definito anche quando si utilizza il `--version-revisions` parametro.

```
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
```

Per aggiornare più versioni con un solo comando, passate un elenco separato da virgole di coppie di versioni e revisioni di versione alle opzioni. `--version-revisions` Il comando di esempio seguente definisce due diverse coppie di versione del pacchetto e di revisione della versione del pacchetto.

```
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
```

Output di esempio:

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

Quando si aggiornano più versioni del pacchetto, le versioni passate `--version-revisions` devono essere le stesse a `--versions` cui è stato passato. Se una revisione viene specificata in modo errato, lo stato di quella versione non verrà aggiornato.

Utilizzo del parametro di stato previsto

Il `update-package-versions-status` comando fornisce il `--expected-status` parametro che supporta la specificazione dello stato corrente previsto di una versione del pacchetto. Se lo stato corrente non corrisponde al valore passato`--expected-status`, lo stato di quella versione del pacchetto non verrà aggiornato.

Ad esempio*my_repo*, nelle versioni 4.0.0 e 4.1.0 del pacchetto npm `chalk` attualmente hanno uno stato di. `Published` Una chiamata a `update-package-versions-status` ciò specifica uno stato previsto di non `Unlisted` riuscirà ad aggiornare entrambe le versioni del pacchetto a causa della mancata corrispondenza dello stato.

```
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
```

Output di esempio:

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

Errori con le singole versioni del pacchetto

Esistono diversi motivi per cui lo stato di una versione del pacchetto non viene aggiornato durante la chiamataupdate-package-versions-status. Ad esempio, la revisione della versione del pacchetto potrebbe essere stata specificata in modo errato o lo stato previsto non corrisponde allo stato corrente. In questi casi, la versione verrà inclusa nella failedVersions mappa nella risposta dell'API. Se una versione fallisce, è possibile che le altre versioni specificate nella stessa chiamata a update-package-versions-status vengano ignorate e il loro stato non sia aggiornato. Tali versioni verranno incluse anche nella failedVersions mappa con un errorCode diSKIPPED.

Nell'attuale implementazione diupdate-package-versions-status, se non è possibile modificare lo stato di una o più versioni, tutte le altre versioni verranno ignorate. Cioè, tutte le versioni vengono aggiornate correttamente o nessuna versione viene aggiornata. Questo comportamento non è garantito nel contratto API; in futuro, alcune versioni potrebbero avere successo mentre altre versioni falliranno in una singola chiamata aupdate-package-versions-status.

Il comando di esempio seguente include un errore di aggiornamento dello stato della versione causato da una mancata corrispondenza della versione della versione del pacchetto. Questo errore di aggiornamento fa sì che un'altra chiamata di aggiornamento dello stato della versione venga ignorata.

```
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
```

Output di esempio:

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

Eliminazione delle versioni dei pacchetti

Lo stato del `Disposed` pacchetto ha un comportamento simile a`Archived`, tranne per il fatto che gli asset del pacchetto verranno eliminati definitivamente, CodeArtifact in modo che all'account del proprietario del dominio non venga più addebitato lo spazio di archiviazione degli asset. Per ulteriori informazioni sullo stato di ogni versione del pacchetto, consultaStato della versione del pacchetto. Per modificare lo stato di una versione del pacchetto in`Disposed`, usa il `dispose-package-versions` comando. Questa funzionalità è distinta dal `update-package-versions-status` fatto che lo smaltimento di una versione del pacchetto non è reversibile. Poiché le risorse del pacchetto verranno eliminate, lo stato della versione non può essere ripristinato in `Archived``Unlisted`, o. `Published` L'unica azione che può essere eseguita su una versione del pacchetto che è stata eliminata è eliminarla utilizzando il `delete-package-versions` comando.

Per chiamare `dispose-package-versions` correttamente, il principale IAM chiamante deve disporre dell'`codeartifact:DisposePackageVersions`autorizzazione sulla risorsa del pacchetto.

Il comportamento del `dispose-package-versions` comando è simile`update-package-versions-status`, incluso il comportamento delle `--expected-status` opzioni `--version-`

`revisions` e descritte nelle sezioni sulla revisione della versione e sullo stato previsto. Ad esempio, il comando seguente tenta di eliminare una versione del pacchetto ma fallisce a causa di uno stato previsto non corrispondente.

```
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
```

Output di esempio:

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

Se lo stesso comando viene eseguito nuovamente con un `--expected-status` of`Published`, l'eliminazione avrà esito positivo.

```
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
```

Output di esempio:

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

Modifica dei controlli di origine dei pacchetti

In AWS CodeArtifact, le versioni dei pacchetti possono essere aggiunte a un repository pubblicandole direttamente, estraendole da un repository upstream o importandole da un archivio pubblico esterno. Consentire l'aggiunta di versioni di pacchetto di un pacchetto sia mediante la pubblicazione diretta che l'importazione da archivi pubblici rende vulnerabili a un attacco di sostituzione delle dipendenze. Per ulteriori informazioni, consulta Attacchi di sostituzione delle dipendenze. Per proteggersi da un attacco di sostituzione delle dipendenze, è possibile configurare i controlli di origine dei pacchetti su un pacchetto in un repository per limitare il modo in cui le versioni di quel pacchetto possono essere aggiunte al repository.

La configurazione dei controlli di origine dei pacchetti dovrebbe essere presa in considerazione da qualsiasi team che desideri consentire alle nuove versioni di pacchetti diversi di provenire sia da fonti interne, come la pubblicazione diretta, sia da fonti esterne, come gli archivi pubblici. Per impostazione predefinita, i controlli di origine dei pacchetti verranno configurati in base al modo in cui la prima versione di un pacchetto viene aggiunta all'archivio. Per informazioni sulle impostazioni del controllo dell'origine del pacchetto e sui relativi valori predefiniti, vedereImpostazioni di controllo dell'origine del pacchetto.

Per rimuovere il record del pacchetto dopo aver utilizzato l'operazione `put-package-origin-configuration` API, utilizzare `delete-package` (vediEliminare un pacchetto o una versione del pacchetto).

Scenari comuni di controllo dell'accesso ai pacchetti

Questa sezione include alcuni scenari comuni in cui una versione del pacchetto viene aggiunta a un CodeArtifact repository. Le impostazioni di controllo dell'origine dei pacchetti verranno impostate per i nuovi pacchetti a seconda di come viene aggiunta la prima versione del pacchetto.

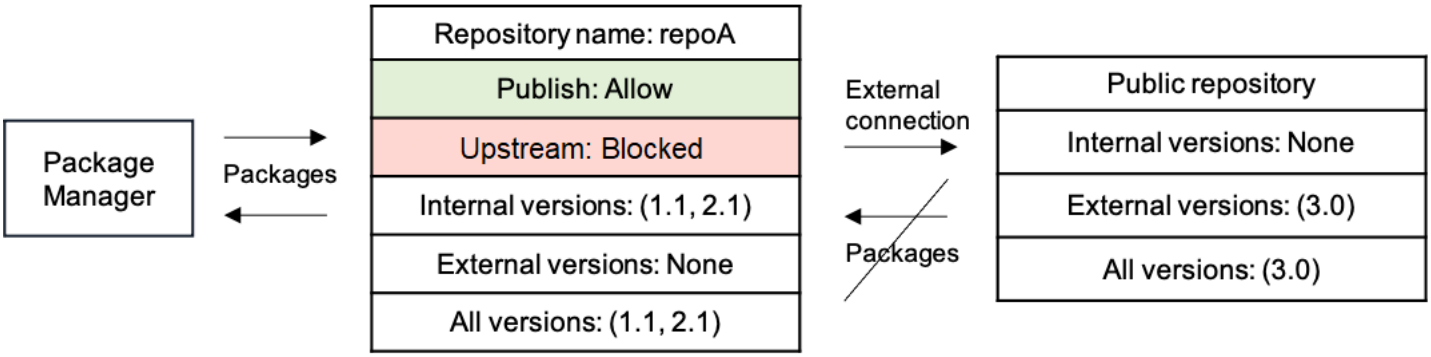
Negli scenari seguenti, un pacchetto interno è un pacchetto pubblicato direttamente da un gestore di pacchetti nel repository, ad esempio un pacchetto creato e gestito dall'utente o dal suo team. Un pacchetto esterno è un pacchetto esistente in un archivio pubblico che può essere inserito nel tuo repository con una connessione esterna.

Viene pubblicata una versione esterna del pacchetto per un pacchetto interno esistente

In questo scenario, si consideri un pacchetto interno, PackageA. Il tuo team pubblica la prima versione del pacchetto per PackageA in un repository. CodeArtifact Poiché questa è la prima versione del pacchetto, le impostazioni di controllo dell'origine del pacchetto vengono impostate

automaticamente su Pubblica: Consenti e Upstream: Block. Dopo che il pacchetto è presente nel repository, un pacchetto con lo stesso nome viene pubblicato in un archivio pubblico collegato al repository. CodeArtifact Potrebbe trattarsi di un tentativo di attacco di sostituzione delle dipendenze contro il pacchetto interno o potrebbe essere solo una coincidenza. Indipendentemente da ciò, i controlli di origine dei pacchetti sono configurati per bloccare l'ingestione della nuova versione esterna per proteggersi da un potenziale attacco.

Nell'immagine seguente, RePoA è il tuo CodeArtifact repository con una connessione esterna a un archivio pubblico. Il tuo repository contiene le versioni 1.1 e 2.1 di PackageA, ma la versione 3.0 è pubblicata nell'archivio pubblico. Normalmente, RePoA ingerisce la versione 3.0 dopo che il pacchetto è stato richiesto da un gestore di pacchetti. Poiché l'ingestione dei pacchetti è impostata su Block, la versione 3.0 non viene inserita nel CodeArtifact repository e non è disponibile per i gestori di pacchetti ad esso collegati.

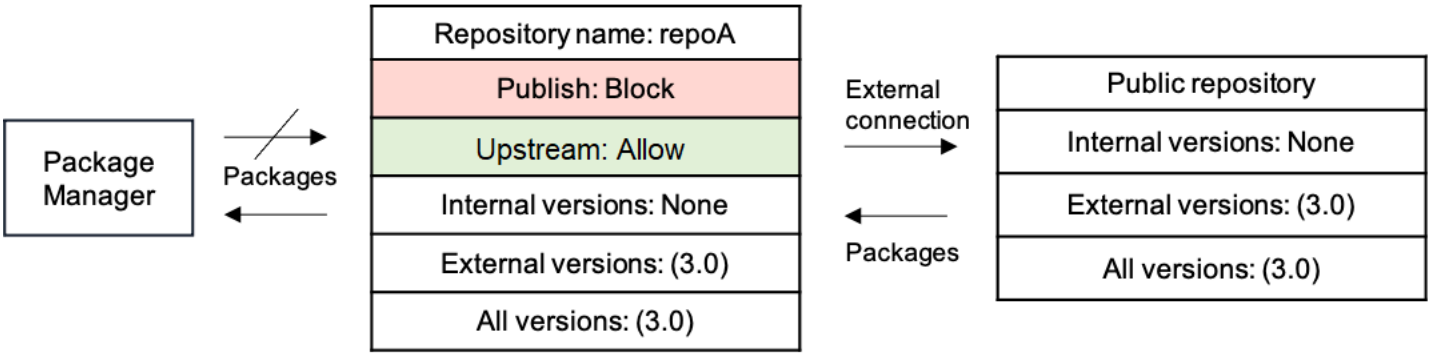


Viene pubblicata una versione interna del pacchetto per un pacchetto esterno esistente

In questo scenario, un pacchetto, packageB, esiste esternamente in un repository pubblico collegato al repository. Quando un gestore di pacchetti connesso al tuo repository richiede PackageB, la versione del pacchetto viene inserita nel tuo repository dal repository pubblico. Poiché questa è la prima versione del pacchetto di PackageB aggiunta al repository, le impostazioni di origine del pacchetto sono configurate su Publish: BLOCK e Upstream: ALLOW. Successivamente, si tenta di pubblicare una versione con lo stesso nome di pacchetto nel repository. O non siete a conoscenza del pacchetto pubblico e state cercando di pubblicare un pacchetto non correlato con lo stesso nome, oppure state cercando di pubblicare una versione con patch, oppure state cercando di pubblicare direttamente la versione esatta del pacchetto che già esiste esternamente. CodeArtifact rifiuterà la versione che state tentando di pubblicare, ma vi permetterà di ignorare esplicitamente il rifiuto e di pubblicare la versione se necessario.

Nell'immagine seguente, RePoA è il tuo CodeArtifact repository con una connessione esterna a un archivio pubblico. Il tuo repository contiene la versione 3.0 che ha importato dal repository pubblico.

Vuoi pubblicare la versione 1.1 nel tuo repository. Normalmente, è possibile pubblicare la versione 1.2 su RePoA, ma poiché la pubblicazione è impostata su Block, la versione 1.2 non può essere pubblicata.



Pubblicazione di una versione patchata di un pacchetto esterno esistente

In questo scenario, un pacchetto, packageB, esiste esternamente in un repository pubblico collegato al repository. Quando un gestore di pacchetti collegato al tuo repository richiede PackageB, la versione del pacchetto viene inserita nel tuo repository dal repository pubblico. Poiché questa è la prima versione del pacchetto di PackageB aggiunta al repository, le impostazioni di origine del pacchetto sono configurate su Publish: BLOCK e Upstream: ALLOW. Il tuo team decide che deve pubblicare le versioni del pacchetto con patch di questo pacchetto nel repository. Per poter pubblicare direttamente le versioni dei pacchetti, il team modifica le impostazioni di controllo dell'origine del pacchetto in Publish: ALLOW e Upstream: BLOCK. Le versioni di questo pacchetto possono ora essere pubblicate direttamente nel repository e importate da archivi pubblici. Dopo che il team ha pubblicato le versioni del pacchetto con patch, ripristina le impostazioni di origine del pacchetto su Publish: BLOCK e Upstream: ALLOW.

Impostazioni di controllo dell'origine del pacchetto

Con i controlli sull'origine dei pacchetti, è possibile configurare il modo in cui le versioni dei pacchetti possono essere aggiunte a un repository. Gli elenchi seguenti includono le impostazioni e i valori disponibili per il controllo dell'origine dei pacchetti.

> **Note**
>
> Le impostazioni e i valori disponibili sono diversi quando si configurano i controlli di origine sui gruppi di pacchetti. Per ulteriori informazioni, consulta Controlli dell'origine dei gruppi di pacchetti.

Pubblicare

Questa impostazione configura se le versioni dei pacchetti possono essere pubblicate direttamente nel repository utilizzando gestori di pacchetti o strumenti simili.

- ALLOW: le versioni dei Package possono essere pubblicate direttamente.
- BLOCK: Le versioni dei pacchetti non possono essere pubblicate direttamente.

A monte

Questa impostazione configura se le versioni dei pacchetti possono essere importate da archivi pubblici esterni o conservate dagli archivi upstream quando richiesto da un gestore di pacchetti.

- CONSENTI: qualsiasi versione del pacchetto può essere conservata da altri CodeArtifact repository configurati come archivi upstream o importata da una fonte pubblica con una connessione esterna.
- BLOCK: le versioni dei pacchetti non possono essere conservate da altri CodeArtifact repository configurati come repository upstream o importate da una fonte pubblica con una connessione esterna.

Impostazioni predefinite per il controllo dell'origine dei pacchetti

Le impostazioni di controllo dell'origine dei pacchetti predefinite sono configurate in base alle impostazioni di controllo dell'origine del gruppo di pacchetti associato al pacchetto. Per ulteriori informazioni sui gruppi di pacchetti e sui controlli di origine dei gruppi di pacchetti, vedere Lavorare con i gruppi di pacchetti in CodeArtifact eControlli dell'origine dei gruppi di pacchetti.

Se un pacchetto è associato a un gruppo di pacchetti con impostazioni di restrizione `ALLOW` per ogni tipo di restrizione, i controlli di origine dei pacchetti predefiniti per un pacchetto si baseranno su come la prima versione di quel pacchetto viene aggiunta all'archivio.

- Se la prima versione del pacchetto viene pubblicata direttamente da un gestore di pacchetti, le impostazioni saranno Publish: ALLOW e Upstream: BLOCK.
- Se la prima versione del pacchetto viene importata da una fonte pubblica, le impostazioni saranno Publish: BLOCK e Upstream: ALLOW.

> **Note**
>
> I pacchetti che esistevano nei CodeArtifact repository prima di maggio 2022 circa avranno i controlli di origine dei pacchetti predefiniti Publish: ALLOW e Upstream: ALLOW. I controlli di origine dei pacchetti devono essere impostati manualmente per tali pacchetti. Gli attuali valori predefiniti sono stati impostati su nuovi pacchetti da quel momento e hanno iniziato a essere applicati quando la funzionalità è stata lanciata il 14 luglio 2022. Per ulteriori informazioni sull'impostazione dei controlli di origine dei pacchetti, consultaModifica dei controlli di origine dei pacchetti.

Altrimenti, se un pacchetto è associato a un gruppo di pacchetti con almeno un'impostazione di restrizione pari a `BLOCK` o`ALLOW_SPECIFIC_REPOSITORIES`, le impostazioni di controllo dell'origine predefinite per quel pacchetto saranno impostate su Publish: ALLOW e Upstream: ALLOW.

In che modo i controlli di origine dei pacchetti interagiscono con i controlli di origine dei gruppi di pacchetti

Poiché i pacchetti hanno impostazioni di controllo dell'origine e i gruppi di pacchetti associati hanno impostazioni di controllo dell'origine, è importante capire come queste due diverse impostazioni interagiscono tra loro.

L'interazione tra le due impostazioni è che un'impostazione di `BLOCK` sempre vince su un'impostazione di`ALLOW`. La tabella seguente elenca alcuni esempi di configurazioni e le relative impostazioni di controllo dell'origine effettive.

Impostazione del controllo dell'origine del pacchetto	Impostazione del controllo dell'origine del gruppo di pacchetti	Impostazione efficace del controllo dell'origine
PUBBLICA: CONSENTI UPSTREAM: CONSENTI	PUBBLICA: CONSENTI UPSTREAM: CONSENTI	PUBBLICA: CONSENTI UPSTREAM: CONSENTI
PUBBLICA: BLOCCA UPSTREAM: CONSENTI	PUBBLICA: CONSENTI UPSTREAM: CONSENTI	PUBBLICA: BLOCCA UPSTREAM: CONSENTI
PUBBLICA: CONSENTI	PUBBLICA: CONSENTI	PUBBLICA: CONSENTI

Impostazione del controllo dell'origine del pacchetto	Impostazione del controllo dell'origine del gruppo di pacchetti	Impostazione efficace del controllo dell'origine
UPSTREAM: CONSENTI	UPSTREAM: BLOCCA	UPSTREAM: BLOCCA

Ciò significa che un pacchetto con le impostazioni di origine di Publish: ALLOW e Upstream: ALLOW viene effettivamente rimandato alle impostazioni di controllo dell'origine del gruppo di pacchetti associato.

Modifica dei controlli di origine dei pacchetti

I controlli di origine dei pacchetti vengono configurati automaticamente in base al modo in cui la prima versione del pacchetto di un pacchetto viene aggiunta al repository, per ulteriori informazioni, vedereImpostazioni predefinite per il controllo dell'origine dei pacchetti. Per aggiungere o modificare i controlli di origine del pacchetto per un pacchetto in un CodeArtifact repository, effettuate i passaggi indicati nella procedura seguente.

Per aggiungere o modificare i controlli di origine del pacchetto (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel pannello di navigazione, scegli Repository e scegli il repository che contiene il pacchetto che desideri modificare.
3. Nella tabella Pacchetti, cerca e seleziona il pacchetto che desideri modificare.
4. Nella pagina di riepilogo del pacchetto, in Origin controls, scegli Modifica.
5. In Modifica controlli di origine, scegli i controlli di origine del pacchetto che desideri impostare per questo pacchetto. Entrambe le impostazioni di controllo dell'origine del pacchetto, Publish e Upstream, devono essere impostate contemporaneamente.

   - Per consentire la pubblicazione diretta delle versioni dei pacchetti, in Pubblica, scegli Consenti. Per bloccare la pubblicazione delle versioni dei pacchetti, scegli Blocca.
   - Per consentire l'inserimento di pacchetti da repository esterni e l'estrazione di pacchetti da repository upstream, in Origini upstream, scegli Consenti. Per bloccare tutte le importazioni e l'estrazione di versioni di pacchetti da repository esterni e upstream, scegliete Blocca.

Per aggiungere o modificare i controlli di origine dei pacchetti ()AWS CLI

1. In caso contrario, configurali AWS CLI seguendo la procedura riportata di seguitoConfigurazione con AWS CodeArtifact.
2. Usa il `put-package-origin-configuration` comando per aggiungere o modificare i controlli di origine del pacchetto. Sostituisci i seguenti campi:
   - Sostituisci *`my_domain`* con il CodeArtifact dominio che contiene il pacchetto che desideri aggiornare.
   - Sostituisci *`my_repo`* con il CodeArtifact repository che contiene il pacchetto che desideri aggiornare.
   - Sostituisci *`npm`* con il formato del pacchetto che desideri aggiornare.
   - Sostituisci *`my_package`* con il nome del pacchetto che desideri aggiornare.
   - Sostituisci *`ALLOW`* e *`BLOCK`* con le impostazioni di controllo dell'origine del pacchetto desiderate.

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

Repository editoriali e upstream

CodeArtifact non consente la pubblicazione di versioni di pacchetti presenti in repository upstream raggiungibili o in archivi pubblici. Ad esempio, supponiamo di voler pubblicare un pacchetto Maven in un repository e `myrepo` di disporre di un repository upstream con una `myrepo` connessione esterna `com.mycompany.mypackage:1.0` a Maven Central. Considerate i seguenti scenari.

1. Le impostazioni di controllo dell'origine del pacchetto `com.mycompany.mypackage` sono Publish: ALLOW e Upstream: ALLOW. Se `com.mycompany.mypackage:1.0` è presente nel repository upstream o in Maven Central, CodeArtifact rifiuta qualsiasi tentativo di pubblicazione su di esso con un errore di conflitto 409. `myrepo` È comunque possibile pubblicare una versione diversa, ad esempio. `com.mycompany.mypackage:1.1`
2. Le impostazioni di controllo dell'origine del pacchetto `com.mycompany.mypackage` sono Publish: ALLOW e Upstream: BLOCK. Puoi pubblicare nel tuo repository qualsiasi versione

`com.mycompany.mypackage` di che non esista già perché le versioni del pacchetto non sono raggiungibili.

3. Le impostazioni di controllo dell'origine del pacchetto `com.mycompany.mypackage` sono Publish: BLOCK e Upstream: ALLOW. Non puoi pubblicare alcuna versione del pacchetto direttamente nel tuo repository.

Lavorare con i gruppi di pacchetti in CodeArtifact

I gruppi di pacchetti possono essere utilizzati per applicare la configurazione a più pacchetti che corrispondono a uno schema definito utilizzando il formato del pacchetto, lo spazio dei nomi del pacchetto e il nome del pacchetto. È possibile utilizzare i gruppi di pacchetti per configurare più comodamente i controlli di origine dei pacchetti per più pacchetti. I controlli di origine dei pacchetti vengono utilizzati per bloccare o consentire l'inserimento o la pubblicazione di nuove versioni dei pacchetti, proteggendo gli utenti da azioni dannose note come attacchi di sostituzione delle dipendenze.

Ogni dominio contiene CodeArtifact automaticamente un gruppo di pacchetti root. Per impostazione predefinita`/*`, questo gruppo di pacchetti radice contiene tutti i pacchetti e consente alle versioni dei pacchetti di accedere ai repository del dominio da tutti i tipi di origine. Il gruppo di pacchetti root può essere modificato, ma non può essere eliminato.

La funzionalità Package Group Configuration funziona in modo sostanzialmente coerente quando si crea un nuovo gruppo di pacchetti o si elimina un gruppo di pacchetti esistente. Ciò significa che al momento della creazione o dell'eliminazione di un gruppo di pacchetti, i controlli di origine verranno applicati ai pacchetti associati previsti, ma con un certo ritardo dovuto all'eventuale comportamento coerente. Il tempo necessario per raggiungere la coerenza finale dipende dal numero di gruppi di pacchetti nel dominio e dal numero di pacchetti nel dominio. Potrebbe esserci un breve periodo in cui i controlli di origine non si riflettono immediatamente sui pacchetti associati dopo la creazione o l'eliminazione di un gruppo di pacchetti.

Inoltre, gli aggiornamenti ai controlli di origine dei gruppi di pacchetti sono efficaci quasi immediatamente. A differenza della creazione o dell'eliminazione di gruppi di pacchetti, le modifiche ai controlli di origine di un gruppo di pacchetti esistente si riflettono sui pacchetti associati senza lo stesso ritardo.

Questi argomenti contengono informazioni sui gruppi di pacchetti in AWS CodeArtifact.

Argomenti

- Crea un gruppo di pacchetti
- Visualizza o modifica un gruppo di pacchetti
- Eliminare un gruppo di pacchetti
- Controlli dell'origine dei gruppi di pacchetti
- Sintassi della definizione del gruppo di pacchetti e comportamento di abbinamento

- Aggiungi un tag a un gruppo di pacchetti in CodeArtifact

Crea un gruppo di pacchetti

È possibile creare un gruppo di pacchetti utilizzando la CodeArtifact console, il AWS Command Line Interface (AWS CLI) o CloudFormation. Per ulteriori informazioni sulla gestione dei gruppi di CodeArtifact pacchetti con CloudFormation, vedereCreare CodeArtifact risorse con AWS CloudFormation.

Creare un gruppo di pacchetti (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il dominio in cui desideri creare un gruppo di pacchetti.
3. Scegli Package groups e scegli Create package group.
4. In Definizione del gruppo di pacchetti, inserisci la definizione del gruppo di pacchetti per il tuo gruppo di pacchetti. La definizione del gruppo di pacchetti determina quali pacchetti sono associati al gruppo. È possibile inserire manualmente la definizione del gruppo di pacchetti con testo oppure utilizzare la modalità visiva per effettuare selezioni e la definizione del gruppo di pacchetti verrà creata automaticamente.
5. Per utilizzare la modalità visiva per creare la definizione del gruppo di pacchetti:

    a. Scegliete Visual per passare alla modalità visiva.

    b. In Formato Package, scegli il formato dei pacchetti da associare a questo gruppo.

    c. In Namespace (Ambito), scegli i criteri dello spazio dei nomi in base ai quali effettuare la corrispondenza.

    - Uguale: corrisponde esattamente allo spazio dei nomi specificato. Se selezionato, inserisci lo spazio dei nomi su cui eseguire la corrispondenza.
    - Vuoto: abbina i pacchetti senza namespace.
    - Inizia con una parola: corrisponde ai namespace che iniziano con una parola specificata. Se selezionato, inserisci la parola prefisso in base alla quale abbinare. Per ulteriori informazioni sulle parole e sui limiti delle parole, vedereParole, confini delle parole e corrispondenza dei prefissi.

- Tutti: abbina i pacchetti in tutti i namespace.

d. Se è selezionato Uguale a, Vuoto o Inizia con parola, in Nome pacchetto scegli i criteri relativi al nome del pacchetto in base ai quali trovare la corrispondenza.

- Esattamente uguale: corrisponde esattamente al nome del pacchetto specificato. Se selezionato, inserisci il nome del pacchetto su cui abbinare.
- Inizia con il prefisso: abbina i pacchetti che iniziano con il prefisso specificato.
- Inizia con una parola: abbina i pacchetti che iniziano con una parola specificata. Se selezionato, inserisci la parola con il prefisso in base alla quale abbinare. Per ulteriori informazioni sulle parole e sui limiti delle parole, vedereParole, confini delle parole e corrispondenza dei prefissi.
- Tutti: abbina tutti i pacchetti.

e. Scegli Avanti per rivedere la definizione.

6. Per inserire la definizione del gruppo di pacchetti con testo:

a. Scegliete Testo per passare alla modalità testo.

b. In Definizione del gruppo di pacchetti, immettere la definizione del gruppo di pacchetti. Per ulteriori informazioni sulla sintassi della definizione del gruppo di pacchetti, vedereSintassi della definizione del gruppo di pacchetti e comportamento di abbinamento.

c. Scegliete Avanti per rivedere la definizione.

7. In Revisione della definizione, esamina i pacchetti che verranno inclusi nel nuovo gruppo di pacchetti in base alla definizione fornita in precedenza. Dopo la revisione, scegli Avanti.
8. In Informazioni sul gruppo di pacchetti, aggiungi facoltativamente una descrizione e un indirizzo email di contatto per il gruppo di pacchetti. Scegli Next (Successivo).
9. In Package origin controls, configura i controlli di origine da applicare ai pacchetti del gruppo. Per ulteriori informazioni sui controlli di origine dei gruppi di pacchetti, vedereControlli dell'origine dei gruppi di pacchetti.
10. Scegliete Crea gruppo di pacchetti.

Crea un gruppo di pacchetti (AWS CLI)

Usa il `create-package-group` comando per creare un gruppo di pacchetti nel tuo dominio. Per l'`--package-group`opzione, inserisci la definizione del gruppo di pacchetti che determina quali pacchetti sono associati al gruppo. Per ulteriori informazioni sulla sintassi della definizione dei

gruppi di pacchetti, vedereSintassi della definizione del gruppo di pacchetti e comportamento di abbinamento.

In caso contrario, configuralo AWS CLI seguendo la procedura riportata di seguito. Configurazione con 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
```

Visualizza o modifica un gruppo di pacchetti

È possibile visualizzare un elenco di tutti i gruppi di pacchetti, visualizzare i dettagli di un gruppo di pacchetti specifico o modificare i dettagli o la configurazione di un gruppo di pacchetti utilizzando la CodeArtifact console o AWS Command Line Interface (AWS CLI).

Visualizza o modifica un gruppo di pacchetti (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il dominio che contiene il gruppo di pacchetti che desideri visualizzare o modificare.
3. Scegli Gruppi di pacchetti e scegli il gruppo di pacchetti che desideri visualizzare o modificare.
4. In Dettagli, visualizza le informazioni sul gruppo di pacchetti, tra cui il gruppo principale, la descrizione, l'ARN, l'e-mail di contatto e i controlli di origine del pacchetto.
5. In Sottogruppi, visualizza un elenco di gruppi di pacchetti che hanno questo gruppo come gruppo principale. I gruppi di pacchetti in questo elenco possono ereditare le impostazioni da questo gruppo di pacchetti. Per ulteriori informazioni, consulta Gerarchia dei gruppi di pacchetti e specificità del modello.
6. In Pacchetti, visualizza i pacchetti che appartengono a questo gruppo di pacchetti in base alla definizione del gruppo di pacchetti. Nella colonna Forza, puoi vedere la forza dell'associazione dei pacchetti. Per ulteriori informazioni, consulta Gerarchia dei gruppi di pacchetti e specificità del modello.

7. Per modificare le informazioni sul gruppo di pacchetti, scegli Modifica gruppo di pacchetti.

   a. In Informazioni, aggiorna la descrizione o le informazioni di contatto del gruppo di pacchetti. Non è possibile modificare la definizione di un gruppo di pacchetti.

   b. In Package group origin controls, aggiorna le impostazioni di controllo dell'origine del gruppo di pacchetti, che determinano in che modo i pacchetti associati possono entrare nei repository nel dominio. Per ulteriori informazioni, consulta Controlli dell'origine dei gruppi di pacchetti.

Visualizza o modifica un gruppo di pacchetti ()AWS CLI

Utilizzate i seguenti comandi per visualizzare o modificare i gruppi di pacchetti con AWS CLI. In caso contrario, configurali AWS CLI seguendo la procedura riportata inConfigurazione con AWS CodeArtifact.

Per visualizzare tutti i gruppi di pacchetti in un dominio, usa il `list-package-groups` comando.

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

Per visualizzare i dettagli su un gruppo di pacchetti, usa il `describe-package-group` comando. Per ulteriori informazioni sulle definizioni dei gruppi di pacchetti, vedereSintassi ed esempi per la definizione dei gruppi di pacchetti.

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

Per visualizzare i gruppi di pacchetti figli di un gruppo di pacchetti, utilizzare il `list-sub-package-groups` comando.

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

Per visualizzare il gruppo di pacchetti associato a un pacchetto, utilizzare il `get-associated-package-group` comando. È necessario utilizzare il nome e lo spazio dei nomi normalizzati del pacchetto per i formati di pacchetto NuGet Python e Swift. Per ulteriori informazioni su come vengono normalizzati i nomi dei pacchetti e gli spazi dei nomi, consulta la documentazione sulla normalizzazione dei nomi di NuGetPython e Swift.

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

Per modificare un gruppo di pacchetti, usa il comando. `update-package-group` Questo comando viene utilizzato per aggiornare le informazioni di contatto o la descrizione di un gruppo di pacchetti. Per informazioni sulle impostazioni di controllo dell'origine dei gruppi di pacchetti e sulla loro aggiunta o modifica, vedereControlli dell'origine dei gruppi di pacchetti. Per ulteriori informazioni sulle definizioni dei gruppi di pacchetti, vedere Sintassi ed esempi per la definizione dei gruppi di pacchetti

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

Eliminare un gruppo di pacchetti

È possibile eliminare un gruppo di pacchetti utilizzando la CodeArtifact console o AWS Command Line Interface (AWS CLI).

Notate il seguente comportamento quando eliminate i gruppi di pacchetti:

- Il gruppo di pacchetti root,`/*`, non può essere eliminato.
- I pacchetti e le versioni dei pacchetti associati a quel gruppo di pacchetti non vengono eliminati.
- Quando un gruppo di pacchetti viene eliminato, i gruppi di pacchetti figli diretti diventeranno figli del gruppo di pacchetti principale diretto del gruppo di pacchetti. Pertanto, se uno qualsiasi dei gruppi di figli eredita delle impostazioni dal genitore, tali impostazioni potrebbero cambiare.

Eliminare un gruppo di pacchetti (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli il dominio che contiene il gruppo di pacchetti che desideri visualizzare o modificare.
3. Scegli Package groups.
4. Scegli il gruppo di pacchetti che desideri eliminare e scegli Elimina.
5. Inserisci delete nel campo e scegli Elimina.

Eliminare un gruppo di pacchetti (AWS CLI)

Per eliminare un gruppo di pacchetti, utilizzate il `delete-package-group` comando.

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

Controlli dell'origine dei gruppi di pacchetti

I controlli di origine dei pacchetti vengono utilizzati per configurare il modo in cui le versioni dei pacchetti possono entrare in un dominio. È possibile impostare i controlli di origine su un gruppo di pacchetti per configurare il modo in cui le versioni di ogni pacchetto associato al gruppo di pacchetti possono accedere a repository specifici nel dominio.

Le impostazioni di controllo dell'origine dei gruppi di pacchetti sono le seguenti:

- Impostazioni di restrizione: Queste impostazioni definiscono se i pacchetti possono accedere a un repository CodeArtifact da archivi di pubblicazione, interni o pubblici esterni.
- Elenchi di repository consentiti: Ogni impostazione di restrizione può essere impostata per consentire l'utilizzo di archivi specifici. Se un'impostazione di restrizione è impostata per consentire repository specifici, a tale restrizione verrà associato un elenco di repository consentiti corrispondente.

> **Note**
>
> Le impostazioni di controllo dell'origine per i gruppi di pacchetti sono leggermente diverse dalle impostazioni di controllo dell'origine per i singoli pacchetti. Per ulteriori informazioni sulle impostazioni di controllo dell'origine per i pacchetti, vedereImpostazioni di controllo dell'origine del pacchetto.

Impostazioni di restrizione

Le impostazioni di restrizione delle impostazioni di controllo dell'origine di un gruppo di pacchetti determinano in che modo i pacchetti associati a quel gruppo possono entrare nei repository del dominio.

PUBLISH

L'`PUBLISH`impostazione configura se le versioni dei pacchetti possono essere pubblicate direttamente in qualsiasi repository del dominio utilizzando gestori di pacchetti o strumenti simili.

- ALLOW: le versioni dei pacchetti possono essere pubblicate direttamente in tutti gli archivi.
- BLOCK: Le versioni dei pacchetti non possono essere pubblicate direttamente in nessun repository.
- ALLOW_SPECIFIC_REPOSITORIES: le versioni dei pacchetti possono essere pubblicate direttamente solo nei repository specificati nell'elenco dei repository consentiti per la pubblicazione.
- INHERIT: l'`PUBLISH`impostazione viene ereditata dal primo gruppo di pacchetti principale con un'impostazione che non lo è. `INHERIT`

EXTERNAL_UPSTREAM

L'`EXTERNAL_UPSTREAM`impostazione configura se le versioni dei pacchetti possono essere importate da archivi pubblici esterni quando richiesto da un gestore di pacchetti. Per un elenco dei repository esterni supportati, vedere. Archivi di connessioni esterne supportati

- CONSENTI: Qualsiasi versione del pacchetto può essere inserita in tutti gli archivi da una fonte pubblica con una connessione esterna.
- BLOCK: Le versioni dei pacchetti non possono essere inserite in alcun repository da una fonte pubblica con una connessione esterna.

- ALLOW_SPECIFIC_REPOSITORIES: le versioni dei pacchetti possono essere importate solo da una fonte pubblica nei repository specificati nell'elenco dei repository consentiti per gli upstream esterni.
- INHERIT: l'impostazione viene ereditata dal primo gruppo di pacchetti principale con `EXTERNAL_UPSTREAM` un'impostazione che non lo è. `INHERIT`

INTERNAL_UPSTREAM

L'`INTERNAL_UPSTREAM`impostazione configura se le versioni dei pacchetti possono essere conservate dagli archivi upstream interni nello stesso CodeArtifact dominio quando richiesto da un gestore di pacchetti.

- CONSENTI: Qualsiasi versione del pacchetto può essere conservata da altri CodeArtifact repository configurati come archivi upstream.
- BLOCK: Le versioni dei pacchetti non possono essere conservate da altri CodeArtifact repository configurati come repository upstream.
- ALLOW_SPECIFIC_REPOSITORIES: le versioni dei pacchetti possono essere conservate solo da CodeArtifact altri repository configurati come repository upstream nei repository specificati nell'elenco dei repository consentiti per gli upstream interni.
- INHERIT: l'impostazione viene ereditata dal primo gruppo di pacchetti principale con un'impostazione che non lo è`INTERNAL_UPSTREAM`. `INHERIT`

Elenchi di repository consentiti

Quando un'impostazione di restrizione è configurata come`ALLOW_SPECIFIC_REPOSITORIES`, il gruppo di pacchetti contiene un elenco di repository consentiti che contiene un elenco di repository consentiti per tale impostazione di restrizione. Pertanto, un gruppo di pacchetti contiene da 0 a 3 elenchi di repository consentiti, uno per ogni impostazione configurata come. `ALLOW_SPECIFIC_REPOSITORIES`

Quando si aggiunge un repository all'elenco di repository consentiti di un gruppo di pacchetti, è necessario specificare a quale elenco di repository consentiti aggiungerlo.

I possibili elenchi di repository consentiti sono i seguenti:

- `EXTERNAL_UPSTREAM`: consente o blocca l'inserimento delle versioni dei pacchetti da repository esterni nel repository aggiunto.

- `INTERNAL_UPSTREAM`: consente o blocca l'estrazione delle versioni dei pacchetti da un altro CodeArtifact repository nel repository aggiunto.
- `PUBLISH`: consente o blocca la pubblicazione diretta delle versioni dei pacchetti dai gestori di pacchetti nell'archivio aggiunto.

Modifica delle impostazioni di controllo dell'origine dei gruppi di pacchetti

Per aggiungere o modificare i controlli di origine per un gruppo di pacchetti, effettuate i passaggi indicati nella procedura seguente. Per informazioni sulle impostazioni del controllo dell'origine del gruppo di pacchetti, vedere Impostazioni di restrizione eElenchi di repository consentiti.

Per aggiungere o modificare i controlli di origine dei gruppi di pacchetti (CLI)

1. In caso contrario, configurali AWS CLI seguendo la procedura riportata di seguito. Configurazione con AWS CodeArtifact
2. Usa il `update-package-group-origin-configuration` comando per aggiungere o modificare i controlli di origine del pacchetto.

   - Per`--domain`, inserisci il CodeArtifact dominio che contiene il gruppo di pacchetti che desideri aggiornare.
   - Per`--domain-owner`, inserisci il numero di account del proprietario del dominio.
   - Per`--package-group`, inserisci il gruppo di pacchetti che desideri aggiornare.
   - Per`--restrictions`, inserisci coppie chiave-valore che rappresentano le restrizioni di controllo dell'origine.
   - Per`--add-allowed-repositories`, inserisci un oggetto JSON contenente il tipo di restrizione e il nome del repository da aggiungere all'elenco dei repository consentiti corrispondenti per la restrizione.
   - Ad esempio`--remove-allowed-repositories`, inserisci un oggetto JSON contenente il tipo di restrizione e il nome del repository da rimuovere dall'elenco dei repository consentiti corrispondente per la restrizione.

```
aws codeartifact update-package-group-origin-configuration \
   --domain my_domain \
   --domain-owner 111122223333 \
   --package-group '/nuget/*' \
   --restrictions INTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES \
```

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

L'esempio seguente aggiunge più restrizioni e più repository in un unico comando.

```
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
```

Esempi di configurazione del controllo dell'origine dei gruppi di pacchetti

Gli esempi seguenti mostrano le configurazioni del controllo dell'origine dei pacchetti per scenari comuni di gestione dei pacchetti.

Consentire la pubblicazione ma non l'importazione di pacchetti con nomi privati

Questo scenario è probabilmente uno scenario comune nella gestione dei pacchetti:

- Consenti la pubblicazione di pacchetti con nomi privati nei repository del tuo dominio dai gestori di pacchetti e impedisci che vengano importati nei repository del tuo dominio da repository pubblici esterni.
- Consenti l'importazione di tutti gli altri pacchetti nei repository del tuo dominio da archivi pubblici esterni e impedisci che vengano pubblicati nei repository del tuo dominio dai gestori di pacchetti.

A tal fine, è necessario configurare un gruppo di pacchetti con uno schema che includa i nomi privati e le impostazioni di origine di PUBLISH: ALLOW, EXTERNAL_UPSTREAM: BLOCK e INTERNAL_UPSTREAM: ALLOW. Ciò garantirà che i pacchetti con nomi privati possano essere pubblicati direttamente, ma non possano essere importati da repository esterni.

I seguenti AWS CLI comandi creano e configurano un gruppo di pacchetti con impostazioni di restrizione dell'origine che corrispondono al comportamento desiderato:

Per creare il gruppo di pacchetti:

```
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"
```

Per aggiornare la configurazione di origine del gruppo di pacchetti:

```
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
```

Consentire l'importazione da repository esterni tramite un repository

In questo scenario, il dominio dispone di più repository. Di questi repository, `repoA` dispone di una connessione upstream `arepoB`, che dispone di una connessione esterna all'archivio pubblico, `npmjs.com` come illustrato di seguito:

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

Si desidera consentire l'inserimento di pacchetti da un gruppo di pacchetti specifico, `/npm/space/anycompany~` da `npmjs.com` in`repoA`, ma solo tramite. `repoB` Volete anche bloccare l'inserimento dei pacchetti associati al gruppo di pacchetti in qualsiasi altro repository del vostro dominio e bloccare la pubblicazione diretta dei pacchetti con i gestori di pacchetti. A tal fine, create e configurate il gruppo di pacchetti come segue:

Impostazioni di restrizione dell'origine di PUBLISH: BLOCK e EXTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES e INTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES.

`repoA`e aggiunto all'elenco di repository consentiti appropriato: `repoB`

- `repoA`deve essere aggiunto alla `INTERNAL_UPSTREAM` lista, in quanto riceverà i pacchetti dal suo upstream interno,. `repoB`
- `repoB`dovrebbe essere aggiunto alla `EXTERNAL_UPSTREAM` lista, in quanto otterrà i pacchetti dal repository esterno,. `npmjs.com`

I seguenti AWS CLI comandi creano e configurano un gruppo di pacchetti con impostazioni di restrizione dell'origine che corrispondono al comportamento desiderato:

Per creare il gruppo di pacchetti:

```
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"
```

Per aggiornare la configurazione di origine del gruppo di pacchetti:

```
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
```

In che modo le impostazioni di controllo dell'origine del gruppo di pacchetti interagiscono con le impostazioni di controllo dell'origine dei pacchetti

Poiché i pacchetti hanno impostazioni di controllo dell'origine e i gruppi di pacchetti associati hanno impostazioni di controllo dell'origine, è importante capire come queste due diverse impostazioni interagiscono tra loro. Per informazioni sull'interazione tra le impostazioni, vedereIn che modo i controlli di origine dei pacchetti interagiscono con i controlli di origine dei gruppi di pacchetti.

Sintassi della definizione del gruppo di pacchetti e comportamento di abbinamento

Questo argomento contiene informazioni sulla definizione dei gruppi di pacchetti, sul comportamento di corrispondenza dei modelli, sulla forza dell'associazione dei pacchetti e sulla gerarchia dei gruppi di pacchetti.

Indice

- Sintassi ed esempi per la definizione dei gruppi di pacchetti
  - Definizione e normalizzazione del gruppo di pacchetti
  - Namespace nelle definizioni dei gruppi di pacchetti
- Gerarchia dei gruppi di pacchetti e specificità del modello
- Parole, confini delle parole e corrispondenza dei prefissi
- Distinzione tra lettere maiuscole e minuscole
- Partita forte e debole
- Varianti aggiuntive

Sintassi ed esempi per la definizione dei gruppi di pacchetti

La sintassi del modello per definire i gruppi di pacchetti segue da vicino la formattazione dei percorsi dei pacchetti. Il percorso del pacchetto viene creato a partire dai componenti delle coordinate di un pacchetto (formato, namespace e nome) aggiungendo una barra all'inizio e separando ciascun componente con una barra. Ad esempio, il percorso del pacchetto npm denominato anycompany-ui-componentsnello spazio dei nomi è/-ui-components. npm/space/anycompany

Un modello di gruppo di pacchetti segue la stessa struttura di un percorso di pacchetto, tranne i componenti che non sono specificati come parte della definizione del gruppo vengono omessi e il modello termina con un suffisso. Il suffisso incluso determina il comportamento di corrispondenza del pattern, nel modo seguente:

- Un $ suffisso corrisponderà alla coordinata dell'intero pacchetto.
- Un ~ suffisso corrisponderà a un prefisso.
- Un * suffisso corrisponderà a tutti i valori del componente definito in precedenza.

Ecco alcuni modelli di esempio per ciascuna delle combinazioni consentite:

1. Tutti i formati di pacchetti: `/*`
2. Un formato di pacchetto specifico: `/npm/*`
3. Formato del pacchetto e prefisso dello spazio dei nomi: `/maven/com.anycompany~`
4. Formato e namespace del pacchetto: `/npm/space/*`
5. Formato del pacchetto, namespace e prefisso del nome: `/npm/space/anycompany-ui~`
6. Formato, namespace e nome del pacchetto: `/maven/org.apache.logging.log4j/log4j-core$`

Come illustrato negli esempi precedenti, il ~ suffisso viene aggiunto alla fine di uno spazio dei nomi o di un nome per rappresentare una corrispondenza di prefisso e * viene dopo una barra se utilizzato per abbinare tutti i valori del componente successivo nel percorso (tutti i formati, tutti gli spazi dei nomi o tutti i nomi).

Definizione e normalizzazione del gruppo di pacchetti

CodeArtifact normalizza i NuGet nomi dei pacchetti Python e Swift e normalizza i namespace dei pacchetti Swift prima di archiviarli. CodeArtifact usa questi nomi normalizzati per abbinare i pacchetti alle definizioni dei gruppi di pacchetti. Pertanto, i gruppi di pacchetti che contengono uno spazio dei nomi o un nome in questi formati devono utilizzare lo spazio dei nomi e il nome normalizzati. Per ulteriori informazioni su come vengono normalizzati i nomi dei pacchetti e gli spazi dei nomi, consulta la documentazione sulla normalizzazione dei nomi di NuGetPython e Swift.

Namespace nelle definizioni dei gruppi di pacchetti

Per pacchetti o formati di pacchetti senza uno spazio dei nomi (Python e NuGet), i gruppi di pacchetti non devono contenere uno spazio dei nomi. La definizione del gruppo di pacchetti per questi gruppi di pacchetti contiene una sezione dello spazio dei nomi vuota. Ad esempio, il percorso per il pacchetto Python denominato requests è /python//requests.

Per i pacchetti o i formati di pacchetto con uno spazio dei nomi (Maven, generic e Swift), lo spazio dei nomi deve essere incluso se è incluso il nome del pacchetto. Per il formato del pacchetto Swift, verrà utilizzato lo spazio dei nomi dei pacchetti normalizzato. Per ulteriori informazioni su come vengono normalizzati gli spazi dei nomi dei pacchetti Swift, consulta. Normalizzazione del nome e dello spazio dei nomi del pacchetto Swift

Gerarchia dei gruppi di pacchetti e specificità del modello

I pacchetti che sono «in» o «associati a» un gruppo di pacchetti sono pacchetti con un percorso di pacchetto che corrisponde allo schema del gruppo ma non corrisponde allo schema di un gruppo più specifico. Ad esempio, dati i gruppi di pacchetti `/npm/*` and`/npm/space/*`, il percorso del pacchetto /npm//react è associato al primo gruppo (`/npm/*`) mentre/npm/space/aui.components e/npm/space/amplify-ui-core sono associati al secondo gruppo (). `/npm/space/*` Anche se un pacchetto può corrispondere a più gruppi, ogni pacchetto è associato solo a un singolo gruppo, la corrispondenza più specifica, e solo la configurazione di un gruppo si applica al pacchetto.

Quando il percorso di un pacchetto corrisponde a più modelli, il modello «più specifico» può essere considerato il modello di corrispondenza più lungo. In alternativa, il modello più specifico è quello che corrisponde a un sottoinsieme appropriato dei pacchetti che corrispondono al modello meno specifico. Nel nostro esempio precedente, ogni pacchetto che corrisponde corrisponde `/npm/space/*` anche`/npm/*`, ma non è vero il contrario, il che rende `/npm/space/*` il modello più specifico perché è un sottoinsieme appropriato di. `/npm/*` Poiché un gruppo è un sottoinsieme di un altro gruppo, crea una gerarchia, in cui si `/npm/space/*` trova un sottogruppo del gruppo principale,. `/npm/*`

Sebbene a un pacchetto si applichi solo la configurazione del gruppo di pacchetti più specifico, tale gruppo può essere configurato per ereditare dalla configurazione del gruppo principale.

Parole, confini delle parole e corrispondenza dei prefissi

Prima di parlare della corrispondenza dei prefissi, definiamo alcuni termini chiave:

- Una parola, una lettera o un numero, seguito da zero o più lettere, numeri o caratteri distintivi (come accenti, dieresi, ecc.).
- Un limite di parola si trova alla fine di una parola, quando viene raggiunto un carattere diverso da una parola. I caratteri non verbali sono caratteri di punteggiatura come, e. `.` `-` `_`

In particolare, lo schema regex di una parola è`[\p{L}\p{N}][\p{L}\p{N}\p{M}]*`, che può essere suddiviso come segue:

- `\p{L}`rappresenta qualsiasi lettera.
- `\p{N}`rappresenta qualsiasi numero.
- `\p{M}`rappresenta qualsiasi carattere distintivo, ad esempio accenti, dieresi, ecc.

Pertanto, `[\p{L}\p{N}]` rappresenta un numero o una lettera e `[\p{L}\p{N}\p{M}]*` rappresenta zero o più lettere, numeri o caratteri distintivi e un limite di parola si trova alla fine di ogni corrispondenza di questo modello regex.

> **Note**
>
> La corrispondenza dei confini delle parole si basa su questa definizione di «parola». Non si basa su parole definite in un dizionario, o CameCase. Ad esempio, non esiste alcun limite di parola in `oneword` o`OneWord`.

Ora che la parola e il limite della parola sono definiti, possiamo usarli per descrivere la corrispondenza dei prefissi in. CodeArtifact Per indicare una corrispondenza di prefisso sul confine di una parola, dopo un carattere di parola viene utilizzato un carattere di corrispondenza (~). Ad esempio, il modello `/npm/space/foo~` corrisponde ai percorsi dei pacchetti `/npm/space/foo` e`/npm/space/foo-bar`, ma non `/npm/space/food` a or. `/npm/space/foot`

È necessario utilizzare un carattere jolly (`*`) al posto di ~ quando si segue un carattere non verbale, come nel pattern. `/npm/*`

Distinzione tra lettere maiuscole e minuscole

Le definizioni dei gruppi di pacchetti fanno distinzione tra maiuscole e minuscole, il che significa che modelli che differiscono solo per maiuscole e minuscole possono esistere come gruppi di pacchetti separati. Ad esempio, un utente può creare gruppi di pacchetti separati con i modelli e `/npm//asyncstorage$` per i tre pacchetti separati esistenti nel registro pubblico di npm: `/npm//AsyncStorage$`/`npm//asyncStorage$`, asyncStorage AsyncStorage, asyncstorage che differiscono solo per maiuscole e minuscole.

Sebbene i casi siano importanti, associa CodeArtifact comunque i pacchetti a un gruppo di pacchetti se il pacchetto presenta una variazione del modello che differisce da caso a caso. Se un utente crea il gruppo di `/npm//AsyncStorage$` pacchetti senza creare gli altri due gruppi mostrati sopra, tutte le varianti maiuscole e minuscole del nome AsyncStorage, incluse asyncStorage e asyncstorage, verranno associate al gruppo di pacchetti. Tuttavia, come descritto nella sezione successiva, queste variazioni verranno gestite in modo diverso rispetto AsyncStorageaPartita forte e debole, che corrisponde esattamente allo schema.

Partita forte e debole

Le informazioni nella sezione precedenteDistinzione tra lettere maiuscole e minuscole, affermano che i gruppi di pacchetti fanno distinzione tra maiuscole e minuscole, e poi proseguono spiegando che non fanno distinzione tra maiuscole e minuscole. Questo perché le definizioni dei gruppi di pacchetti CodeArtifact hanno un concetto di corrispondenza forte (o corrispondenza esatta) e di corrispondenza debole (o corrispondenza di variazione). Una forte corrispondenza si ha quando il pacchetto corrisponde esattamente allo schema, senza alcuna variazione. Una corrispondenza debole si verifica quando il pacchetto corrisponde a una variante dello schema, ad esempio lettere maiuscole diverse. Un comportamento di corrispondenza debole impedisce ai pacchetti che sono variazioni dello schema di un gruppo di pacchetti di passare a un gruppo di pacchetti più generale. Quando un pacchetto è una variante (weak match) del pattern del gruppo corrispondente più specifico, il pacchetto viene associato al gruppo ma il pacchetto viene bloccato invece di applicare la configurazione di controllo dell'origine del gruppo, impedendo che qualsiasi nuova versione del pacchetto venga estratta dai flussi iniziali o pubblicata. Questo comportamento riduce il rischio di attacchi alla catena di approvvigionamento derivanti dalla confusione delle dipendenze di pacchetti con nomi quasi identici.

Per illustrare un comportamento debole, supponiamo che il gruppo di pacchetti `/npm/*` consenta l'ingestione e blocchi la pubblicazione. Un gruppo di pacchetti più specifico`/npm//anycompany-spicy-client$`, è configurato per bloccare l'ingestione e consentire la pubblicazione. Il pacchetto denominato anycompany-spicy-clientcorrisponde perfettamente al gruppo di pacchetti, che consente la pubblicazione delle versioni del pacchetto e blocca l'acquisizione delle versioni del pacchetto. L'unico maiuscolo del nome del pacchetto che può essere pubblicato è anycompany-spicy-client, poiché corrisponde perfettamente al modello di definizione del pacchetto. Una variante diversa tra maiuscole e minuscole, come AnyCompany-spicy-client, è bloccata dalla pubblicazione perché non corrisponde a nulla. Ancora più importante, il gruppo di pacchetti blocca l'inserimento di tutte le varianti tra maiuscole e minuscole, non solo del nome in minuscolo utilizzato nel modello, riducendo il rischio di un attacco di confusione delle dipendenze.

Varianti aggiuntive

Oltre alle differenze tra maiuscole e minuscole, la corrispondenza debole ignora anche le differenze nelle sequenze di trattini`-`, punti`.`, `_` trattini bassi e caratteri confondibili (ad esempio caratteri dall'aspetto simile di alfabeti separati). Durante la normalizzazione, utilizzata per la corrispondenza debole, CodeArtifact esegue la piegatura tra maiuscole e minuscole (simile alla conversione in minuscolo), sostituisce le sequenze di caratteri con trattini, punti e sottolineature con un singolo punto e normalizza i caratteri confondibili.

La corrispondenza debole considera trattini, punti e caratteri di sottolineatura come equivalenti ma non li ignora completamente. Ciò significa che foo-bar, foo.bar, foo.. bar e foo_bar sono tutti equivalenti a match deboli, ma foobar no. Sebbene diversi archivi pubblici implementino misure per prevenire questo tipo di variazioni, la protezione fornita dagli archivi pubblici non rende superflua questa funzionalità dei gruppi di pacchetti. Ad esempio, gli archivi pubblici come il registro npm Public Registry impediranno nuove varianti del pacchetto denominato my-package solo se my-package è già stato pubblicato su di esso. Se my-package è un pacchetto interno e */npm//my-package$* viene creato un gruppo di pacchetti che consente la pubblicazione e blocca l'ingestione, probabilmente non vorrai pubblicare my-package nel registro pubblico di npm per impedire che una variante come my.package sia consentita.

Sebbene alcuni formati di pacchetti come Maven trattino questi caratteri in modo diverso (Maven lo tratta `.` come un separatore della gerarchia dei namespace ma non `-` or`_`), qualcosa come com.act-on potrebbe comunque essere confuso con com.act.on.

> **Note**
>
> Nota che ogni volta che più varianti sono associate a un gruppo di pacchetti, un amministratore può creare un nuovo gruppo di pacchetti per una variante specifica per configurare un comportamento diverso per quella variante.

Aggiungi un tag a un gruppo di pacchetti in CodeArtifact

I tag sono coppie chiave-valore associate a risorse AWS. Puoi applicare tag ai tuoi gruppi di pacchetti in CodeArtifact. Per informazioni sull'etichettatura CodeArtifact delle risorse, sui casi d'uso, sui vincoli di chiave e valore dei tag e sui tipi di risorse supportati, consulta. Applicazione di tag alle risorse

È possibile utilizzare la CLI per specificare i tag quando si crea un gruppo di pacchetti o si aggiunge, rimuove o si aggiorna il valore dei tag di un gruppo di pacchetti esistente.

Gruppi di pacchetti di tag (CLI)

È possibile utilizzare la CLI per gestire i tag dei gruppi di pacchetti.

In caso contrario, configurali AWS CLI seguendo la procedura riportata di seguito. Configurazione con AWS CodeArtifact

> **Tip**
>
> Per aggiungere tag, devi fornire l'Amazon Resource Name (ARN) del gruppo di pacchetti. Per ottenere l'ARN del gruppo di pacchetti, esegui il `describe-package-group` comando:
>
> ```
> aws codeartifact describe-package-group \
>    --domain my_domain \
>    --package-group /npm/scope/anycompany~ \
>    --query packageGroup.arn
> ```

Argomenti

- Aggiungere tag a un gruppo di pacchetti (CLI)
- Visualizza i tag per un gruppo di pacchetti (CLI)
- Modifica i tag per un gruppo di pacchetti (CLI)
- Rimuovere i tag da un gruppo di pacchetti (CLI)

Aggiungere tag a un gruppo di pacchetti (CLI)

È possibile aggiungere tag ai gruppi di pacchetti al momento della creazione o a un gruppo di pacchetti esistente. Per informazioni sull'aggiunta di tag a un gruppo di pacchetti al momento della creazione, consultaCrea un gruppo di pacchetti.

Per aggiungere un tag a un gruppo di pacchetti esistente con AWS CLI, nel terminale o nella riga di comando, esegui il tag-resource comando, specificando l'Amazon Resource Name (ARN) del gruppo di pacchetti a cui desideri aggiungere i tag e la chiave e il valore del tag che desideri aggiungere. Per informazioni sul gruppo di pacchetti ARNs, consulta. Gruppo di pacchetti ARNs

È possibile aggiungere più di un tag a un gruppo di pacchetti. Ad esempio, per etichettare un gruppo di pacchetti, */npm/scope/anycompany~* con due tag, una chiave di tag denominata *key1* con il valore del tag di *value1* e una chiave di tag denominata *key2* con il valore del tag di*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
```

In caso di successo, questo comando non produce alcun risultato.

Visualizza i tag per un gruppo di pacchetti (CLI)

Segui questi passaggi per utilizzare AWS CLI per visualizzare i AWS tag per un gruppo di pacchetti. Se non sono stati aggiunti tag, l'elenco restituito è vuoto.

Nel terminale o nella riga di comando, esegui il list-tags-for-resource comando con l'Amazon Resource Name (ARN) del gruppo di pacchetti. Per informazioni sul gruppo di pacchetti ARNs, consultaGruppo di pacchetti ARNs.

Ad esempio, per visualizzare un elenco di chiavi e valori di tag per un gruppo di pacchetti, */npm/scope/anycompany~* denominato con un valore ARN di `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~
```

Se il comando viene eseguito correttamente, restituisce informazioni simili alle seguenti:

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

Modifica i tag per un gruppo di pacchetti (CLI)

Segui questi passaggi per utilizzare AWS CLI per modificare un tag per un gruppo di pacchetti. È possibile modificare il valore di una chiave esistente o aggiungere un'altra chiave. Puoi anche rimuovere i tag da un gruppo di pacchetti, come mostrato nella sezione successiva.

Nel terminale o nella riga di comando, esegui il tag-resource comando, specificando l'ARN del gruppo di pacchetti in cui desideri aggiornare un tag e specifica la chiave del tag e il valore del tag. Per informazioni sul gruppo di pacchetti ARNs, vedere. Gruppo di pacchetti 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
```

In caso di successo, questo comando non produce alcun risultato.

Rimuovere i tag da un gruppo di pacchetti (CLI)

Segui questi passaggi per AWS CLI rimuovere un tag da un gruppo di pacchetti.

> **Note**
>
> Se elimini un gruppo di pacchetti, tutte le associazioni di tag vengono rimosse dal gruppo di pacchetti eliminato. Non è necessario rimuovere i tag prima di eliminare un gruppo di pacchetti.

Nel terminale o nella riga di comando, esegui il untag-resource comando, specificando l'ARN del gruppo di pacchetti in cui desideri rimuovere i tag e la chiave del tag che desideri rimuovere. Per informazioni sul gruppo di pacchetti ARNs, vedere. Gruppo di pacchetti ARNs

Ad esempio, per rimuovere più tag su un gruppo di pacchetti*/npm/scope/anycompany~*, con le chiavi dei tag *key1* e*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
```

In caso di successo, questo comando non ha alcun risultato. Dopo aver rimosso i tag, è possibile visualizzare i tag rimanenti nel gruppo di pacchetti utilizzando il `list-tags-for-resource` comando.

Lavorare con i domini in CodeArtifact

CodeArtifact i domini semplificano la gestione di più repository all'interno di un'organizzazione. È possibile utilizzare un dominio per applicare autorizzazioni a molti repository di proprietà di account AWS diversi. Una risorsa viene archiviata una sola volta in un dominio, anche se è disponibile in più repository.

Sebbene sia possibile avere più domini, consigliamo un unico dominio di produzione che contenga tutti gli artefatti pubblicati in modo che i team di sviluppo possano trovare e condividere i pacchetti. Puoi utilizzare un secondo dominio di preproduzione per testare le modifiche alla configurazione del dominio di produzione.

Questi argomenti descrivono come utilizzare la CodeArtifact console e come CloudFormation creare o configurare i CodeArtifact domini. AWS CLI

Argomenti

- Panoramica del dominio
- Creare un dominio
- Eliminazione di un dominio
- Politiche di dominio
- Aggiungi un tag a un dominio in CodeArtifact

Panoramica del dominio

Quando lavori con CodeArtifact, i domini sono utili per quanto segue:

- Storage deduplicato: una risorsa deve essere archiviata solo una volta in un dominio, anche se è disponibile in 1 o 1.000 repository. Ciò significa che paghi per lo storage una sola volta.
- Copia rapida: quando trasferisci pacchetti da un repository upstream a un CodeArtifact repository downstream o utilizzi l'CopyPackageVersions API, devono essere aggiornati solo i record di metadati. Non viene copiata alcuna risorsa. In questo modo è possibile configurare rapidamente un nuovo repository per lo staging o il test. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.
- Condivisione semplificata tra repository e team: tutte le risorse e i metadati di un dominio sono crittografati con un'unica chiave AWS KMS key (chiave KMS). Non è necessario gestire una chiave per ogni repository o concedere a più account l'accesso a una singola chiave.

- Applica la policy su più repository: l'amministratore di dominio può applicare la policy a tutto il dominio. Ciò include la limitazione degli account che hanno accesso agli archivi del dominio e di chi può configurare le connessioni agli archivi pubblici da utilizzare come fonti di pacchetti. Per ulteriori informazioni, consulta Politiche di dominio.
- Nomi di repository univoci: il dominio fornisce uno spazio dei nomi per i repository. I nomi dei repository devono essere univoci solo all'interno del dominio. È necessario utilizzare nomi significativi e facili da capire.

I nomi di dominio devono essere univoci all'interno di un account.

Non è possibile creare un repository senza un dominio. Quando si utilizza l'CreateRepositoryAPI per creare un repository, è necessario specificare un nome di dominio. Non è possibile spostare un repository da un dominio all'altro.

Un repository può appartenere allo stesso AWS account proprietario del dominio o a un account diverso. Se gli account proprietari sono diversi, all'account proprietario del repository deve essere concessa l'autorizzazione sulla risorsa del `CreateRepository` dominio. È possibile farlo aggiungendo una politica delle risorse al dominio utilizzando il comando. PutDomainPermissionsPolicy

Sebbene un'organizzazione possa avere più domini, si consiglia di disporre di un unico dominio di produzione che contenga tutti gli elementi pubblicati in modo che i team di sviluppo possano trovare e condividere i pacchetti all'interno dell'organizzazione. Un secondo dominio di preproduzione può essere utile per testare le modifiche alla configurazione del dominio di produzione.

Domini con più account

I nomi di dominio devono essere unici solo all'interno di un account, il che significa che potrebbero esserci più domini all'interno di una regione con lo stesso nome. Per questo motivo, se desideri accedere a un dominio di proprietà di un account al quale non sei autenticato, devi fornire l'ID del proprietario del dominio insieme al nome di dominio sia nella CLI che nella console. Vedi i seguenti esempi di CLI.

Accedi a un dominio di proprietà di un account su cui sei autenticato:

Quando accedi a un dominio all'interno dell'account con cui sei autenticato, devi solo specificare il nome di dominio. L'esempio seguente elenca i pacchetti presenti nel *`my_repo`* repository del *`my_domain`* dominio di proprietà del tuo account.

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

Accedi a un dominio di proprietà di un account al quale non sei autenticato:

Quando accedi a un dominio di proprietà di un account al quale non sei autenticato, devi specificare il proprietario del dominio e il nome del dominio. L'esempio seguente elenca i pacchetti presenti nel *other-repo* repository del *other-domain* dominio di proprietà di un account al quale non sei autenticato. Notate l'aggiunta del parametro. `--domain-owner`

```
aws codeartifact list-packages --domain other-domain --domain-owner 111122223333 --
repository other-repo
```

Tipi di AWS KMS chiavi supportati in CodeArtifact

CodeArtifact supporta solo chiavi KMS simmetriche. Non puoi utilizzare una chiave KMS asimmetrica per crittografare i tuoi domini. CodeArtifact Per ulteriori informazioni, consulta Identificazione delle chiavi KMS simmetriche e asimmetriche. Per informazioni su come creare una nuova chiave gestita dal cliente, consulta Creazione di chiavi KMS con crittografia simmetrica nella Guida per gli sviluppatori.AWS Key Management Service

CodeArtifact supporta AWS KMS External Key Stores (XKS). Sei responsabile della disponibilità, della durabilità e della latenza delle operazioni chiave con le chiavi XKS, che possono influire sulla disponibilità, la durata e la latenza con. CodeArtifact Alcuni esempi di effetti dell'uso delle chiavi XKS con: CodeArtifact

- Poiché ogni risorsa di un pacchetto richiesto e tutte le sue dipendenze sono soggette alla latenza di decrittografia, la latenza di compilazione può essere aumentata in modo sostanziale con un aumento della latenza delle operazioni XKS.
- Poiché tutte le risorse sono crittografate CodeArtifact, una perdita dei materiali chiave XKS comporterà la perdita di tutte le risorse associate al dominio utilizzando la chiave XKS.

Per ulteriori informazioni sulle chiavi XKS, consulta Archivi di chiavi esterni nella Guida per gli AWS Key Management Service sviluppatori.

Creare un dominio

È possibile creare un dominio utilizzando la CodeArtifact console, il AWS Command Line Interface (AWS CLI) o CloudFormation. Quando crei un dominio, questo non contiene alcun repository. Per

ulteriori informazioni, consulta Creazione di un repository . Per ulteriori informazioni sulla gestione dei CodeArtifact domini con CloudFormation, consulta. Creare CodeArtifact risorse con AWS CloudFormation

Argomenti

- Creare un dominio (console)
- Crea un dominio (AWS CLI)
- Esempio di politica AWS KMS chiave

Creare un dominio (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel riquadro di navigazione, scegli Domini, quindi scegli Crea dominio.
3. In Nome, inserisci un nome per il tuo dominio.
4. Espandere Additional configuration (Configurazione aggiuntiva).
5. Usa una AWS KMS key (chiave KMS) per crittografare tutte le risorse del tuo dominio. Puoi utilizzare una chiave KMS AWS gestita o una chiave KMS gestita da te. Per ulteriori informazioni sui tipi di chiavi KMS supportati in CodeArtifact, consulta. Tipi di AWS KMS chiavi supportati in CodeArtifact

   - Scegli la chiave gestita AWS se desideri utilizzare la chiave predefinita Chiave gestita da AWS.
   - Scegli la chiave gestita dal cliente se desideri utilizzare una chiave KMS da te gestita. Per utilizzare una chiave KMS che gestisci, in ARN della chiave gestita dal cliente, cerca e scegli la chiave KMS.

   Per ulteriori informazioni, consulta la sezione Chiave gestita da AWSCustomer managed key nella Developer Guide.AWS Key Management Service
6. Scegli Crea dominio.

Crea un dominio (AWS CLI)

Per creare un dominio con AWS CLI, usa il `create-domain` comando. È necessario utilizzare una AWS KMS key (chiave KMS) per crittografare tutte le risorse del dominio. Puoi utilizzare una chiave KMS AWS gestita o una chiave KMS gestita da te. Se utilizzi una chiave KMS AWS gestita, non utilizzare il parametro. `--encryption-key`

Per ulteriori informazioni sui tipi di chiavi KMS supportati in CodeArtifact, consulta. Tipi di AWS KMS chiavi supportati in CodeArtifact Per ulteriori informazioni sulle chiavi KMS, consulta Chiave gestita da AWSla sezione relativa alla chiave gestita dal cliente nella Guida per gli AWS Key Management Service sviluppatori.

```
aws codeartifact create-domain --domain my_domain
```

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul nuovo dominio.

```
{
    "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"
    }
}
```

Se utilizzi una chiave KMS che gestisci, includi il relativo Amazon Resource Name (ARN) nel parametro. `--encryption-key`

```
aws codeartifact create-domain --domain my_domain --encryption-key arn:aws:kms:us-
west-2:111122223333:key/your-kms-key
```

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul nuovo dominio.

```
{
    "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"
    }
}
```

Creare un dominio con tag

Per creare un dominio con tag, aggiungi il `--tags` parametro al tuo comando. `create-domain`

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

Esempio di politica AWS KMS chiave

Quando crei un dominio in CodeArtifact, utilizzi una chiave KMS per crittografare tutte le risorse del dominio. Puoi scegliere una chiave KMS AWS gestita o una chiave gestita dal cliente che gestisci tu stesso. Per ulteriori informazioni sulle chiavi KMS, consulta la Guida per gli AWS Key Management Service sviluppatori.

Per utilizzare una chiave gestita dal cliente, la chiave KMS deve avere una politica chiave che consenta l'accesso a. CodeArtifact Una politica chiave è una politica delle risorse per una AWS KMS chiave e rappresenta il modo principale per controllare l'accesso alle chiavi KMS. Ogni chiave KMS deve avere esattamente una policy chiave. Le istruzioni nella policy delle chiavi determinano chi dispone dell'autorizzazione per utilizzare la chiave KMS e come questa può essere utilizzata.

L'esempio seguente di dichiarazione politica chiave consente di AWS CodeArtifact creare sovvenzioni e visualizzare i dettagli chiave per conto degli utenti autorizzati. Questa dichiarazione politica limita l'autorizzazione ad CodeArtifact agire per conto dell'ID dell'account specificato utilizzando le chiavi di `kms:CallerAccount` condizione `kms:ViaService` e. Inoltre concede tutte le AWS KMS autorizzazioni all'utente root IAM, in modo che la chiave possa essere gestita dopo la creazione.

JSON

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

```
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Allow access through AWS CodeArtifact for all principals in
 the account that are authorized to use CodeArtifact",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:CallerAccount": "111122223333",
                    "kms:ViaService": "codeartifact.us-west-2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

Eliminazione di un dominio

È possibile eliminare un dominio utilizzando la CodeArtifact console o AWS Command Line Interface (AWS CLI).

Argomenti

- Restrizioni all'eliminazione del dominio
- Eliminare un dominio (console)

- Elimina un dominio ()AWS CLI

Restrizioni all'eliminazione del dominio

Normalmente, non è possibile eliminare un dominio che contiene repository. Prima di eliminare il dominio, è necessario eliminare i relativi repository. Per ulteriori informazioni, consulta Eliminare un repository.

Tuttavia, se CodeArtifact non hai più accesso alla chiave KMS del dominio, puoi eliminare il dominio anche se contiene ancora repository. Questa situazione si verificherà se elimini la chiave KMS del dominio o revochi la concessione KMS utilizzata per accedere alla chiave. CodeArtifact In questo stato, non è possibile accedere ai repository del dominio o ai pacchetti in essi archiviati. Inoltre, l'elenco e l'eliminazione dei repository non sono possibili quando CodeArtifact non è possibile accedere alla chiave KMS del dominio. Per questo motivo, l'eliminazione del dominio non verifica se il dominio contiene repository quando la chiave KMS del dominio è inaccessibile.

> **Note**
>
> Quando un dominio che contiene ancora repository viene eliminato, CodeArtifact eliminerà i repository in modo asincrono entro 15 minuti. Dopo l'eliminazione del dominio, i repository saranno ancora visibili nella CodeArtifact console e nell'output del `list-repositories` comando fino alla pulizia automatica del repository.

Eliminare un dominio (console)

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel pannello di navigazione, scegli Domini, quindi scegli il dominio che desideri eliminare.
3. Scegli Elimina.

Elimina un dominio ()AWS CLI

Usa il delete-domain comando per eliminare un dominio.

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

I dati in formato JSON vengono visualizzati nell'output con i dettagli sul dominio eliminato.

```
{
    "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"
    }
}
```

Politiche di dominio

CodeArtifact supporta l'utilizzo di autorizzazioni basate sulle risorse per controllare l'accesso. Le autorizzazioni basate sulle risorse consentono di specificare chi ha accesso a una risorsa e quali azioni può eseguire su di essa. Per impostazione predefinita, solo l'account AWS proprietario del dominio può creare e accedere ai repository nel dominio. Puoi applicare un documento di policy a un dominio per consentire ad altri responsabili IAM di accedervi.

Per ulteriori informazioni, consulta Politiche e autorizzazioni e Politiche basate sull'identità e Politiche basate sulle risorse.

Argomenti

- Abilita l'accesso tra più account a un dominio
- Esempio di policy di dominio
- Esempio di politica di dominio con AWS Organizations
- Imposta una politica di dominio
- Leggi una politica di dominio
- Eliminare una politica di dominio

Abilita l'accesso tra più account a un dominio

Una politica delle risorse è un file di testo in formato JSON. Il file deve specificare un principale (attore), una o più azioni e un effetto (`AllowoDeny`). Per creare un repository

in un dominio di proprietà di un altro account, al principale deve essere concessa l'`CreateRepository`autorizzazione sulla risorsa del dominio.

Ad esempio, la seguente politica in materia di risorse concede all'account l'`123456789012`autorizzazione a creare un repository nel dominio.

JSON

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

Per consentire la creazione di repository con tag, è necessario includere l'autorizzazione. `codeartifact:TagResource` Ciò consentirà inoltre all'account di aggiungere tag al dominio e a tutti i repository in esso contenuti.

La politica del dominio viene valutata per tutte le operazioni relative al dominio e a tutte le risorse all'interno del dominio. Ciò significa che la politica del dominio può essere utilizzata per applicare le autorizzazioni ai repository e ai pacchetti del dominio. Quando l'`Resource`elemento è impostato su`*`, l'istruzione si applica a tutte le risorse del dominio. Ad esempio, se la policy di cui sopra fosse inclusa anche `codeartifact:DescribeRepository` nell'elenco delle azioni IAM consentite, la policy consentirebbe di richiamare `DescribeRepository` ogni repository del dominio. Una policy di dominio può essere utilizzata per applicare autorizzazioni a risorse specifiche del dominio utilizzando una risorsa specifica ARNs nell'`Resource`elemento.

> **Note**
>
> È possibile utilizzare sia le politiche di dominio che quelle di repository per configurare le autorizzazioni. Quando sono presenti entrambe le politiche, entrambe verranno valutate e, se consentita da una delle due politiche, sarà consentita un'azione. Per ulteriori informazioni, consulta Interazione tra le politiche del repository e del dominio.

Per accedere ai pacchetti in un dominio di proprietà di un altro account, a un principale deve essere concessa l'`GetAuthorizationToken`autorizzazione sulla risorsa del dominio. Ciò consente al proprietario del dominio di esercitare il controllo sugli account che possono leggere il contenuto degli archivi del dominio.

Ad esempio, la seguente politica in materia di risorse concede all'account l'`123456789012`autorizzazione a recuperare un token di autenticazione per qualsiasi repository del dominio.

JSON

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

> **Note**
>
> A un principale che desidera recuperare pacchetti da un endpoint del repository deve essere concessa l'`ReadFromRepository`autorizzazione sulla risorsa del repository oltre

all'autorizzazione sul dominio. `GetAuthorizationToken` Allo stesso modo, a un principale che desidera pubblicare pacchetti su un endpoint del repository deve essere concessa l'autorizzazione in aggiunta a. `PublishPackageVersion` `GetAuthorizationToken` Per ulteriori informazioni sulle `PublishPackageVersion` autorizzazioni `ReadFromRepository` e, vedere Repository Policies.

Esempio di policy di dominio

Quando più account utilizzano un dominio, è necessario concedere agli account un set di autorizzazioni di base per consentire il pieno utilizzo del dominio. La seguente politica delle risorse elenca una serie di autorizzazioni che consentono il pieno utilizzo del dominio.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "BasicDomainPolicy",
            "Action": [
                "codeartifact:GetDomainPermissionsPolicy",
                "codeartifact:ListRepositoriesInDomain",
                "codeartifact:GetAuthorizationToken",
                "codeartifact:DescribeDomain",
                "codeartifact:CreateRepository"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            }
        }
    ]
}
```

> **Note**
>
> Non è necessario creare una politica di dominio se un dominio e tutti i relativi repository sono di proprietà di un singolo account e devono essere utilizzati solo da quell'account.

Esempio di politica di dominio con AWS Organizations

È possibile utilizzare la chiave di `aws:PrincipalOrgID` condizione per concedere l'accesso a un CodeArtifact dominio da tutti gli account dell'organizzazione, come segue.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "DomainPolicyForOrganization",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
             "codeartifact:GetDomainPermissionsPolicy",
             "codeartifact:ListRepositoriesInDomain",
             "codeartifact:GetAuthorizationToken",
             "codeartifact:DescribeDomain",
             "codeartifact:CreateRepository"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "aws:PrincipalOrgID":["o-xxxxxxxxxx"]}
        }
    }
}
```

Per ulteriori informazioni sull'utilizzo della chiave `aws:PrincipalOrgID` condition, consulta AWS Global Condition Context Keys nella IAM User Guide.

Imposta una politica di dominio

È possibile utilizzare il `put-domain-permissions-policy` comando per allegare una politica a un dominio.

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

Quando si chiama`put-domains-permissions-policy`, la politica delle risorse sul dominio viene ignorata durante la valutazione delle autorizzazioni. Ciò garantisce che il proprietario di un dominio non possa escludersi dal dominio, il che gli impedirebbe di aggiornare la politica delle risorse.

> **Note**
>
> Non è possibile concedere le autorizzazioni a un altro AWS account per aggiornare la politica delle risorse su un dominio utilizzando una politica delle risorse, poiché la politica delle risorse viene ignorata durante la chiamata. put-domain-permissions-policy

Output di esempio:

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

L'output del comando contiene l'Amazon Resource Name (ARN) della risorsa di dominio, il contenuto completo del documento di policy e un identificatore di revisione. L'identificatore di revisione può essere passato a utilizzando l'opzione. `put-domain-permissions-policy --policy-revision` Ciò garantisce che una revisione nota del documento venga sovrascritta e non una versione più recente impostata da un altro autore.

Leggi una politica di dominio

Per leggere una versione esistente di un documento di policy, usa il `get-domain-permissions-policy` comando. Per formattare l'output in modo da renderlo leggibile, usa `--output` e `--query policy.document` insieme al modulo `json.tool` Python, come segue.

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

Output di esempio:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "BasicDomainPolicy",
            "Action": [
                "codeartifact:GetDomainPermissionsPolicy",
                "codeartifact:ListRepositoriesInDomain",
                "codeartifact:GetAuthorizationToken",
                "codeartifact:CreateRepository"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            }
        }
    ]
}
```

Eliminare una politica di dominio

Usa il `delete-domain-permissions-policy` comando per eliminare una politica da un dominio.

```
aws codeartifact delete-domain-permissions-policy --domain my_domain --domain-
owner 111122223333
```

Il formato dell'output è lo stesso dei `delete-domain-permissions-policy` comandi `get-domain-permissions-policy` and.

Aggiungi un tag a un dominio in CodeArtifact

I tag sono coppie chiave-valore associate a risorse AWS. Puoi applicare tag ai tuoi domini in CodeArtifact. Per informazioni sull'etichettatura CodeArtifact delle risorse, sui casi d'uso, sui vincoli di chiave e valore dei tag e sui tipi di risorse supportati, consulta. Applicazione di tag alle risorse

È possibile utilizzare la CLI per specificare i tag quando si crea un dominio. Puoi utilizzare la console o la CLI per aggiungere o rimuovere tag e aggiornare i valori dei tag in un dominio. Puoi aggiungere fino a 50 tag a ciascun dominio.

Argomenti

- Domini di tag (CLI)
- Domini di tag (console)

Domini di tag (CLI)

Puoi utilizzare la CLI per gestire i tag di dominio.

Argomenti

- Aggiungere tag a un dominio (CLI)
- Visualizzare i tag per un dominio (CLI)
- Modifica dei tag per un dominio (CLI)
- Rimuovere tag da un dominio (CLI)

Aggiungere tag a un dominio (CLI)

Puoi usare la console o il AWS CLI per taggare i domini.

Per aggiungere un tag a un dominio quando lo crei, consultaCreazione di un repository .

In queste fasi, si assume che sia già installata una versione recente della AWS CLI o che sia aggiornata alla versione corrente. Per ulteriori informazioni, consultare Installing the AWS Command Line Interface.

Nel terminale o nella riga di comando, esegui il tag-resource comando, specificando l'Amazon Resource Name (ARN) del dominio a cui desideri aggiungere i tag e la chiave e il valore del tag che desideri aggiungere.

> **Note**
>
> Per ottenere l'ARN del dominio, esegui il `describe-domain` comando:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Puoi aggiungere più di un tag a un dominio. Ad esempio, per etichettare un dominio denominato *my_domain* con due tag, una chiave di tag denominata *key1* con il valore del *value1* tag e una chiave di tag denominata *key2* con il valore del tag di*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
```

In caso di successo, questo comando non produce alcun risultato.

Visualizzare i tag per un dominio (CLI)

Segui questi passaggi per utilizzare AWS CLI per visualizzare i AWS tag per un dominio. Se non sono stati aggiunti tag, l'elenco restituito è vuoto.

Nel terminale o nella riga di comando, esegui il list-tags-for-resource comando con l'Amazon Resource Name (ARN) del dominio.

> **Note**
>
> Per ottenere l'ARN del dominio, esegui il `describe-domain` comando:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Ad esempio, per visualizzare un elenco di chiavi e valori di tag per un dominio denominato *my_domain* con il valore `arn:aws:codeartifact:`*us-west-2*`:`*123456789012*`:domain/`*my_domain* ARN:

```
aws codeartifact list-tags-for-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain
```

Se il comando viene eseguito correttamente, restituisce informazioni simili alle seguenti:

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

Modifica dei tag per un dominio (CLI)

Segui questi passaggi per utilizzare AWS CLI per modificare un tag per un dominio. È possibile modificare il valore di una chiave esistente o aggiungere un'altra chiave. Puoi anche rimuovere i tag da un dominio, come illustrato nella sezione successiva.

Nel terminale o nella riga di comando, esegui il tag-resource comando, specificando l'ARN del dominio in cui desideri aggiornare un tag e specifica la chiave del tag e il valore del tag:

> **Note**
>
> Per ottenere l'ARN del dominio, esegui il `describe-domain` comando:
>
> ```
> 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
```

In caso di successo, questo comando non produce alcun risultato.

Rimuovere tag da un dominio (CLI)

Segui questi passaggi per utilizzare AWS CLI per rimuovere un tag da un dominio.

> **Note**
>
> Se elimini un dominio, tutte le associazioni di tag vengono rimosse dal dominio eliminato. Non è necessario rimuovere i tag prima di eliminare un dominio.

Nel terminale o nella riga di comando, esegui il untag-resource comando, specificando l'ARN del dominio in cui desideri rimuovere i tag e la chiave del tag che desideri rimuovere.

> **Note**
>
> Per ottenere l'ARN del dominio, esegui il `describe-domain` comando:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Ad esempio, per rimuovere più tag su un dominio denominato *mydomain* con le chiavi dei tag *key1* e*key2*:

```
aws codeartifact untag-resource --resource-arn arn:aws:codeartifact:us-
west-2:123456789012:domain/my_domain --tag-keys key1 key2
```

In caso di successo, questo comando non produce alcun risultato. Dopo aver rimosso i tag, è possibile visualizzare i tag rimanenti sul dominio utilizzando il `list-tags-for-resource` comando.

Domini di tag (console)

È possibile utilizzare la console o l'interfaccia a riga di comando per aggiungere tag alle risorse.

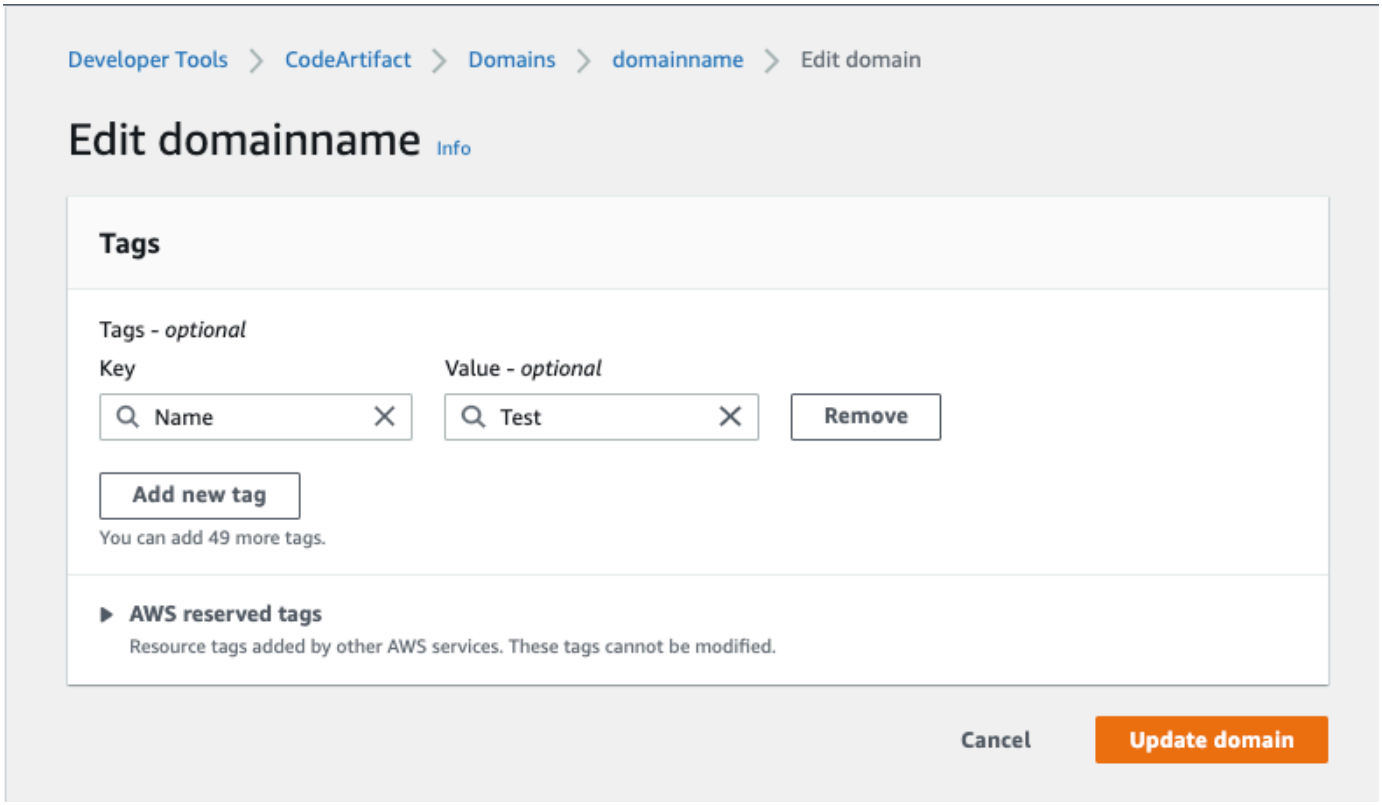
Argomenti

- Aggiungere tag a un dominio (console)
- Visualizza i tag per un dominio (console)
- Modifica i tag per un dominio (console)
- Rimuovi i tag da un dominio (console)

Aggiungere tag a un dominio (console)

Puoi usare la console per aggiungere tag a un dominio esistente.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Domini, scegli il dominio a cui desideri aggiungere i tag.
3. Espandi la sezione Dettagli.
4. In Tag di dominio, scegli Aggiungi tag di dominio se non ci sono tag nel dominio o scegli Visualizza e modifica i tag di dominio se ce ne sono.
5. Scegli Aggiungi nuovo tag.
6. Nei campi Chiave e Valore, inserisci il testo per ogni tag che desideri aggiungere. Il campo Value (Valore) è facoltativo. Ad esempio, in Key (Chiave), immettere **`Name`**. In Valore, immetti **`Test`**.



7. (Facoltativo) Scegliere Add tag (Aggiungi tag) per aggiungere ulteriori righe e inserire più tag.
8. Scegli Aggiorna dominio.

Visualizza i tag per un dominio (console)

Puoi usare la console per elencare i tag dei domini esistenti.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Domini, scegli il dominio in cui desideri visualizzare i tag.
3. Espandi la sezione Dettagli.
4. In Tag di dominio, scegli Visualizza e modifica i tag di dominio.

> **Note**
>
> Se non ci sono tag aggiunti a questo dominio, la console leggerà Aggiungi tag di dominio.

Modifica i tag per un dominio (console)

Puoi usare la console per modificare i tag che sono stati aggiunti al dominio.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nella pagina Domini, scegli il dominio in cui desideri aggiornare i tag.
3. Espandi la sezione Dettagli.
4. In Tag di dominio, scegli Visualizza e modifica i tag di dominio.

> **Note**
>
> Se non ci sono tag aggiunti a questo dominio, la console leggerà Aggiungi tag di dominio.

5. Nei campi Key (Chiave) e Value (Valore), aggiornare i valori di ogni campo in base alle esigenze. Ad esempio, per la chiave **`Name`**, in Value (Valore), modificare **`Test`** in **`Prod`**.
6. Scegli Aggiorna dominio.

Rimuovi i tag da un dominio (console)

Puoi utilizzare la console per eliminare i tag dai domini.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.

2. Nella pagina Domini, scegli il dominio in cui desideri rimuovere i tag.
3. Espandi la sezione Dettagli.
4. In Tag di dominio, scegli Visualizza e modifica i tag di dominio.

> **Note**
>
> Se non ci sono tag aggiunti a questo dominio, la console leggerà Aggiungi tag di dominio.

5. Accanto alla chiave e al valore per ogni tag che desideri eliminare, scegli Rimuovi.
6. Scegli Aggiorna dominio.

Utilizzo CodeArtifact con Cargo

Questi argomenti descrivono come usare Cargo, il gestore di pacchetti Rust, con CodeArtifact.

> **Note**
>
> CodeArtifact supporta solo Cargo 1.74.0 e versioni successive. Cargo 1.74.0 è la prima versione che supporta l'autenticazione su un repository. CodeArtifact

Argomenti

- Configura e usa Cargo con CodeArtifact
- Supporto al comando Cargo

Configura e usa Cargo con CodeArtifact

Puoi usare Cargo per pubblicare e scaricare casse dai CodeArtifact repository o per recuperare casse da crates.io, il registro delle casse della community di Rust. Questo argomento descrive come configurare Cargo per l'autenticazione e l'utilizzo di un repository. CodeArtifact

Configura Cargo con CodeArtifact

Per utilizzare Cargo da cui installare e pubblicare casse AWS CodeArtifact, devi prima configurarle con le informazioni del tuo CodeArtifact repository. Segui i passaggi di una delle seguenti procedure per configurare Cargo con le informazioni e le credenziali dell'endpoint CodeArtifact del repository.

Configura Cargo utilizzando le istruzioni della console

Puoi utilizzare le istruzioni di configurazione nella console per connettere Cargo al tuo CodeArtifact repository. Le istruzioni della console forniscono una configurazione Cargo personalizzata per il tuo CodeArtifact repository. Puoi utilizzare questa configurazione personalizzata per configurare Cargo senza dover trovare e inserire le tue CodeArtifact informazioni.

1. Apri la AWS CodeArtifact console su https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel pannello di navigazione, scegli Repository, quindi scegli un repository per connetterti a Cargo.

3. Scegli Visualizza le istruzioni di connessione.
4. Scegli il tuo sistema operativo.
5. Scegli Cargo.
6. Segui le istruzioni generate per connettere Cargo al tuo CodeArtifact repository.

Configura Cargo manualmente

Se non puoi o non vuoi usare le istruzioni di configurazione della console, puoi utilizzare le seguenti istruzioni per connettere Cargo al tuo CodeArtifact repository manualmente.

macOS and Linux

Per configurare Cargo con CodeArtifact, devi definire il tuo CodeArtifact repository come registro nella configurazione Cargo e fornire le credenziali.

- Sostituiscilo *my_registry* con il nome del tuo registro.
- *my_domain*Sostituiscilo con il tuo nome di CodeArtifact dominio.
- Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
- *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

Copia la configurazione per pubblicare e scaricare i pacchetti Cargo nel tuo repository e salvala nel `~/.cargo/config.toml` file per una configurazione a livello di sistema o `.cargo/config.toml` per una configurazione a livello di progetto:

```
[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

Per configurare Cargo con CodeArtifact, devi definire il tuo CodeArtifact repository come registro nella configurazione Cargo e fornire le credenziali.

- Sostituiscilo *my_registry* con il nome del tuo registro.
- *my_domain*Sostituiscilo con il tuo nome di CodeArtifact dominio.
- Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
- *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

Copia la configurazione per scaricare solo i pacchetti Cargo dal tuo repository e salvala nel `%USERPROFILE%\.cargo\config.toml` file per una configurazione a livello di sistema o `.cargo\config.toml` per una configurazione a livello di progetto:

```
[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. Per configurare Cargo con CodeArtifact, devi definire il tuo CodeArtifact repository come registro nella configurazione Cargo e fornire le credenziali.

   - Sostituiscilo *my_registry* con il nome del tuo registro.
   - *my_domain*Sostituiscilo con il tuo nome di CodeArtifact dominio.
   - Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.

- *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

Copia la configurazione per pubblicare e scaricare i pacchetti Cargo nel tuo repository e salvala nel `%USERPROFILE%\.cargo\config.toml` file per una configurazione a livello di sistema o `.cargo\config.toml` per una configurazione a livello di progetto.

Si consiglia di utilizzare il provider di credenziali`cargo:token`, che utilizza le credenziali memorizzate nel file. `~/.cargo/credentials.toml` Potresti riscontrare un errore durante `cargo publish` l'utilizzo `cargo:token-from-stdout` perché il client Cargo non taglia correttamente il token di autorizzazione durante. `cargo publish`

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/cargo/my_repo/"
credential-provider = "cargo:token"

[registry]
default = "my_registry"

[source.crates-io]
replace-with = "my_registry"
```

2. Per pubblicare i pacchetti Cargo nel tuo repository con Windows, devi utilizzare il CodeArtifact `get-authorization-token` comando e il `login` comando Cargo per recuperare un token di autorizzazione e le tue credenziali.

   - *my_registry*Sostituiscilo con il nome del registro come definito in. `[registries.my_registry]`
   - *my_domain*Sostituiscilo con il tuo nome di CodeArtifact dominio.
   - Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.

```
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**
>
> Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

La `[registries.my_registry]` sezione dell'esempio precedente definisce un registro con *my_registry* `index` e fornisce `credential-provider` informazioni.

- `index`specifica l'URL dell'indice del registro, che è l'endpoint del CodeArtifact repository che termina con un. `/` Il `sparse+` prefisso è obbligatorio per i registri che non sono repository Git.

  > **Note**
  >
  > Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

- `credential-provider`specifica il fornitore di credenziali per il registro specificato. Se `credential-provider` non è impostato, `registry.global-credential-providers` verranno utilizzati i provider inclusi. Se impostata `credential-provider` `sucargo:token-from-stdout`, il client Cargo recupererà automaticamente il nuovo token di autorizzazione durante la pubblicazione o il download dal CodeArtifact repository, pertanto non è necessario aggiornare manualmente il token di autorizzazione ogni 12 ore.

La `[registry]` sezione definisce il registro predefinito utilizzato.

- `default`specifica il nome del registro definito in`[registries.my_registry]`, da utilizzare per impostazione predefinita durante la pubblicazione o il download dal CodeArtifact repository.

La `[source.crates-io]` sezione definisce il registro predefinito utilizzato quando non ne viene specificato uno.

- `replace-with = "my_registry"`sostituisce il registro pubblico, crates.io con il CodeArtifact repository definito in. `[registries.my_registry]` Questa configurazione è consigliata se è necessario richiedere pacchetti dalla connessione esterna come crates.io.

  Per ottenere tutti i vantaggi di CodeArtifact, come il controllo dell'origine dei pacchetti che previene gli attacchi legati alla confusione delle dipendenze, si consiglia di utilizzare la sostituzione del

codice sorgente. Con la sostituzione dei sorgenti, invia tramite CodeArtifact proxy tutte le richieste alla connessione esterna e copia il pacchetto dalla connessione esterna al tuo repository. Senza la sostituzione del codice sorgente, il client Cargo recupererà direttamente il pacchetto in base alla configurazione nel `Cargo.toml` file del progetto. Se una dipendenza non è contrassegnata con`registry=my_registry`, il client Cargo la recupererà direttamente da crates.io senza comunicare con il vostro repository. CodeArtifact

> **Note**
>
> Se inizi a utilizzare la sostituzione del codice sorgente e poi aggiorni il file di configurazione per non utilizzare la sostituzione del codice sorgente, potresti riscontrare degli errori. Anche lo scenario opposto può causare errori. Pertanto, si consiglia di evitare di modificare la configurazione del progetto.

Installazione delle casse Cargo

Utilizza le seguenti procedure per installare le casse Cargo da un CodeArtifact repository o da crates.io.

Installa le casse Cargo da CodeArtifact

Puoi utilizzare la CLI Cargo (`cargo`) per installare rapidamente una versione specifica di una cassa Cargo dal tuo repository. CodeArtifact

Per installare casse Cargo da un repository con CodeArtifact **`cargo`**

1. In caso contrario, segui i passaggi indicati Configura e usa Cargo con CodeArtifact per configurare la `cargo` CLI per utilizzare il tuo CodeArtifact repository con le credenziali appropriate.
2. Usa il seguente comando per installare Cargo crates da: CodeArtifact

   ```
   cargo add my_cargo_package@1.0.0
   ```

   Per ulteriori informazioni, consulta cargo add in The Cargo Book.

Publishing Cargo casse su CodeArtifact

Utilizza la seguente procedura per pubblicare casse Cargo in un CodeArtifact repository utilizzando la CLI`cargo`.

1. In caso contrario, segui i passaggi indicati Configura e usa Cargo con CodeArtifact per configurare la `cargo` CLI per utilizzare il tuo CodeArtifact repository con le credenziali appropriate.
2. Usa il seguente comando per pubblicare le casse Cargo in un repository: CodeArtifact

   ```
   cargo publish
   ```

   Per ulteriori informazioni, vedete cargo publish in The Cargo Book.

Supporto al comando Cargo

Le sezioni seguenti riassumono i comandi Cargo supportati dai CodeArtifact repository, oltre a comandi specifici che non sono supportati.

Indice

- Comandi supportati che richiedono l'accesso al registro
- Comandi non supportati

Comandi supportati che richiedono l'accesso al registro

Questa sezione elenca i comandi Cargo in cui il client Cargo richiede l'accesso al registro con cui è stato configurato. È stato verificato che questi comandi funzionino correttamente quando vengono richiamati su un CodeArtifact repository.

Comando	Descrizione
costruire	Crea pacchetti locali e le loro dipendenze.
controlla	Verifica la presenza di errori nei pacchetti locali e nelle loro dipendenze.
recupera	Recupera le dipendenze di un pacchetto.

Comando	Descrizione
pubblicare	Pubblica un pacchetto nel registro.

Comandi non supportati

Questi comandi Cargo non sono supportati dai CodeArtifact repository.

Comando	Descrizione
proprietario	Gestisce i proprietari della cassa nel registro.
cerca	Cerca i pacchetti nel registro.

Utilizzo CodeArtifact con Maven

Il formato di repository Maven è utilizzato da molti linguaggi diversi, tra cui Java, Kotlin, Scala e Clojure. È supportato da molti strumenti di compilazione diversi, tra cui Maven, Gradle, Scala SBT, Apache Ivy e Leiningen.

Abbiamo testato e confermato la compatibilità con le seguenti versioni: CodeArtifact

- Ultima versione di Maven: 3.6.3.
- È stata testata anche l'ultima versione di Gradle: 6.4.1. La 5.5.1 è stata testata.
- È stata testata anche l'ultima versione di Clojure: 1.11.1.

Argomenti

- Uso CodeArtifact con Gradle
- Usare CodeArtifact con mvn
- Utilizzare CodeArtifact con deps.edn
- Pubblicazione con curl
- Usa i checksum Maven
- Usa le istantanee di Maven
- Richiesta di pacchetti Maven da upstream e connessioni esterne
- Risoluzione dei problemi con Maven

Uso CodeArtifact con Gradle

Dopo aver inserito il token di CodeArtifact autenticazione in una variabile di ambiente, come descritto in Passa un token di autenticazione utilizzando una variabile di ambiente, segui queste istruzioni per utilizzare i pacchetti Maven da e pubblicare nuovi pacchetti in un repository. CodeArtifact

Argomenti

- Recupera le dipendenze
- Recupera i plugin
- Pubblica artefatti
- Esegui una build Gradle in IntelliJ IDEA

Recupera le dipendenze

Per recuperare le dipendenze da CodeArtifact una build Gradle, utilizzate la seguente procedura.

Per recuperare le dipendenze da una build Gradle CodeArtifact

1. In caso contrario, crea e archivia un token di CodeArtifact autenticazione in una variabile di ambiente seguendo la procedura in. Passa un token di autenticazione utilizzando una variabile di ambiente
2. Aggiungete una `maven` sezione alla `repositories` sezione del `build.gradle` file di progetto.

```
maven {
         url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
         credentials {
             username "aws"
             password System.env.CODEARTIFACT_AUTH_TOKEN
         }
}
```

L'`url`esempio precedente è l'endpoint del tuo CodeArtifact repository. Gradle utilizza l'endpoint per connettersi al tuo repository. Nell'esempio, `my_domain` è il nome del tuo dominio, `111122223333` è l'ID del proprietario del dominio ed `my_repo` è il nome del tuo repository. È possibile recuperare l'endpoint di un repository utilizzando il comando. `get-repository-endpoint` AWS CLI

Ad esempio, con un repository denominato *`my_repo`* all'interno di un dominio denominato*`my_domain`*, il comando è il seguente:

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format maven
```

Il `get-repository-endpoint` comando restituirà l'endpoint del repository:

```
url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
```

L'`credentials`oggetto nell'esempio precedente include il token di CodeArtifact autenticazione creato nel passaggio 1 che Gradle utilizza per l'autenticazione. CodeArtifact

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

3. (Facoltativo): per utilizzare il CodeArtifact repository come unica fonte per le dipendenze del progetto, rimuovi tutte le altre sezioni in from. `repositories` `build.gradle` Se hai più di un repository, Gradle cerca in ogni repository le dipendenze nell'ordine in cui sono elencate.
4. Dopo aver configurato il repository, puoi aggiungere le dipendenze del progetto alla sezione con la sintassi Gradle standard. `dependencies`

```
dependencies {
    implementation 'com.google.guava:guava:27.1-jre'
    implementation 'commons-cli:commons-cli:1.4'
    testImplementation 'org.testng:testng:6.14.3'
}
```

Recupera i plugin

Per impostazione predefinita, Gradle risolverà i plugin dal Gradle Plugin Portal pubblico. Per estrarre i plugin da un CodeArtifact repository, utilizzate la seguente procedura.

Per estrarre i plugin da un repository CodeArtifact

1. Se non l'hai fatto, crea e archivia un token di CodeArtifact autenticazione in una variabile di ambiente seguendo la procedura in. Passa un token di autenticazione utilizzando una variabile di ambiente
2. Aggiungi un `pluginManagement` blocco al tuo `settings.gradle` file. Il `pluginManagement` blocco deve apparire prima di qualsiasi altra istruzione`settings.gradle`, vedi il seguente frammento:

```
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
            }
        }
    }
}
```

Ciò assicurerà che Gradle risolva i plugin dal repository specificato. Il repository deve avere un repository upstream con una connessione esterna al Gradle Plugin Portal (ad esempio`gradle-plugins-store`) in modo che i plugin Gradle comunemente richiesti siano disponibili per la build. Per ulteriori informazioni, consultate la documentazione di Gradle.

Pubblica artefatti

Questa sezione descrive come pubblicare una libreria Java creata con Gradle in un repository. CodeArtifact

Innanzitutto, aggiungi il `maven-publish` plugin alla `plugins` sezione del file del `build.gradle` progetto.

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

Quindi, aggiungi una `publishing` sezione al `build.gradle` file di progetto.

```
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
            }
        }
    }
}
```

Il `maven-publish` plugin genera un file POM basato su `groupIdartifactId`, e `version` specificato nella `publishing` sezione.

Una volta `build.gradle` completate queste modifiche, esegui il comando seguente per creare il progetto e caricarlo nel repository.

```
./gradlew publish
```

`list-package-versions`Utilizzatelo per verificare che il pacchetto sia stato pubblicato correttamente.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven\
  --namespace com.company.framework --package my-package-name
```

Output di esempio:

```
{
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "example",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Per ulteriori informazioni, consulta questi argomenti sul sito Web di Gradle:

- Creazione di librerie Java

- Pubblicazione di un progetto come modulo

Esegui una build Gradle in IntelliJ IDEA

È possibile eseguire una build Gradle in IntelliJ IDEA da cui estrae le dipendenze. CodeArtifact Per eseguire l'autenticazione CodeArtifact, è necessario fornire a Gradle un token di autorizzazione. CodeArtifact Esistono tre metodi per fornire un token di autenticazione.

- Metodo 1: memorizzazione del token di autenticazione in. `gradle.properties` Utilizzate questo metodo se siete in grado di sovrascrivere o aggiungere elementi al contenuto del `gradle.properties` file.
- Metodo 2: memorizzazione del token di autenticazione in un file separato. Utilizzate questo metodo se non desiderate modificare il `gradle.properties` file.
- Metodo 3: generazione di un nuovo token di autenticazione per ogni esecuzione eseguendolo `aws` come script in linea in. `build.gradle` Usa questo metodo se vuoi che lo script Gradle recuperi un nuovo token ad ogni esecuzione. Il token non verrà archiviato nel file system.

Token stored in gradle.properties

Metodo 1: memorizzazione del token di autenticazione in **`gradle.properties`**

> **Note**
>
> L'esempio mostra il `gradle.properties` file che si trova in`GRADLE_USER_HOME`.

1. Aggiorna il `build.gradle` file con il seguente frammento:

```
repositories {
    maven {
            url
 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username "aws"
                password "$codeartifactToken"
            }
    }
```

```
}
```

2. Per recuperare i plugin CodeArtifact, aggiungi un `pluginManagement` blocco al tuo file. `settings.gradle` Il `pluginManagement` blocco deve apparire prima di qualsiasi altra istruzione in. `settings.gradle`

```
pluginManagement {
    repositories {
        maven {
            name 'my_repo'
            url
 'https://my_domain-111122223333.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password "$codeartifactToken"
            }
        }
    }
}
```

3. Recupera un token di CodeArtifact autenticazione:

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. Scrivi il token di autenticazione nel file: `gradle.properties`

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > ~/.gradle/gradle.properties
```

Token stored in separate file

Metodo 2: memorizzazione del token di autenticazione in un file separato

1. Aggiorna il `build.gradle` file con il seguente frammento:

```
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. Per recuperare i plugin CodeArtifact, aggiungi un `pluginManagement` blocco al tuo file. `settings.gradle` Il `pluginManagement` blocco deve apparire prima di qualsiasi altra istruzione in. `settings.gradle`

```
pluginManagement {
    def props = new Properties()
    file("file").withInputStream { props.load(it) }
    repositories {
        maven {
            name 'my_repo'
            url
 'https://my_domain-111122223333.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password props.getProperty("codeartifactToken")
            }
        }
    }
}
```

3. Recupera un token di CodeArtifact autenticazione:

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. Scrivi il token di autenticazione nel file specificato nel tuo file: `build.gradle`

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > file
```

Token generated for each run in build.gradle

Metodo 3: generazione di un nuovo token di autenticazione per ogni esecuzione eseguendolo **`aws`** come script in linea in **`build.gradle`**

1. Aggiorna il `build.gradle` file con il seguente frammento:

```
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. Per recuperare i plugin CodeArtifact, aggiungi un `pluginManagement` blocco al tuo file. `settings.gradle` Il `pluginManagement` blocco deve apparire prima di qualsiasi altra istruzione in. `settings.gradle`

```
pluginManagement {
    def codeartifactToken = "aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name".execute().text
    repositories {
        maven {
            name 'my_repo'
            url
 'https://my_domain-111122223333.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username 'aws'
                password codeartifactToken
            }
        }
    }
```

```
}
```

Usare CodeArtifact con mvn

Si usa il `mvn` comando per eseguire le build di Maven. Questa sezione mostra come configurare l'uso di un `mvn` repository. CodeArtifact

Argomenti

- Recupera le dipendenze
- Pubblica artefatti
- Pubblica artefatti di terze parti
- Limita i download delle dipendenze di Maven a un repository CodeArtifact
- Informazioni sul progetto Apache Maven

Recupera le dipendenze

`mvn`Per configurare il recupero delle dipendenze da un CodeArtifact repository, devi modificare il file di configurazione di Maven e, facoltativamente`settings.xml`, il POM del tuo progetto.

1. Se non l'hai fatto, crea e archivia un token di CodeArtifact autenticazione in una variabile di ambiente come descritto in per configurare l'autenticazione nel Passa un token di autenticazione utilizzando una variabile di ambiente tuo repository. CodeArtifact
2. In `settings.xml` (in genere si trova in`~/.m2/settings.xml`), aggiungi una `<servers>` sezione con un riferimento alla variabile di `CODEARTIFACT_AUTH_TOKEN` ambiente in modo che Maven passi il token nelle richieste HTTP.

```
<settings>
...
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
```

```
</settings>
```

3. Aggiungi l'endpoint URL per il tuo CodeArtifact repository in un elemento. `<repository>` Puoi farlo nel file `settings.xml` POM del tuo progetto.

   Puoi recuperare l'endpoint del tuo repository usando il comando. `get-repository-endpoint` AWS CLI

   Ad esempio, con un repository denominato *`my_repo`* all'interno di un dominio denominato*`my_domain`*, il comando è il seguente:

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --repository my_repo --
   format maven
   ```

   Il `get-repository-endpoint` comando restituirà l'endpoint del repository:

   ```
   url 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   maven/my_repo/'
   ```

   > **Note**
   >
   > Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

   Aggiungi l'endpoint del repository come segue. `settings.xml`

   ```
   <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>
```

In alternativa, è possibile aggiungere la `<repositories>` sezione a un file POM di progetto da utilizzare solo CodeArtifact per quel progetto.

```
<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**
>
> È possibile utilizzare qualsiasi valore nell'`<id>`elemento, ma deve essere lo stesso in entrambi `<server>` gli `<repository>` elementi. Ciò consente di includere le credenziali specificate nelle richieste di CodeArtifact.

Dopo aver apportato queste modifiche alla configurazione, puoi creare il progetto.

```
mvn compile
```

Maven registra l'URL completo di tutte le dipendenze scaricate sulla console.

```
[INFO] ------------------< com.example.example:myapp >------------------
[INFO] Building myapp 1.0
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom
```

```
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom (11 kB at 3.9 kB/s)
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom (68 kB at 123
 kB/s)
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar (54 kB at 134 kB/s)
```

Pubblica artefatti

Per pubblicare un artefatto Maven in un CodeArtifact repository, devi anche modificare e progettare POM. `mvn` `~/.m2/settings.xml`

1. Se non l'hai fatto, crea e archivia un token di CodeArtifact autenticazione in una variabile di ambiente come descritto in Passa un token di autenticazione utilizzando una variabile di ambiente per configurare l'autenticazione nel tuo repository. CodeArtifact
2. Aggiungi una `<servers>` sezione a `settings.xml` con un riferimento alla variabile di `CODEARTIFACT_AUTH_TOKEN` ambiente in modo che Maven passi il token nelle richieste HTTP.

   ```
   <settings>
   ...
       <servers>
           <server>
               <id>codeartifact</id>
               <username>aws</username>
               <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
           </server>
       </servers>
   ...
   </settings>
   ```

3. Aggiungi una `<distributionManagement>` sezione a quella del tuo progetto. `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>
```

Dopo aver apportato queste modifiche alla configurazione, puoi creare il progetto e pubblicarlo nel repository specificato.

```
mvn deploy
```

`list-package-versions`Utilizzatelo per verificare che il pacchetto sia stato pubblicato correttamente.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven \
  --namespace com.company.framework --package my-package-name
```

Output di esempio:

```
{
    "defaultDisplayVersion": null,
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "my-package-name",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Pubblica artefatti di terze parti

Puoi pubblicare artefatti Maven di terze parti in un repository con. CodeArtifact `mvn deploy:deploy-file` Questo può essere utile per gli utenti che desiderano pubblicare artefatti e dispongono solo di file JAR e non hanno accesso al codice sorgente del pacchetto o ai file POM.

Il `mvn deploy:deploy-file` comando genererà un file POM basato sulle informazioni passate nella riga di comando.

Pubblica artefatti Maven di terze parti

1. In caso contrario, crea e archivia un token di CodeArtifact autenticazione in una variabile di ambiente come descritto in Passa un token di autenticazione utilizzando una variabile di ambiente Per configurare l'autenticazione nel tuo repository. CodeArtifact
2. Crea un `~/.m2/settings.xml` file con i seguenti contenuti:

```
<settings>
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
</settings>
```

3. Eseguire il comando `mvn deploy:deploy-file`:

```
mvn deploy:deploy-file -DgroupId=commons-cli          \
-DartifactId=commons-cli       \
-Dversion=1.4                  \
-Dfile=./commons-cli-1.4.jar   \
-Dpackaging=jar                \
-DrepositoryId=codeartifact    \
-Durl=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/repo-name/
```

> **Note**
>
> L'esempio precedente pubblica `commons-cli 1.4`. Modifica gli argomenti GroupiD, artifactID, version e file per pubblicare un JAR diverso.

Queste istruzioni si basano su esempi contenuti nella Guida alla distribuzione di terze parti in un repository remoto tratti dalla documentazione di JARs Apache Maven.

Limita i download delle dipendenze di Maven a un repository CodeArtifact

Se un pacchetto non può essere recuperato da un repository configurato, per impostazione predefinita, il `mvn` comando lo recupera da Maven Central. Aggiungi l'`mirrors`elemento a per utilizzare sempre il tuo `settings.xml` repository. `mvn` 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>
```

Se aggiungi un `mirrors` elemento, devi avere anche un `pluginRepository` elemento nel tuo `settings.xml` or`pom.xml`. L'esempio seguente recupera le dipendenze delle applicazioni e i plugin Maven da un repository. CodeArtifact

```
<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>
```

L'esempio seguente recupera le dipendenze dell'applicazione da un CodeArtifact repository e recupera i plugin Maven da Maven Central.

```
<profiles>
   <profile>
     <id>default</id>
     ...
     <pluginRepositories>
       <pluginRepository>
         <id>central-plugins</id>
         <name>Central Plugins</name>
         <url>https://repo.maven.apache.org/maven2/</url>
         <releases>
             <enabled>true</enabled>
         </releases>
         <snapshots>
             <enabled>true</enabled>
         </snapshots>
       </pluginRepository>
     </pluginRepositories>
   ....
   </profile>
 </profiles>
```

Informazioni sul progetto Apache Maven

Per ulteriori informazioni su Maven, consulta questi argomenti sul sito Web del progetto Apache Maven:

- Configurazione di più repository
- Riferimento alle impostazioni
- Gestione della distribuzione
- Profili

Utilizzare CodeArtifact con deps.edn

Si utilizza `deps.edn` with `clj` per gestire le dipendenze per i progetti Clojure. Questa sezione mostra come configurare l'uso di un `deps.edn` repository. CodeArtifact

Argomenti

- Recupera le dipendenze
- Pubblica artefatti

Recupera le dipendenze

`Clojure`Per configurare il recupero delle dipendenze da un CodeArtifact repository, è necessario modificare il file di configurazione di Maven,. `settings.xml`

1. In`oltre``settings.xml`, aggiungi una `<servers>` sezione con un riferimento alla variabile di `CODEARTIFACT_AUTH_TOKEN` ambiente in modo che Clojure passi il token nelle richieste HTTP.

   > **Note**
   >
   > Clojure si aspetta che il file settings.xml si trovi in. `~/.m2/settings.xml` Se altrove, crea il file in questa posizione.

   ```
   <settings>
   ...
       <servers>
   ```

```
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

2. Se non ne hai già uno, genera un file XML POM da utilizzare `clj -Spom` per il tuo progetto.
3. Nel file `deps.edn` di configurazione, aggiungi un repository corrispondente all'id del server di Maven. `settings.xml`

```
:mvn/repos {
  "clojars" nil
  "central" nil
  "codeartifact" {:url "https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_repo/"}
}
```

> **Note**
>
> - `tools.deps`garantisce che i `clojars` repository `central` and verranno controllati prima per le librerie Maven. Successivamente, verranno controllati gli altri repository elencati in. `deps.edn`
> - Per impedire il download diretto da Clojars e Maven Central, `central` è necessario impostarlo su. `clojars nil`

Assicurati di avere il token di CodeArtifact autenticazione in una variabile di ambiente (vedi). Passa un token di autenticazione utilizzando una variabile di ambiente Quando si crea il pacchetto dopo queste modifiche, `deps.edn` verranno recuperate le dipendenze in. CodeArtifact

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

Pubblica artefatti

1. Aggiorna le impostazioni di Maven e includile CodeArtifact come `deps.edn` server riconosciuto da Maven (vedi). Recupera le dipendenze Puoi usare uno strumento come deps-deploy per caricare artefatti su. CodeArtifact
2. Nel tuo`build.clj`, aggiungi un'`deploy`attività per caricare gli artefatti richiesti nel repository precedentemente configurato. `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. Pubblica l'artefatto eseguendo il comando: `clj -T:build deploy`

Per ulteriori informazioni sulla modifica degli archivi predefiniti, vedere Modifica dei repository predefiniti nel Clojure Deps and CLI Reference Rationale.

Pubblicazione con curl

Questa sezione mostra come utilizzare il client HTTP per `curl` pubblicare artefatti Maven in un repository. CodeArtifact La pubblicazione di artefatti con `curl` può essere utile se non si dispone o non si desidera installare il client Maven nei propri ambienti.

Pubblica un artefatto Maven con **`curl`**

1. Recupera un token di CodeArtifact autorizzazione seguendo i passaggi indicati Passa un token di autenticazione utilizzando una variabile di ambiente e torna a questi passaggi.
2. Usa il seguente `curl` comando per pubblicare il JAR in un CodeArtifact repository:

   In ciascuno dei `curl` comandi di questa procedura, sostituite i seguenti segnaposto:

   - Sostituiscilo *`my_domain`* con il tuo CodeArtifact nome di dominio.
   - Sostituiscilo *`111122223333`* con l'ID del proprietario del tuo CodeArtifact dominio.

- Sostituiscilo *us-west-2* con la regione in cui si trova il tuo CodeArtifact dominio.
- *my_repo*Sostituiscilo con il nome CodeArtifact del tuo repository.

```
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**
>
> È necessario anteporre un `@` carattere al valore del `--data-binary` parametro. Quando si inserisce il valore tra virgolette, `@` deve essere incluso tra virgolette.

3. Utilizzate il seguente `curl` comando per pubblicare il POM in un repository: CodeArtifact

```
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. A questo punto, l'artefatto Maven si troverà nel tuo CodeArtifact repository con uno stato di. `Unfinished` Per poter consumare il pacchetto, deve trovarsi nello stato. `Published` Puoi spostare il pacchetto da `Unfinished` a `Published` caricando un `maven-metadata.xml` file sul pacchetto o chiamando l'UpdatePackageVersionsStatus API per modificare lo stato.

   a. Opzione 1: utilizza il seguente `curl` comando per aggiungere un `maven-metadata.xml` file al pacchetto:

   ```
   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
   ```

   Di seguito è riportato un esempio del contenuto di un `maven-metadata.xml` file:

```
<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. Opzione 2: aggiorna lo stato del pacchetto `Published` con l'`UpdatePackageVersionsStatus`API.

```
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
```

Se disponi solo del file JAR di un artefatto, puoi pubblicare una versione del pacchetto consumabile in un CodeArtifact repository utilizzando. `mvn` Questo può essere utile se non avete accesso al codice sorgente o al POM dell'artefatto. Per informazioni dettagliate, vedi Pubblica artefatti di terze parti.

Usa i checksum Maven

Quando un artefatto Maven viene pubblicato in un AWS CodeArtifact repository, il checksum associato a ogni risorsa o file del pacchetto viene utilizzato per convalidare il caricamento. Esempi di risorse sono i file jar, pom e war. Per ogni risorsa, l'artefatto Maven contiene più file di checksum che utilizzano il nome della risorsa con un'estensione aggiuntiva, ad esempio o. `md5` `sha1` Ad esempio, i file di checksum per un file denominato potrebbero essere e. `my-maven-package.jar` `my-maven-package.jar.md5` `my-maven-package.jar.sha1`

> **Note**
>
> Maven usa il termine. `artifact` In questa guida, un pacchetto Maven è uguale a un artefatto Maven. Per ulteriori informazioni, vedi pacchetto.AWS CodeArtifact

Archiviazione con checksum

CodeArtifact non memorizza i checksum di Maven come risorse. Ciò significa che i checksum non appaiono come singole risorse nell'output dell'API. ListPackageVersionAssets Invece, i checksum calcolati da CodeArtifact sono disponibili per ogni risorsa in tutti i tipi di checksum supportati. Ad esempio, parte della risposta alla chiamata ListPackageVersionAssets sulla versione del pacchetto Maven è: `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
    }
}
```

Anche se i checksum non sono archiviati come risorse, i client Maven possono comunque pubblicare e scaricare i checksum nelle posizioni previste. Ad esempio, se si `commons-lang:commons-lang 2.1` trovasse in un repository chiamato`maven-repo`, il percorso URL per il checksum SHA-256 del file JAR sarebbe:

```
/maven/maven-repo/commons-lang/commons-lang/2.1/commons-lang-2.1.jar.sha256
```

Se stai caricando pacchetti Maven esistenti (ad esempio, pacchetti precedentemente archiviati in Amazon S3) su un CodeArtifact client HTTP generico, ad esempio`curl`, non è necessario caricare i checksum. CodeArtifact li genererà automaticamente. Se desideri verificare che gli asset siano stati caricati correttamente, puoi utilizzare l'operazione ListPackageVersionAssets API per confrontare i checksum nella risposta con i valori di checksum originali per ogni risorsa.

I checksum non corrispondono durante la pubblicazione

Oltre agli asset e ai checksum, gli artefatti di Maven contengono anche un file. `maven-metadata.xml` La normale sequenza di pubblicazione per un pacchetto Maven prevede che tutti gli asset e i checksum vengano caricati per primi, seguiti da. `maven-metadata.xml` Ad esempio, la sequenza di pubblicazione per la versione del pacchetto Maven `commons-lang 2.1` descritta in precedenza, presupponendo che il client sia configurato per pubblicare file con checksum SHA-256, sarebbe:

```
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
```

Quando si carica il file di checksum per una risorsa, ad esempio un file JAR, la richiesta di caricamento del checksum avrà esito negativo e restituirà una risposta 400 (Bad Request) se c'è una mancata corrispondenza tra il valore del checksum caricato e il valore del checksum calcolato

da. CodeArtifact Se la risorsa corrispondente non esiste, la richiesta avrà esito negativo con una risposta 404 (Not Found). Per evitare questo errore, dovete prima caricare la risorsa e poi caricare il checksum.

Quando `maven-metadata.xml` viene caricato, CodeArtifact normalmente cambia lo stato della versione del pacchetto Maven da a. `Unfinished` `Published` Se viene rilevata una mancata corrispondenza del checksum per qualsiasi risorsa, CodeArtifact restituirà un 400 (Bad Request) in risposta alla richiesta di pubblicazione. `maven-metadata.xml` Questo errore può causare l'interruzione del caricamento dei file per quella versione del pacchetto da parte del client. Se ciò si verifica e il `maven-metadata.xml` file non viene caricato, le risorse della versione del pacchetto già caricate non possono essere scaricate. Questo perché lo stato della versione del pacchetto non è impostato `Published` e rimane invariato`Unfinished`.

CodeArtifact consente di aggiungere ulteriori risorse a una versione del pacchetto Maven anche dopo che `maven-metadata.xml` è stata caricata e lo stato della versione del pacchetto è stato impostato su. `Published` In questo stato, anche una richiesta di caricamento di un file di checksum non corrispondente fallirà con una risposta 400 (Bad Request). Tuttavia, poiché lo stato della versione del pacchetto è già stato impostato su`Published`, potete scaricare qualsiasi risorsa dal pacchetto, incluse quelle per cui il caricamento del file con il checksum non è riuscito. Quando scaricate un checksum per una risorsa il cui caricamento del file di checksum non è riuscito, il valore del checksum ricevuto dal client sarà il valore del checksum calcolato in CodeArtifact base ai dati della risorsa caricata.

CodeArtifact i confronti tra checksum fanno distinzione tra maiuscole e minuscole e i checksum calcolati da sono formattati in minuscolo. CodeArtifact Pertanto, se il checksum `909FA780F76DA393E992A3D2D495F468` viene caricato, avrà esito negativo con una mancata corrispondenza del checksum perché non lo considera uguale a. CodeArtifact `909fa780f76da393e992a3d2d495f468`

Recupero in seguito a mancate corrispondenze nei checksum

Se il caricamento di un checksum non riesce a causa di una mancata corrispondenza del checksum, prova una delle seguenti operazioni per il ripristino:

- Esegui nuovamente il comando che pubblica l'artefatto Maven. Questo potrebbe funzionare se un problema di rete danneggia il file di checksum. Se questo risolve il problema di rete, il checksum corrisponde e il download ha esito positivo.
- Eliminare la versione del pacchetto e ripubblicarla. Per ulteriori informazioni, DeletePackageVersionsconsulta AWS CodeArtifact API Reference.

Usa le istantanee di Maven

Un'istantanea Maven è una versione speciale di un pacchetto Maven che si riferisce all'ultimo codice filiale di produzione. È una versione di sviluppo che precede la versione di rilascio finale. È possibile identificare una versione istantanea di un pacchetto Maven dal suffisso `SNAPSHOT` aggiunto alla versione del pacchetto. Ad esempio, l'istantanea della versione è. `1.1` `1.1-SNAPSHOT` Per ulteriori informazioni, consulta Cos'è una versione SNAPSHOT? sul sito Web del progetto Apache Maven.

AWS CodeArtifact supporta la pubblicazione e l'utilizzo di istantanee Maven. Le istantanee uniche che utilizzano un numero di versione basato sul tempo sono le uniche istantanee supportate. CodeArtifact non supporta istantanee non univoche generate dai client Maven 2. È possibile pubblicare un'istantanea Maven supportata in qualsiasi repository. CodeArtifact

Argomenti

- Pubblicazione di istantanee in CodeArtifact
- Consumo di versioni istantanee
- Eliminazione delle versioni istantanee
- Pubblicazione di istantanee con curl
- Istantanee e connessioni esterne
- Istantanee e repository upstream

Pubblicazione di istantanee in CodeArtifact

AWS CodeArtifact supporta i modelli di richiesta utilizzati dai client, ad esempio`mvn`, per la pubblicazione di istantanee. Per questo motivo, è possibile seguire la documentazione dello strumento di compilazione o del gestore di pacchetti senza avere una comprensione dettagliata di come vengono pubblicate le istantanee di Maven. Se state facendo qualcosa di più complesso, questa sezione descrive in dettaglio come CodeArtifact gestisce le istantanee.

Quando un'istantanea di Maven viene pubblicata in un CodeArtifact repository, la sua versione precedente viene conservata in una nuova versione chiamata build. Ogni volta che viene pubblicata un'istantanea di Maven, viene creata una nuova versione di build. Tutte le versioni precedenti di un'istantanea vengono mantenute nelle relative versioni di build. Quando viene pubblicata un'istantanea di Maven, lo stato della versione del pacchetto è impostato su `Published` e lo stato della build che contiene la versione precedente è impostato su. `Unlisted` Questo comportamento

si applica solo alle versioni del pacchetto Maven in cui la versione del pacchetto ha come suffisso. `-SNAPSHOT`

Ad esempio, le versioni snapshot di un pacchetto Maven chiamato `com.mycompany.myapp:pkg-1` vengono caricate in un repository chiamato. CodeArtifact `my-maven-repo` La versione snapshot è. `1.0-SNAPSHOT` Finora non è stata pubblicata alcuna versione di`com.mycompany.myapp:pkg-1`. Innanzitutto, le risorse della build iniziale vengono pubblicate nei seguenti percorsi:

```
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
```

Nota che il timestamp `20210728.194552-1` viene generato dal client che pubblica le build di snapshot.

Dopo il caricamento dei file.pom e.jar, l'unica versione esistente nel repository è. `com.mycompany.myapp:pkg-1` `1.0-20210728.194552-1` Ciò accade anche se la versione specificata nel percorso precedente è. `1.0-SNAPSHOT` Lo stato della versione del pacchetto a questo punto è`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"
}
```

Successivamente, il client carica il `maven-metadata.xml` file per la versione del pacchetto:

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

Quando il file maven-metadata.xml viene caricato correttamente, CodeArtifact crea la versione del `1.0-SNAPSHOT` pacchetto e imposta la `1.0-20210728.194552-1` versione su`Unlisted`.

```
aws codeartifact list-package-versions --domain my-domain --repository \
  my-maven-repo --package pkg-1 --namespace com.mycompany.myapp --format maven
{
    "versions": [
        {
            "version": "1.0-20210728.194552-1",
            "revision": "GipMW+599JmwTcTLaXo9YvDsVQ2bcrrk/02rWJhoKUU=",
            "status": "Unlisted"
        },
        {
            "version": "1.0-SNAPSHOT",
            "revision": "tWu8n3IX5HR82vzVZQAxlwcvvA4U/+S80edWNAkil24=",
            "status": "Published"
        }
    ],
    "defaultDisplayVersion": "1.0-SNAPSHOT",
    "format": "maven",
    "package": "pkg-1",
    "namespace": "com.mycompany.myapp"
}
```

A questo punto, la versione snapshot `1.0-SNAPSHOT` può essere utilizzata in una build. Sebbene `com.mycompany.myapp:pkg-1` nel repository siano presenti due versioni`my-maven-repo`, entrambe contengono le stesse risorse.

```
aws codeartifact list-package-version-assets --domain my-domain --repository \
  my-maven-repo --format maven --namespace com.mycompany.myapp \
 --package pkg-1 --package-version 1.0-SNAPSHOT--query 'assets[*].name'
[
     "pkg-1-1.0-20210728.194552-1.jar",
     "pkg-1-1.0-20210728.194552-1.pom"
]
```

L'esecuzione dello stesso `list-package-version-assets` comando mostrato in precedenza con il `--package-version` parametro modificato in `1.0-20210728.194552-1` restituisce un output identico.

Man mano che `1.0-SNAPSHOT` vengono aggiunte altre versioni di al repository, viene creata una nuova versione `Unlisted` del pacchetto per ogni nuova build. Le risorse della versione `1.0-`

SNAPSHOT vengono aggiornate ogni volta in modo che la versione faccia sempre riferimento alla build più recente di quella versione. L'aggiornamento `1.0-SNAPSHOT` con le risorse più recenti viene avviato caricando il `maven-metadata.xml` file per la nuova build.

Consumo di versioni istantanee

Se si richiede un'istantanea, viene restituita la versione con lo stato`Published`. Questa è sempre la versione più recente dell'istantanea di Maven. Puoi anche richiedere una build particolare di un'istantanea utilizzando il numero di versione della build (ad esempio,`1.0-20210728.194552-1`) anziché la versione dell'istantanea (ad esempio,`1.0-SNAPSHOT`) nel percorso URL. Per vedere le versioni di build di un'istantanea Maven, usa l'ListPackageVersions API nella Guida CodeArtifact API e imposta il parametro status su. `Unlisted`

Eliminazione delle versioni istantanee

Per eliminare tutte le versioni di build di un'istantanea di Maven, usa l'DeletePackageVersionsAPI, specificando le versioni che desideri eliminare.

Pubblicazione di istantanee con curl

Se disponi di versioni di snapshot esistenti archiviate in Amazon Simple Storage Service (Amazon S3) o in un altro prodotto di artifact repository, potresti volerle ripubblicare su. AWS CodeArtifact A causa del modo in cui CodeArtifact supporta le istantanee Maven (vediPubblicazione di istantanee in CodeArtifact), la pubblicazione di istantanee con un client HTTP generico, come ad esempio, `curl` è più complessa della pubblicazione delle versioni di rilascio di Maven, come descritto in. Pubblicazione con curl Tieni presente che questa sezione non è rilevante se stai creando e distribuendo versioni di snapshot con un client Maven come o. `mvn` `gradle` È necessario seguire la documentazione relativa a quel client.

La pubblicazione di una versione snapshot implica la pubblicazione di una o più build di una versione snapshot. In CodeArtifact, se ci sono n build di una versione snapshot, ci saranno n+1 CodeArtifact versioni: n versioni di build tutte con uno stato di `Unlisted` e una versione snapshot (l'ultima build pubblicata) con uno stato di. `Published` La versione snapshot (ovvero la versione con una stringa di versione che contiene «-SNAPSHOT») contiene un set di risorse identico all'ultima build pubblicata. Il modo più semplice per creare questa struttura utilizzando `curl` è il seguente:

1. Pubblica tutte le risorse di tutte le build utilizzando`curl`.

2. Pubblica il `maven-metadata.xml` file dell'ultima build (ovvero la build con il timbro data-ora più recente) con. `curl` Questo creerà una versione con «`-SNAPSHOT`» nella stringa della versione e con il set di risorse corretto.
3. Utilizza l'UpdatePackageVersionsStatusAPI per impostare lo stato di tutte le versioni di build non più recenti su`Unlisted`.

Utilizzate i seguenti `curl` comandi per pubblicare risorse snapshot (come i file.jar e .pom) per la versione snapshot di un pacchetto: `1.0-SNAPSHOT` `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
```

Quando si utilizzano questi esempi:

- *`my_domain`*Sostituiscilo con il tuo nome di CodeArtifact dominio.
- Sostituiscilo *`111122223333`* con l' Account AWS ID del proprietario del tuo CodeArtifact dominio.
- Sostituiscilo *`us-west-2`* con quello Regione AWS in cui si trova il tuo CodeArtifact dominio.
- *`my_maven_repo`*Sostituiscilo con il nome CodeArtifact del tuo repository.

> **Important**
>
> È necessario anteporre il carattere al valore del `--data-binary` parametro. `@` Quando si inserisce il valore tra virgolette, `@` deve essere incluso tra virgolette.

Potresti avere più di due risorse da caricare per ogni build. Ad esempio, potrebbero esserci Javadoc e file JAR di origine oltre ai file JAR principali e. `pom.xml` Non è necessario pubblicare file di checksum per gli asset della versione del pacchetto perché genera CodeArtifact automaticamente i checksum per ogni risorsa caricata. Per verificare che le risorse siano state caricate correttamente, recuperate i checksum generati utilizzando il `list-package-version-assets` comando e confrontateli con i checksum originali. Per ulteriori informazioni su come CodeArtifact gestisce i checksum di Maven, consulta. Usa i checksum Maven

Utilizzate il seguente comando curl per pubblicare il `maven-metadata.xml` file per l'ultima versione di build:

```
curl --user "aws:$CODEARTIFACT_AUTH_TOKEN" -H "Content-Type: application/octet-stream"
 \
     -X PUT https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_maven_repo/com/mycompany/app/pkg-1/1.0-SNAPSHOT/maven-metadata.xml \
     --data-binary @maven-metadata.xml
```

Il `maven-metadata.xml` file deve fare riferimento ad almeno una delle risorse nell'ultima versione di build dell'`<snapshotVersions>`elemento. Inoltre, il `<timestamp>` valore deve essere presente e deve corrispondere al timestamp riportato nei nomi dei file delle risorse. Ad esempio, per la `20210729.171330-2` build pubblicata in precedenza, il contenuto di `maven-metadata.xml` sarebbe:

```
<?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>
```

Dopo `maven-metadata.xml` la pubblicazione, l'ultimo passaggio consiste nell'impostare tutte le altre versioni di build (ovvero tutte le versioni di build tranne la build più recente) per avere lo stato della versione del pacchetto di`Unlisted`. Ad esempio, se la `1.0-SNAPSHOT` versione ha due build, di cui la prima build è`20210728.194552-1`, il comando su cui impostare quella build `Unlisted` è:

```
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
```

Istantanee e connessioni esterne

Le istantanee di Maven non possono essere recuperate da un archivio pubblico Maven tramite una connessione esterna. AWS CodeArtifact supporta solo l'importazione di versioni di release di Maven.

Istantanee e repository upstream

In generale, le istantanee di Maven funzionano allo stesso modo delle versioni di release di Maven se utilizzate con repository upstream, ma esiste una limitazione se si prevede di pubblicare istantanee della stessa versione del pacchetto su due repository che hanno una relazione a monte. Ad esempio, supponiamo che ci siano due repository in un dominio e che dove sia un upstream di. AWS CodeArtifact R U U R Se pubblichi una nuova build inR, quando un client Maven richiede l'ultima build di quella versione snapshot, CodeArtifact restituisce la versione più recente da. U Questo può essere inaspettato poiché l'ultima versione è ora disponibile, no. R U Esistono due modi per evitarlo:

1. Non pubblicate build di una versione istantanea come `1.0-SNAPSHOT` inR, se `1.0-SNAPSHOT` esiste in. U
2. Usa i controlli di origine CodeArtifact del pacchetto per disabilitare gli upstream su quel pacchetto in. R Quest'ultimo ti permetterà di pubblicare versioni di `1.0-SNAPSHOT` inR, ma R impedirà anche di recuperare altre versioni di quel pacchetto U che non siano già state conservate.

Richiesta di pacchetti Maven da upstream e connessioni esterne

Importazione di nomi di risorse standard

Quando importa una versione di pacchetto Maven da un repository pubblico, come Maven Central, CodeArtifact AWS tenta di importare tutte le risorse in quella versione del pacchetto. Come descritto in, l'importazione avviene quando: Richiesta di una versione del pacchetto con repository upstream

- Un client richiede una risorsa Maven da un repository. CodeArtifact
- La versione del pacchetto non è già presente nel repository o nei suoi upstream.
- Esiste una connessione esterna raggiungibile a un repository Maven pubblico.

Anche se il client potrebbe aver richiesto solo una risorsa, CodeArtifact tenta di importare tutte le risorse che riesce a trovare per quella versione del pacchetto. Il modo in cui CodeArtifact scopre quali risorse sono disponibili per una versione del pacchetto Maven dipende dal particolare archivio pubblico. Alcuni repository Maven pubblici supportano la richiesta di un elenco di risorse, ma altri no. Per i repository che non forniscono un modo per elencare le risorse, CodeArtifact genera una serie di nomi di risorse che è probabile che esistano. Ad esempio, quando viene richiesta una risorsa della versione `junit 4.13.2` del pacchetto Maven, CodeArtifact tenterà di importare le seguenti risorse:

- `junit-4.13.2.pom`
- `junit-4.13.2.jar`
- `junit-4.13.2-javadoc.jar`
- `junit-4.13.2-sources.jar`

Importazione di nomi di asset non standard

Quando un client Maven richiede una risorsa che non corrisponde a uno dei modelli sopra descritti, CodeArtifact verifica se quella risorsa è presente nell'archivio pubblico. Se la risorsa è presente, verrà importata e aggiunta al record della versione del pacchetto esistente, se esistente. Ad esempio, la versione del pacchetto Maven `com.android.tools.build:aapt2 7.3.1-8691043` contiene le seguenti risorse:

- `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`

Quando un client richiede il file POM, se non CodeArtifact è in grado di elencare gli asset della versione del pacchetto, il POM sarà l'unica risorsa importata. Questo perché nessuna delle altre risorse corrisponde ai modelli di denominazione degli asset standard. Tuttavia, quando il client richiede una delle risorse JAR, tale risorsa verrà importata e aggiunta alla versione del pacchetto esistente archiviata in CodeArtifact. Le versioni del pacchetto presenti sia nel repository più a valle (l'archivio verso cui il client ha effettuato la richiesta) che nel repository con la connessione esterna collegata verranno aggiornate per contenere la nuova risorsa, come descritto in. Package retention dai repository upstream

Normalmente, una volta che una versione del pacchetto viene conservata in un CodeArtifact repository, non è influenzata dalle modifiche apportate ai repository upstream. Per ulteriori informazioni, consulta Package retention dai repository upstream. Tuttavia, il comportamento degli asset Maven con nomi non standard descritto in precedenza è un'eccezione a questa regola. Sebbene la versione downstream del pacchetto non cambierà senza che un client richieda una risorsa aggiuntiva, in questa situazione, la versione del pacchetto conservata viene modificata dopo essere stata inizialmente mantenuta e quindi non è immutabile. Questo comportamento è necessario perché altrimenti le risorse Maven con nomi non standard non sarebbero accessibili tramite. CodeArtifact Il comportamento si attiva anche se vengono aggiunti a una versione del pacchetto Maven su un repository pubblico dopo che la versione del pacchetto è stata conservata in un repository. CodeArtifact

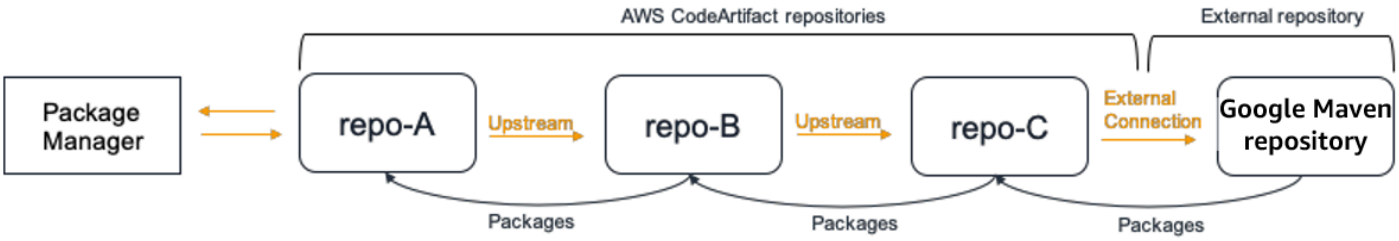
Verifica delle origini delle risorse

Quando si aggiunge una nuova risorsa a una versione del pacchetto Maven precedentemente conservata, CodeArtifact conferma che l'origine della versione del pacchetto conservata è la stessa dell'origine della nuova risorsa. Ciò impedisce la creazione di una versione del pacchetto «mista» in cui risorse diverse provengono da diversi archivi pubblici. Senza questo controllo, potrebbe verificarsi una combinazione di risorse se una versione del pacchetto Maven viene pubblicata su più di un repository pubblico e tali repository fanno parte del grafico upstream di un CodeArtifact repository.

Importazione di nuove risorse e dello stato delle versioni dei pacchetti nei repository upstream

Lo stato della versione del pacchetto delle versioni dei pacchetti negli archivi upstream può CodeArtifact impedire il mantenimento di tali versioni nei repository downstream.

Ad esempio, supponiamo che un dominio abbia tre repository:`repo-A`,, e `repo-B``repo-C`, dove si trova un upstream di e `repo-B` sta a monte di. `repo-A` `repo-C` `repo-B`



La versione `7.3.1` del pacchetto Maven `com.android.tools.build:aapt2` è presente in `repo-B` e ha uno stato di. `Published` Non è presente in. `repo-A` Se un client richiede una risorsa di questa versione del pacchetto da`repo-A`, la risposta sarà 200 (OK) e la versione del pacchetto Maven `7.3.1` verrà mantenuta. `repo-A` Tuttavia, se lo stato della versione del pacchetto `7.3.1` in `repo-B` è `Archived` o`Disposed`, la risposta sarà 404 (Not Found) perché le risorse delle versioni del pacchetto in questi due stati non sono scaricabili.

Nota che l'impostazione del controllo di origine del pacchetto su `upstream=BLOCK` for `com.android.tools.build:aapt2` in `repo-A``repo-B`, `repo-C` impedirà il recupero di nuove risorse per tutte le versioni di quel pacchetto`repo-A`, indipendentemente dallo stato della versione del pacchetto.

Risoluzione dei problemi con Maven

Le seguenti informazioni potrebbero aiutarti a risolvere i problemi più comuni con Maven e. CodeArtifact

Disabilita i put paralleli per correggere l'errore 429: Troppe richieste

A partire dalla versione 3.9.0, Maven carica gli artefatti del pacchetto in parallelo (fino a 5 file alla volta). Ciò può causare la risposta occasionale con CodeArtifact un codice di risposta all'errore 429 (Too Many Requests). Se si verifica questo errore, è possibile disabilitare le uscite parallele per risolverlo.

Per disabilitare le uscite parallele, impostate la `aether.connector.basic.parallelPut` proprietà su `false` nel vostro profilo nel `settings.xml` file, come mostrato nell'esempio seguente:

```
<settings>
    <profiles>
        <profile>
            <id>default</id>
            <properties>
                <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
            </properties>
        </profile>
    </profiles>
<settings>
```

Per ulteriori informazioni, consulta le opzioni di configurazione di Artifact Resolver nella documentazione di Maven.

Usare CodeArtifact con npm

Questi argomenti descrivono come usare npm, il gestore di pacchetti Node.js, con. CodeArtifact

> **Note**
>
> CodeArtifact supporta `node v4.9.1` e versioni successive `npm v5.0.0` e successive.

Argomenti

- Configura e usa npm con CodeArtifact
- Configura e usa Yarn con CodeArtifact
- supporto per i comandi npm
- gestione dei tag npm
- Support per gestori di pacchetti compatibili con npm

Configura e usa npm con CodeArtifact

Dopo aver creato un repository in CodeArtifact, puoi usare il client npm per installare e pubblicare pacchetti. Il metodo consigliato per configurare npm con l'endpoint del repository e il token di autorizzazione consiste nell'utilizzare il comando. `aws codeartifact login` Puoi anche configurare npm manualmente.

Indice

- Configurazione di npm con il comando login
- Configurazione di npm senza utilizzare il comando login
- Esecuzione dei comandi npm
- Verifica dell'autenticazione e dell'autorizzazione di npm
- Tornare al registro npm predefinito
- Risoluzione dei problemi di installazioni lente con npm 8.x o versioni successive

Configurazione di npm con il comando login

Usa il `aws codeartifact login` comando per recuperare le credenziali da utilizzare con npm.

> **Note**
>
> Se accedi a un repository in un dominio di tua proprietà, non è necessario includerlo. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

> **Important**
>
> Se si utilizza npm 10.x o una versione successiva, è necessario utilizzare la AWS CLI versione 2.9.5 o successiva per eseguire correttamente il comando. `aws codeartifact login`

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

Questo comando apporta le seguenti modifiche al file ~/.npmrc:

- Aggiunge un token di autorizzazione dopo averlo recuperato utilizzando le tue credenziali. CodeArtifact AWS
- Imposta il registro npm sul repository specificato dall'opzione. `--repository`
- Per npm 6 e versioni precedenti: aggiunge `"always-auth=true"` in modo che il token di autorizzazione venga inviato per ogni comando npm.

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

Configurazione di npm senza utilizzare il comando login

Puoi configurare npm con il tuo CodeArtifact repository senza il `aws codeartifact login` comando aggiornando manualmente la configurazione di npm.

Per configurare npm senza usare il comando login

1. In una riga di comando, recupera un token di CodeArtifact autorizzazione e memorizzalo in una variabile di ambiente. npm utilizzerà questo token per autenticarsi con il tuo repository. CodeArtifact

   > **Note**
   >
   > Il comando seguente è per macchine macOS o Linux. Per informazioni sulla configurazione delle variabili di ambiente su un computer Windows, vedere. Passa un token di autenticazione utilizzando una variabile di ambiente

   ```
   CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

2. Ottieni l'endpoint del tuo CodeArtifact repository eseguendo il comando seguente. L'endpoint del tuo repository viene utilizzato per indirizzare npm al tuo repository per installare o pubblicare pacchetti.

   - Sostituiscilo *my_domain* con il tuo nome di dominio. CodeArtifact
   - Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
   - *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format npm
   ```

   L'URL seguente è un esempio di endpoint del repository.

   ```
   https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/
   ```

> **Important**
>
> L'URL del registro deve terminare con una barra (/). In caso contrario, non è possibile connettersi al repository.

3. Usa il `npm config set` comando per impostare il registro nel tuo CodeArtifact repository. Sostituisci l'URL con l'URL dell'endpoint del repository del passaggio precedente.

```
npm config set
 registry=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/
```

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

4. Usa il `npm config set` comando per aggiungere il tuo token di autorizzazione alla tua configurazione npm.

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:_authToken=$CODEARTIFACT_AUTH_TOKEN
```

Per npm 6 o versioni precedenti: per fare in modo che npm passi sempre il token di autenticazione a CodeArtifact, anche per `GET` le richieste, imposta la `always-auth` variabile di configurazione con. `npm config set`

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:always-auth=true
```

Esempio di file di configurazione npm () **`.npmrc`**

Di seguito è riportato un `.npmrc` file di esempio dopo aver seguito le istruzioni precedenti per impostare l'endpoint del CodeArtifact registro, aggiungere un token di autenticazione e configurare. `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
```

Esecuzione dei comandi npm

Dopo aver configurato il client npm, puoi eseguire i comandi npm. Supponendo che un pacchetto sia presente nel tuo repository o in uno dei suoi repository upstream, puoi installarlo con. `npm install` Ad esempio, utilizzate quanto segue per installare il pacchetto. `lodash`

```
npm install lodash
```

Usa il comando seguente per pubblicare un nuovo pacchetto npm in un CodeArtifact repository.

```
npm publish
```

Per informazioni su come creare pacchetti npm, vedere Creazione di moduli Node.js sul sito Web della documentazione di npm. Per un elenco dei comandi npm supportati da CodeArtifact, vedere npm Command Support.

Verifica dell'autenticazione e dell'autorizzazione di npm

L'invocazione del `npm ping` comando è un modo per verificare quanto segue:

- Hai configurato correttamente le tue credenziali in modo da poterti autenticare in un repository. CodeArtifact
- La configurazione dell'autorizzazione ti concede l'autorizzazione. `ReadFromRepository`

L'output di una chiamata riuscita di è `npm ping` simile al seguente.

```
$ npm -d ping
npm info it worked if it ends with ok
npm info using npm@6.4.1
npm info using node@v9.5.0
npm info attempt registry request try #1 at 4:30:59 PM
npm http request GET https://<domain>.d.codeartifact.us-west-2.amazonaws.com/npm/
shared/-/ping?write=true
```

```
npm http 200 https:///npm/shared/-/ping?write=true
Ping success: {}
npm timing npm Completed in 716ms
npm info ok
```

L'`-d`opzione fa sì che npm stampi informazioni di debug aggiuntive, incluso l'URL del repository. Queste informazioni semplificano la conferma che npm sia configurato per utilizzare il repository previsto.

Tornare al registro npm predefinito

La configurazione di npm con CodeArtifact imposta il registro npm sul repository specificato. CodeArtifact Puoi eseguire il seguente comando per riportare il registro npm al registro predefinito al termine della connessione a. CodeArtifact

```
npm config set registry https://registry.npmjs.com/
```

Risoluzione dei problemi di installazioni lente con npm 8.x o versioni successive

Esiste un problema noto nelle versioni 8.x e successive di npm in cui se viene effettuata una richiesta a un repository di pacchetti e il repository reindirizza il client ad Amazon S3 anziché trasmettere direttamente le risorse, il client npm può bloccarsi per diversi minuti per dipendenza.

Poiché i CodeArtifact repository sono progettati per reindirizzare sempre la richiesta ad Amazon S3, a volte si verifica questo problema, che causa lunghi tempi di compilazione a causa dei lunghi tempi di installazione di npm. Le istanze di questo comportamento si presenteranno come una barra di avanzamento che viene visualizzata per diversi minuti.

Per evitare questo problema, utilizzate i `progress=false` flag `--no-progress` o con i comandi npm cli, come illustrato nell'esempio seguente.

```
npm install lodash --no-progress
```

Configura e usa Yarn con CodeArtifact

Dopo aver creato un repository, puoi usare il client Yarn per gestire i pacchetti npm.

> **Note**
>
> `Yarn 1.X` legge e utilizza le informazioni dal file di configurazione npm (.npmrc), mentre non lo fa. `Yarn 2.X` La configurazione per `Yarn 2.X` deve essere definita nel file .yarnrc.yml.

Indice

- Configura Yarn 1.X con il comando aws codeartifact login
- Configura Yarn 2.X con il comando yarn config set

Configura Yarn 1.X con il comando **`aws codeartifact login`**

InfattiYarn 1.X, puoi configurare Yarn CodeArtifact usando il comando. `aws codeartifact login` Il `login` comando configurerà il file ~/.npmrc con le informazioni e le credenziali dell'endpoint del repository. CodeArtifact ConYarn 1.X, i `yarn` comandi utilizzano le informazioni di configurazione dal file ~/.npmrc.

Per configurare con il comando login **`Yarn 1.X`**

1. Se non l'hai già fatto, configura AWS le tue credenziali da utilizzare con AWS CLI, come descritto inGuida introduttiva con CodeArtifact.
2. Per eseguire correttamente il `aws codeartifact login` comando, è necessario installare npm. Vedi Download e installazione di Node.js e npm nella documentazione di npm per le istruzioni di installazione.
3. Usa il `aws codeartifact login` comando per recuperare le CodeArtifact credenziali e configurare il tuo file ~/.npmrc.

   - *`my_domain`* CodeArtifact Sostituiscilo con il tuo nome di dominio.
   - Sostituiscilo *`111122223333`* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
   - *`my_repo`*Sostituiscilo con il nome del tuo CodeArtifact repository.

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

Il `login` comando apporta le seguenti modifiche al file ~/.npmrc:

- Aggiunge un token di autorizzazione dopo averlo recuperato utilizzando le tue credenziali. CodeArtifact AWS
- Imposta il registro npm sul repository specificato dall'opzione. `--repository`
- Per npm 6 e versioni precedenti: aggiunge `"always-auth=true"` in modo che il token di autorizzazione venga inviato per ogni comando npm.

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

4. Per npm 7.X e 8.X, devi aggiungerlo `always-auth=true` al tuo file ~/.npmrc per usare Yarn.

   - Apri il tuo file ~/.npmrc in un editor di testo e aggiungilo in una nuova riga. `always-auth=true`

Puoi usare il `yarn config list` comando per verificare che Yarn stia usando la configurazione corretta. Dopo aver eseguito il comando, controlla i valori nella `info npm config` sezione. Il contenuto dovrebbe essere simile al frammento seguente.

```
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
}
```

Configura Yarn 2.X con il comando **`yarn config set`**

La procedura seguente descrive in dettaglio come configurare `Yarn 2.X` aggiornando la `.yarnrc.yml` configurazione dalla riga di comando con il `yarn config set` comando.

Per aggiornare la **`yarnrc.yml`** configurazione dalla riga di comando

1. Se non l'hai già fatto, configura AWS le tue credenziali da utilizzare con AWS CLI, come descritto inGuida introduttiva con CodeArtifact.
2. Usa il `aws codeartifact get-repository-endpoint` comando per ottenere l'endpoint del tuo CodeArtifact repository.

   - Sostituiscilo *`my_domain`* con il tuo CodeArtifact nome di dominio.
   - Sostituiscilo *`111122223333`* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
   - *`my_repo`*Sostituiscilo con il nome del tuo CodeArtifact repository.

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format npm
   ```

3. Aggiorna il `npmRegistryServer` valore nel file .yarnrc.yml con l'endpoint del repository.

   ```
   yarn config set npmRegistryServer
    "https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"
   ```

4. Recupera un token di CodeArtifact autorizzazione e memorizzalo in una variabile di ambiente.

   > **Note**
   >
   > Il comando seguente è per macchine macOS o Linux. Per informazioni sulla configurazione delle variabili di ambiente su un computer Windows, vedere. Passa un token di autenticazione utilizzando una variabile di ambiente

   - *`my_domain`*Sostituiscilo con il tuo nome di CodeArtifact dominio.
   - Sostituiscilo *`111122223333`* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
   - *`my_repo`*Sostituiscilo con il nome del tuo CodeArtifact repository.

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

5. Usa il `yarn config set` comando per aggiungere il token di CodeArtifact autenticazione al tuo file .yarnrc.yml. Sostituisci l'URL nel comando seguente con l'URL dell'endpoint del repository del passaggio 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
 "${CODEARTIFACT_AUTH_TOKEN}"
```

6. Utilizzate il `yarn config set` comando per impostare il valore di `npmAlwaysAuth` a. `true` Sostituisci l'URL nel comando seguente con l'URL dell'endpoint del repository del passaggio 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAlwaysAuth'
 "true"
```

Dopo la configurazione, il file di configurazione .yarnrc.yml dovrebbe avere contenuti simili al seguente frammento.

```
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/"
```

Puoi anche usare il comando per controllare i valori di and. `yarn config` `npmRegistries` `npmRegistryServer`

supporto per i comandi npm

Le sezioni seguenti riepilogano i comandi npm supportati, dai CodeArtifact repository, oltre ai comandi specifici che non sono supportati.

Indice

- Comandi supportati che interagiscono con un repository
- Comandi lato client supportati
- Comandi non supportati

Comandi supportati che interagiscono con un repository

Questa sezione elenca i comandi npm in cui il client npm effettua una o più richieste al registro con cui è stato configurato (ad esempio, con). `npm config set registry` È stato verificato che questi comandi funzionino correttamente quando vengono richiamati su un repository. CodeArtifact

Comando	Descrizione
bug	Cerca di indovinare la posizione dell'URL del bug tracker di un pacchetto, quindi prova ad aprirlo.
ci	Installa un progetto con una tabula rasa.
deprecare	Depreca una versione di un pacchetto.
dist-tag	Modifica i tag di distribuzione dei pacchetti.
documenti	Cerca di indovinare la posizione dell'URL della documentazione di un pacchetto, quindi prova ad aprirlo utilizzando il parametro `--browser` config.
dottore	Esegue una serie di controlli per garantire che l'installazione di npm abbia ciò di cui ha bisogno per gestire i JavaScript pacchetti.
installa	Installa un pacchetto.

Comando	Descrizione
install-ci-test	Installa un progetto con una lavagna pulita ed esegue dei test. Pseudonimo:. `npm cit` Questo comando viene eseguito e `npm ci` seguito immediatamente da un`npm test`.
install-test	Installa il pacchetto ed esegue i test. Viene eseguito e `npm install` seguito immediata mente da un`npm test`.
obsoleto	Controlla il registro configurato per vedere se alcuni pacchetti installati sono attualmente obsoleti.
ping	Esegue il ping del registro npm configurato o specificato e verifica l'autenticazione.
pubblica	Pubblica una versione del pacchetto nel registro.
update	Indovina la posizione dell'URL del repository di un pacchetto, quindi tenta di aprirlo utilizzando il parametro config. `--browser`
visualizzare	Visualizza i metadati del pacchetto. Può essere usato per stampare le proprietà dei metadati.

Comandi lato client supportati

Questi comandi non richiedono alcuna interazione diretta con un repository, quindi CodeArtifact non è necessario fare nulla per supportarli.

Comando	Descrizione
costruire	Costruisce un pacchetto.
cache	Manipola la cache dei pacchetti.

Comando	Descrizione
completamento	Abilita il completamento delle schede in tutti i comandi npm.
config	Aggiorna il contenuto dell'utente e dei `npmrc` file globali.
deduplicare	Cerca nell'albero dei pacchetti locale e tenta di semplificare la struttura spostando le dipendenz e più in alto nell'albero, dove possono essere condivise in modo più efficace da più pacchetti dipendenti.
modifica	Modifica un pacchetto installato. Seleziona una dipendenza nella directory di lavoro corrente e apre la cartella del pacchetto nell'editor predefinito.
esplora	Sfoglia un pacchetto installato. Genera una subshell nella directory del pacchetto installato specificato. Se viene specificato un comando, questo viene eseguito nella sottoshell, che poi termina immediatamente.
help	Ottiene aiuto su npm.
aiuto-ricerca	Cerca nella documentazione di aiuto di npm.
init	Crea un `package.json` file.
collegamento	Collega simbolicamente una cartella di pacchetto.
ls	Elenca i pacchetti installati.
pacchetto	Crea un archivio tar da un pacchetto.

Comando	Descrizione
prefisso	Visualizza il prefisso. Questa è la directory principale più vicina a contenere un `package.json` file, a meno che non `-g` venga specificata anche questa.
prugna	Rimuove i pacchetti che non sono elencati nell'elenco delle dipendenze del pacchetto principale.
ricostruire	Esegue il `npm build` comando sulle cartelle corrispondenti.
riavviare	Esegue gli script di arresto, riavvio e avvio di un pacchetto e i pre e post script associati.
root	Stampa la `node_modules` cartella effettiva fino allo standard.
esegui script	Esegue script di pacchetti arbitrari.
pellicola shrinkwrap	Blocca le versioni dipendenti per la pubblicaz ione.
disinstallare	Disinstalla un pacchetto.

Comandi non supportati

Questi comandi npm non sono supportati dai CodeArtifact repository.

Comando	Descrizione	Note
accesso	Imposta il livello di accesso sui pacchetti pubblicati.	CodeArtifact utilizza un modello di autorizzazione diverso dal repository pubblico npmjs.

Comando	Descrizione	Note
aggiungi utente	Aggiunge un account utente del registro	CodeArtifact utilizza un modello utente diverso dal repository pubblico npmjs.
audit	Esegue un controllo di sicurezza.	CodeArtifact attualmente non vende dati sulle vulnerabilità di sicurezza.
gancio	Gestisce gli hook di npm, inclusi l'aggiunta, la rimozione, l'elenco e l'aggiornamento.	CodeArtifact attualmente non supporta alcun tipo di meccanismo di notifica delle modifiche.
Login	Autentica un utente. Questo è un alias per `npm adduser`.	CodeArtifact utilizza un modello di autenticazione diverso dal repository pubblico npmjs. Per informazioni, consulta Autenticazione con npm.
Disconnessione	Esci dal registro.	CodeArtifact utilizza un modello di autenticazione diverso dal repository pubblico npmjs. Non è possibile disconnettersi da un CodeArtif act repository, ma i token di autenticazione scadono dopo la data di scadenza configura bile. La durata predefinita del token è di 12 ore.
proprietario	Gestisce i proprietari dei pacchetti.	CodeArtifact utilizza un modello di autorizzazioni diverso dal repository pubblico npmjs.

Comando	Descrizione	Note
profile	Modifica le impostazioni sul profilo del registro.	CodeArtifact utilizza un modello utente diverso dal repository pubblico npmjs.
cerca	Cerca nel registro i pacchetti che corrispondono ai termini di ricerca.	CodeArtifact supporta funzionalità di ricerca limitate con il comando list-packages.
stella	Contrassegna i tuoi pacchetti preferiti.	CodeArtifact attualmente non supporta alcun tipo di meccanismo dei preferiti.
stelle	Visualizza i pacchetti contrassegnati come preferiti.	CodeArtifact attualmente non supporta alcun tipo di meccanismo dei preferiti.
squadra	Gestisce i team organizzativi e le appartenenze ai team.	CodeArtifact utilizza un modello di appartenenza di utenti e gruppi diverso dal repository pubblico npmjs. Per informazioni, consulta Identità (utenti, gruppi e ruoli) nella Guida per l'utente IAM.
token	Gestisce i token di autentica zione.	CodeArtifact utilizza un modello diverso per ottenere i token di autenticazione. Per informazioni, consulta Autenticazione con npm.
annullare la pubblicazione	Rimuove un pacchetto dal registro.	CodeArtifact non supporta la rimozione di una versione del pacchetto da un repositor y utilizzando il client npm. È possibile utilizzare il comando delete-package-version.

Comando	Descrizione	Note
whoami	Visualizza il nome utente npm.	CodeArtifact utilizza un modello utente diverso dal repository pubblico npmjs.

gestione dei tag npm

I registri npm supportano i tag, che sono alias di stringa per le versioni dei pacchetti. È possibile utilizzare i tag per fornire un alias anziché i numeri di versione. Ad esempio, potresti avere un progetto con più flussi di sviluppo e utilizzare un tag diverso (ad esempio,,, `stable` `betadev`,`canary`) per ogni stream. Per ulteriori informazioni, consulta dist-tag sul sito Web di npm.

Per impostazione predefinita, npm utilizza il `latest` tag per identificare la versione corrente di un pacchetto. `npm install` *`pkg`*(senza `@`*`version`* o `@`*`tag`* specificatore) installa il tag più recente. In genere, i progetti utilizzano il tag più recente solo per le versioni di rilascio stabili. Altri tag vengono utilizzati per le versioni instabili o non definitive.

Modifica i tag con il client npm

I tre `npm dist-tag` comandi (`add`, `rm`, `ls`) funzionano in modo identico nei CodeArtifact repository come nel registro npm predefinito.

tag npm e API CopyPackageVersions

Quando si utilizza l'`CopyPackageVersions`API per copiare una versione del pacchetto npm, tutti i tag che costituiscono l'alias di quella versione vengono copiati nel repository di destinazione. Quando una versione che viene copiata ha un tag presente anche nella destinazione, l'operazione di copia imposta il valore del tag nel repository di destinazione in modo che corrisponda al valore nel repository di origine.

Ad esempio, supponiamo che sia il repository S che il repository D contengano una singola versione del `web-helper` pacchetto con il set di tag più recente, come mostrato in questa tabella.

Repository	Nome pacchetto	Tag del pacchetto
S	`web-helper`	più recente (alias per la versione 1.0.1)

Repository	Nome pacchetto	Tag del pacchetto
D	`web-helper`	più recente (alias per la versione 1.0.0)

`CopyPackageVersions`viene richiamato per copiare `web-helper` 1.0.1 da S a D. Al termine dell'operazione, il `latest` tag `web-helper` negli alias del repository D 1.0.1, non 1.0.0.

Se è necessario modificare i tag dopo la copia, utilizzare il `npm dist-tag` comando per modificare i tag direttamente nel repository di destinazione. Per ulteriori informazioni sull'`CopyPackageVersions`API, consulta Copiare i pacchetti tra i repository.

tag npm e repository upstream

Quando npm richiede i tag per un pacchetto e le versioni di quel pacchetto sono presenti anche in un repository upstream, CodeArtifact unisce i tag prima di restituirli al client. Ad esempio, un repository denominato R ha un repository upstream denominato U. La tabella seguente mostra i tag per un pacchetto denominato presente in entrambi `web-helper` i repository.

Repository	Nome pacchetto	Tag del pacchetto
R	`web-helper`	più recente (alias per la versione 1.0.0)
U	`web-helper`	alpha (alias per la versione 1.0.1)

In questo caso, quando il client npm recupera i tag per il *`web-helper`* pacchetto dal repository R, riceve sia i tag latest che quelli alpha. Le versioni a cui puntano i tag non cambieranno.

Quando lo stesso tag è presente sullo stesso pacchetto sia nel repository upstream che downstream, CodeArtifact utilizza il tag presente nel repository upstream. Ad esempio, supponiamo che i tag su webhelper siano stati modificati in modo da assomigliare ai seguenti.

Repository	Nome pacchetto	Tag del pacchetto
R	`web-helper`	più recente (alias per la versione 1.0.0)
U	`web-helper`	più recente (alias per la versione 1.0.1)

In questo caso, quando il client npm recupera i tag per il pacchetto web-helper dal repository R, l'ultimo tag utilizzerà l'alias della versione 1.0.1 perché è quello che c'è nel repository upstream. Ciò semplifica l'utilizzo di nuove versioni dei pacchetti in un repository upstream che non sono ancora presenti in un repository downstream mediante l'esecuzione. `npm update`

L'utilizzo del tag nel repository upstream può essere problematico quando si pubblicano nuove versioni di un pacchetto in un repository downstream. Ad esempio, supponiamo che l'ultimo tag sul pacchetto web-helper sia lo stesso sia in R che in U.

Repository	Nome pacchetto	Tag del pacchetto
R	`web-helper`	più recente (alias per la versione 1.0.1)
U	`web-helper`	più recente (alias per la versione 1.0.1)

Quando la versione 1.0.2 viene pubblicata su R, npm aggiorna l'ultimo tag alla 1.0.2.

Repository	Nome pacchetto	Tag del pacchetto
R	`web-helper`	più recente (alias per la versione 1.0.2)
U	`web-helper`	più recente (alias per la versione 1.0.1)

Tuttavia, il client npm non vede mai questo valore di tag perché il valore di latest in U è 1.0.1. L'`npm install`esecuzione sul repository R subito dopo la pubblicazione della 1.0.2 installa 1.0.1 anziché la versione appena pubblicata. Per installare la versione pubblicata più di recente, è necessario specificare la versione esatta del pacchetto, come segue.

```
npm install web-helper@1.0.2
```

Support per gestori di pacchetti compatibili con npm

Questi altri gestori di pacchetti sono compatibili CodeArtifact e funzionano con il formato di pacchetti npm e il protocollo npm wire:

- gestore di pacchetti pnpm. L'ultima versione confermata per funzionare CodeArtifact è la 3.3.4, rilasciata il 18 maggio 2019.
- Gestore di pacchetti Yarn. L'ultima versione confermata per funzionare CodeArtifact è la 1.21.1, rilasciata l'11 dicembre 2019.

> **Note**
>
> Ti consigliamo di usare Yarn 2.x con. CodeArtifact Yarn 1.x non dispone di nuovi tentativi HTTP, il che significa che è più suscettibile a guasti di servizio intermittenti che provocano codici di stato o errori di livello 500. Non è possibile configurare una strategia di riprova diversa per Yarn 1.x, ma questa è stata aggiunta in Yarn 2.x. Puoi usare Yarn 1.x, ma potrebbe essere necessario aggiungere nuovi tentativi di livello superiore negli script di compilazione. Ad esempio, esegui il comando yarn in un ciclo in modo che riprovi se il download dei pacchetti fallisce.

Utilizzo CodeArtifact con NuGet

Questi argomenti descrivono come utilizzare e pubblicare `NuGet` pacchetti utilizzando CodeArtifact.

> **Note**
>
> AWS CodeArtifact supporta solo la versione 4.8 e successive del NuGet file.exe.

Argomenti

- Uso CodeArtifact con Visual Studio
- Utilizzo CodeArtifact con la CLI nuget o dotnet
- NuGet normalizzazione del nome del pacchetto, della versione e del nome dell'asset
- NuGet compatibilità

Uso CodeArtifact con Visual Studio

Puoi utilizzare i pacchetti CodeArtifact direttamente da Visual Studio con il CodeArtifact Credential Provider. Il provider di credenziali semplifica la configurazione e l'autenticazione dei CodeArtifact repository in Visual Studio ed è disponibile in. AWS Toolkit for Visual Studio

> **Note**
>
> Non AWS Toolkit for Visual Studio è disponibile per Visual Studio per Mac.

Per configurare e utilizzare NuGet con gli strumenti CLI, vedere. Utilizzo CodeArtifact con la CLI nuget o dotnet

Argomenti

- Configurare Visual Studio con il CodeArtifact Credential Provider
- Utilizzare la console Visual Studio Package Manager

Configurare Visual Studio con il CodeArtifact Credential Provider

Il CodeArtifact Credential Provider semplifica la configurazione e l'autenticazione continua tra CodeArtifact Visual Studio e Visual Studio. CodeArtifact i token di autenticazione sono validi per un massimo di 12 ore. Per evitare di dover aggiornare manualmente il token mentre si lavora in Visual Studio, il provider di credenziali recupera periodicamente un nuovo token prima della scadenza del token corrente.

> **Important**
>
> Per utilizzare il provider di credenziali, assicurati che dal `nuget.config` file vengano cancellate tutte AWS CodeArtifact le credenziali esistenti che potrebbero essere state aggiunte manualmente o eseguendo la configurazione in precedenza. `aws codeartifact login` NuGet

Utilizzare CodeArtifact in Visual Studio con AWS Toolkit for Visual Studio

1. Installa il AWS Toolkit for Visual Studio utilizzando i seguenti passaggi. Il toolkit è compatibile con Visual Studio 2017 e 2019 utilizzando questi passaggi. AWS CodeArtifact non supporta Visual Studio 2015 e versioni precedenti.

    1. Il Toolkit for Visual Studio for Visual Studio 2017 e Visual Studio 2019 è distribuito in Visual Studio Marketplace. Puoi anche installare e aggiornare il toolkit all'interno di Visual Studio utilizzando Strumenti ≫ Estensioni e aggiornamenti (Visual Studio 2017) o Estensioni ≫ Gestisci estensioni (Visual Studio 2019).
    2. Dopo aver installato il toolkit, aprilo scegliendo AWS Explorer dal menu Visualizza.
2. Configura il Toolkit for Visual Studio con le AWS tue credenziali seguendo i passaggi descritti in AWS Fornitura delle credenziali nella AWS Toolkit for Visual Studio Guida per l'utente.
3. (Facoltativo) Imposta il AWS profilo con cui desideri utilizzare. CodeArtifact Se non è impostato, CodeArtifact utilizzerà il profilo predefinito. Per impostare il profilo, vai su Strumenti > NuGet Package Manager > Seleziona CodeArtifact AWS profilo.
4. Aggiungi il tuo CodeArtifact repository come origine del pacchetto in Visual Studio.

    1. Vai al tuo repository nella finestra AWS Explorer, fai clic con il pulsante destro del mouse e seleziona. `Copy NuGet Source Endpoint`
    2. Utilizzate il comando Strumenti > Opzioni e scorrete fino a NuGet Package Manager.

3. Seleziona il nodo Package Sources.
4. Seleziona +, modifica il nome e incolla l'endpoint dell'URL del repository copiato nel passaggio 3a nella casella Origine, quindi seleziona Aggiorna.
5. Seleziona la casella di controllo relativa alla fonte del pacchetto appena aggiunto per abilitarla.

> **Note**
>
> Ti consigliamo di aggiungere una connessione esterna NuGeta.org al tuo CodeArtifact repository e di disabilitare l'origine del pacchetto nuget.org in Visual Studio. Quando si utilizza una connessione esterna, tutti i pacchetti recuperati da NuGet.org verranno archiviati nel repository. CodeArtifact Se NuGet.org non è più disponibile, le dipendenze dell'applicazione saranno ancora disponibili per le build CI e lo sviluppo locale. Per ulteriori informazioni sulle connessioni esterne, vedere. Connect un CodeArtifact repository a un repository pubblico

5. Riavvia Visual Studio per rendere effettive le modifiche.

Dopo la configurazione, Visual Studio può utilizzare i pacchetti dal tuo CodeArtifact repository, da uno qualsiasi dei suoi repository upstream o da NuGet.org se hai aggiunto una connessione esterna. Per altre informazioni sulla navigazione e l'installazione NuGet dei pacchetti in Visual Studio, vedi Installare e gestire i pacchetti in Visual Studio utilizzando il NuGet Package Manager nella NuGet documentazione.

Utilizzare la console Visual Studio Package Manager

La console di Visual Studio Package Manager non utilizzerà la versione Visual Studio di CodeArtifact Credential Provider. Per utilizzarlo, dovrai configurare il provider di credenziali della riga di comando. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con la CLI nuget o dotnet.

Utilizzo CodeArtifact con la CLI nuget o dotnet

Puoi utilizzare strumenti CLI come `nuget` e `dotnet` da cui pubblicare e utilizzare pacchetti. CodeArtifact Questo documento fornisce informazioni sulla configurazione degli strumenti CLI e sul loro utilizzo per pubblicare o utilizzare pacchetti.

Argomenti

- Configurare la CLI nuget o dotnet

- Consuma pacchetti da NuGet CodeArtifact
- Pubblica NuGet pacchetti su CodeArtifact
- CodeArtifact NuGet Riferimento al provider di credenziali
- CodeArtifact NuGet Versioni di Credential Provider

Configurare la CLI nuget o dotnet

È possibile configurare la CLI nuget o dotnet con CodeArtifact NuGet il Credential Provider, con o manualmente. AWS CLI La configurazione NuGet con il provider di credenziali è altamente consigliata per una configurazione semplificata e un'autenticazione continua.

Metodo 1: Configurazione con il provider di credenziali CodeArtifact NuGet

Il CodeArtifact NuGet Credential Provider semplifica l'autenticazione e la configurazione degli strumenti CodeArtifact CLI NuGet . CodeArtifact i token di autenticazione sono validi per un massimo di 12 ore. Per evitare di dover aggiornare manualmente il token durante l'utilizzo della CLI nuget o dotnet, il provider di credenziali recupera periodicamente un nuovo token prima della scadenza del token corrente.

> **Important**
>
> Per utilizzare il provider di credenziali, assicurati che dal file vengano cancellate tutte AWS CodeArtifact le credenziali esistenti che potrebbero essere state aggiunte manualmente o eseguendo la `nuget.config` configurazione in precedenza. `aws codeartifact login` NuGet

Installa e configura il Credential Provider CodeArtifact NuGet

dotnet

1. Scarica l'ultima versione di AWS. CodeArtifact. NuGet. CredentialProvider strumento da NuGet .org con il seguente `dotnet` comando.

   ```
   dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
   ```

2. Utilizzate il `codeartifact-creds install` comando per copiare il provider di credenziali nella cartella NuGet plugins.

```
dotnet codeartifact-creds install
```

3. (Facoltativo): imposta il AWS profilo che desideri utilizzare con il provider di credenziali. Se non è impostato, il fornitore di credenziali utilizzerà il profilo predefinito. Per ulteriori informazioni sui AWS CLI profili, vedere Profili denominati.

```
dotnet codeartifact-creds configure set profile profile_name
```

nuget

Esegui i seguenti passaggi per utilizzare la NuGet CLI per installare il CodeArtifact NuGet Credential Provider da un bucket Amazon S3 e configurarlo. Il provider di credenziali utilizzerà il AWS CLI profilo predefinito; per ulteriori informazioni sui profili, consulta Profili denominati.

1. Scarica la versione più recente di CodeArtifact NuGet Credential Provider (codeartifact-nuget-credentialprovider.zip) da un bucket Amazon S3.

   Per visualizzare e scaricare le versioni precedenti, consulta. CodeArtifact NuGet Versioni di Credential Provider
2. Decomprimere il file.
3. Copia il file AWS. CodeArtifact. NuGetCredentialProvidercartella dalla cartella netfx `%user_profile%/.nuget/plugins/netfx/` a Windows o Linux o `~/.nuget/plugins/netfx` macOS.
4. Copia il file AWS. CodeArtifact. NuGetCredentialProvidercartella dalla cartella netcore `%user_profile%/.nuget/plugins/netcore/` a Windows o Linux o `~/.nuget/plugins/netcore` macOS.

Dopo aver creato un repository e configurato il provider di credenziali, puoi utilizzare gli strumenti o la `nuget` `dotnet` CLI per installare e pubblicare i pacchetti. Per ulteriori informazioni, consultare Consuma pacchetti da NuGet CodeArtifact e Pubblica NuGet pacchetti su CodeArtifact.

Metodo 2: configura nuget o dotnet con il comando login

Il `codeartifact login` comando contenuto in AWS CLI aggiunge un endpoint del repository e un token di autorizzazione al file di NuGet configurazione, permettendo a nuget o dotnet di connettersi al repository. CodeArtifact Ciò modificherà la NuGet configurazione a livello utente che si trova in

Windows e/o Mac/Linux. `%appdata%\NuGet\NuGet.Config` `~/.config/NuGet/NuGet.Config` `~/.nuget/NuGet/NuGet.Config` Per ulteriori informazioni sulle NuGet configurazioni, vedere Configurazioni comuni. NuGet

Configura nuget o dotnet con il comando **`login`**

1. Configura AWS le tue credenziali da utilizzare con AWS CLI, come descritto in. Guida introduttiva con CodeArtifact
2. Assicurati che lo strumento NuGet CLI (`nugetodotnet`) sia stato installato e configurato correttamente. Per istruzioni, consulta la documentazione di nuget o dotnet.
3. Utilizzate il CodeArtifact `login` comando per recuperare le credenziali da utilizzare con. NuGet

> **Note**
>
> Se accedi a un repository in un dominio di tua proprietà, non è necessario includerlo. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

dotnet

> **Important**
>
> Utenti Linux e macOS: poiché la crittografia non è supportata su piattaforme diverse da Windows, le credenziali recuperate verranno archiviate come testo semplice nel file di configurazione.

```
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
```

Il comando di login consentirà di:

- Recupera un token di autorizzazione CodeArtifact utilizzando le tue AWS credenziali.
- Aggiorna la NuGet configurazione a livello utente con una nuova voce per l'origine del NuGet pacchetto. Verrà chiamata la fonte che punta all'endpoint CodeArtifact del repository. *domain_name/repo_name*

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

Dopo aver creato un repository e configurato l'autenticazione, puoi utilizzare i client `nugetdotnet`, o `msbuild` CLI per installare e pubblicare i pacchetti. Per ulteriori informazioni, consultare Consuma pacchetti da NuGet CodeArtifact e Pubblica NuGet pacchetti su CodeArtifact.

Metodo 3: configura nuget o dotnet senza il comando login

Per la configurazione manuale, è necessario aggiungere un endpoint del repository e un token di autorizzazione al file di NuGet configurazione per consentire a nuget o dotnet di connettersi al repository. CodeArtifact

Configura manualmente nuget o dotnet per connetterti al tuo repository. CodeArtifact

1. Determina l'endpoint del tuo CodeArtifact repository usando il comando. `get-repository-endpoint` AWS CLI

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format nuget
   ```

   Output di esempio:

   ```
   {
      "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-
   west-2.amazonaws.com/nuget/my_repo/"
   }
   ```

2. Ottieni un token di autorizzazione per connetterti al tuo repository dal tuo gestore di pacchetti utilizzando il comando. `get-authorization-token` AWS CLI

   ```
   aws codeartifact get-authorization-token --domain my_domain
   ```

Output di esempio:

```
{
   "authorizationToken": "eyJ2I...viOw",
   "expiration": 1601616533.0
}
```

3. Crea l'URL completo dell'endpoint del repository aggiungendolo `/v3/index.json` all'URL restituito `get-repository-endpoint` nel passaggio 3.
4. Configura nuget o dotnet per utilizzare l'endpoint del repository dal passaggio 1 e il token di autorizzazione dal passaggio 2.

> **Note**
>
> L'URL di origine deve terminare `/v3/index.json` affinché nuget o dotnet si connettano correttamente a un repository. CodeArtifact

dotnet

Utenti Linux e macOS: poiché la crittografia non è supportata su piattaforme diverse da Windows, è necessario aggiungere il `--store-password-in-clear-text` flag al comando seguente. Tieni presente che in questo modo la password verrà memorizzata come testo semplice nel file di configurazione.

```
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...viOw --username aws
```

> **Note**
>
> Per aggiornare una fonte esistente, usa il `dotnet nuget update source` comando.

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
```

Output di esempio:

```
Package source with Name: domain_name/repo_name added successfully.
```

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

Consuma pacchetti da NuGet CodeArtifact

Dopo aver configurato NuGet con CodeArtifact, puoi utilizzare i NuGet pacchetti archiviati nel tuo CodeArtifact repository o in uno dei suoi repository upstream.

Per utilizzare una versione del pacchetto da un CodeArtifact repository o da uno dei suoi repository upstream con `nuget` o `dotnet`, esegui il comando seguente sostituendolo *packageName* con il nome del pacchetto che desideri consumare e *packageSourceName* con il nome sorgente del repository nel file di configurazione. CodeArtifact NuGet Se hai usato il `login` comando per configurare la NuGet configurazione, il nome sorgente è. *domain_name/repo_name*

> **Note**
>
> Quando viene richiesto un pacchetto, il NuGet client memorizza nella cache le versioni di quel pacchetto esistenti. A causa di questo comportamento, l'installazione potrebbe non riuscire per un pacchetto richiesto in precedenza prima che la versione desiderata diventasse disponibile. Per evitare questo errore e installare correttamente un pacchetto esistente, è possibile NuGet svuotare la cache prima dell'installazione con `nuget locals all --clear` o `dotnet nuget locals all --clear`, oppure evitare di utilizzare la cache durante `restore` i comandi `install` and specificando l'`-NoCache`opzione for `nuget` o l'`--no-cache`opzione per`dotnet`.

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

Per installare una versione specifica di un pacchetto

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

Per ulteriori informazioni, consulta Gestire i pacchetti utilizzando l'interfaccia della riga di comando nuget.exe o Installare e gestire i pacchetti utilizzando l'interfaccia della riga di comando dotnet nella documentazione Microsoft.

Consuma pacchetti NuGet da .org NuGet

È possibile utilizzare NuGet pacchetti da NuGet.org tramite un CodeArtifact repository configurando il repository con una connessione esterna a .org. NuGet I pacchetti consumati da NuGet.org vengono importati e archiviati nel tuo repository. CodeArtifact Per ulteriori informazioni sull'aggiunta di connessioni esterne, consulta. Connect un CodeArtifact repository a un repository pubblico

Pubblica NuGet pacchetti su CodeArtifact

Dopo aver configurato NuGet con CodeArtifact, puoi utilizzare `nuget` o pubblicare le versioni dei pacchetti nei `dotnet` repository. CodeArtifact

Per inviare una versione del pacchetto a un CodeArtifact repository, esegui il comando seguente con il percorso completo del `.nupkg` file e il nome sorgente del CodeArtifact repository nel file di configurazione NuGet . Se hai usato il `login` comando per configurare la NuGet configurazione, il nome della fonte è. `domain_name/repo_name`

> **Note**
>
> Puoi creare un NuGet pacchetto se non ne hai uno da pubblicare. Per ulteriori informazioni, consulta Flusso di lavoro per la creazione di Package nella documentazione Microsoft.

dotnet

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

nuget

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg -Source packageSourceName
```

CodeArtifact NuGet Riferimento al provider di credenziali

Il CodeArtifact NuGet Credential Provider semplifica la configurazione e l'autenticazione NuGet con i tuoi repository. CodeArtifact

CodeArtifact NuGet Comandi Credential Provider

Questa sezione include l'elenco dei comandi per il CodeArtifact NuGet Credential Provider. Questi comandi devono avere il prefisso `dotnet codeartifact-creds` come nell'esempio seguente.

```
dotnet codeartifact-creds command
```

- `configure set profile profile`: configura il provider di credenziali per utilizzare il profilo fornito. AWS
- `configure unset profile`: rimuove il profilo configurato, se impostato.
- `install`: copia il provider di credenziali nella `plugins` cartella.
- `install --profile profile`: copia il provider di credenziali nella `plugins` cartella e lo configura per utilizzare il profilo fornito. AWS
- `uninstall`: disinstalla il provider di credenziali. Ciò non rimuove le modifiche al file di configurazione.
- `uninstall --delete-configuration`: disinstalla il provider di credenziali e rimuove tutte le modifiche al file di configurazione.

CodeArtifact NuGet Registri di Credential Provider

Per abilitare la registrazione per il CodeArtifact NuGet Credential Provider, è necessario impostare il file di registro nel proprio ambiente. I log del provider di credenziali contengono informazioni di debug utili come:

- Il profilo utilizzato per effettuare le connessioni AWS
- Eventuali errori di autenticazione
- Se l'endpoint fornito non è un URL CodeArtifact

Imposta il file di registro del CodeArtifact NuGet Credential Provider

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

Dopo aver impostato il file di registro, qualsiasi `codeartifact-creds` comando aggiungerà il relativo output di registro al contenuto di quel file.

CodeArtifact NuGet Versioni di Credential Provider

La tabella seguente contiene informazioni sulla cronologia delle versioni e collegamenti per il download per il CodeArtifact NuGet Credential Provider.

Versione	Modifiche	Data di pubblicazione	Link per il download (S3)
1.0.2 (più recente)	Dipendenze aggiornate	26/06/2024	Scarica v1.0.2
1.0.1	È stato aggiunto il supporto per i profili net5, net6 e SSO	03/05/2022	Scarica v1.0.1
1.0.0	Versione iniziale di CodeArtifact NuGet Credential Provider	20/11/2020	Scarica v1.0.0

NuGet normalizzazione del nome del pacchetto, della versione e del nome dell'asset

CodeArtifact normalizza i nomi e le versioni dei pacchetti e delle risorse prima di archiviarli, il che significa che i nomi o le versioni dei pacchetti CodeArtifact possono essere diversi da quelli forniti al momento della pubblicazione del pacchetto o della risorsa.

Normalizzazione dei nomi dei pacchetti: CodeArtifact normalizza i nomi dei NuGet pacchetti convertendo tutte le lettere in minuscolo.

Normalizzazione della versione del NuGet pacchetto: CodeArtifact normalizza le versioni dei pacchetti utilizzando lo stesso schema di. NuGet Le seguenti informazioni sono tratte dai numeri di versione normalizzati riportati nella documentazione. NuGet

- Gli zeri iniziali vengono rimossi dai numeri di versione:
  - `1.00`viene trattato come `1.0`
  - `1.01.1`viene trattato come `1.1.1`
  - `1.00.0.1`viene trattato come `1.0.0.1`
- Verrà omesso uno zero nella quarta parte del numero di versione:
  - `1.0.0.0`viene trattato come `1.0.0`
  - `1.0.01.0`viene trattato come `1.0.1`
- SemVer i metadati della build 2.0.0 vengono rimossi:
  - `1.0.7+r3456`viene trattato come `1.0.7`

Normalizzazione del nome dell'asset del pacchetto: CodeArtifact costruisce il nome dell'asset del NuGet pacchetto dal nome del pacchetto e dalla versione del pacchetto normalizzati.

Il nome del pacchetto e il nome della versione non normalizzati possono essere utilizzati con le richieste API e CLI perché CodeArtifact esegue la normalizzazione degli input del nome del pacchetto e della versione per tali richieste. Ad esempio, gli input di `--package Newtonsoft.JSON` e `--version 12.0.03.0` verrebbero normalizzati e restituirebbero un pacchetto con un nome e una versione di pacchetto normalizzati di. `newtonsoft.json` `12.0.3`

È necessario utilizzare il nome dell'asset del pacchetto normalizzato nelle richieste API e CLI CodeArtifact poiché non esegue la normalizzazione dell'input. `--asset`

È necessario utilizzare nomi e versioni normalizzati in. ARNs

Per trovare il nome normalizzato di un pacchetto, usa il `aws codeartifact list-packages` comando. Per ulteriori informazioni, consulta Elenca i nomi dei pacchetti.

Per trovare il nome non normalizzato di un pacchetto, usa il comando. `aws codeartifact describe-package-version` Il nome non normalizzato del pacchetto viene restituito nel campo. `displayName` Per ulteriori informazioni, consulta Visualizza e aggiorna i dettagli e le dipendenze della versione del pacchetto.

NuGet compatibilità

Questa guida contiene informazioni sulla CodeArtifact compatibilità con diversi NuGet strumenti e versioni.

Argomenti

- NuGet Compatibilità generale
- NuGet supporto da riga di comando

NuGet Compatibilità generale

AWS CodeArtifact supporta NuGet 4.8 e versioni successive.

AWS CodeArtifact supporta solo la versione 3 del NuGet protocollo HTTP. Ciò significa che alcuni comandi CLI che si basano sulla versione 2 del protocollo non sono supportati. Per maggiori informazioni, consulta la sezione supporto per i comandi nuget.exe.

AWS CodeArtifact non supporta PowerShellGet 2.x.

NuGet supporto da riga di comando

AWS CodeArtifact supporta gli NuGet strumenti CLI (`nuget.exe`) e.NET Core (`dotnet`).

supporto per i comandi nuget.exe

Poiché supporta CodeArtifact solo il protocollo HTTP NuGet di V3 of, i seguenti comandi non funzioneranno se utilizzati contro CodeArtifact risorse:

- `list`: Il `nuget list` comando visualizza un elenco di pacchetti da una determinata fonte. Per ottenere un elenco di pacchetti in un CodeArtifact repository, puoi usare il Elenca i nomi dei pacchetti comando dalla AWS CLI.

Usare CodeArtifact con Python

Questi argomenti descrivono come usare`pip`, il gestore di pacchetti Python e `twine` l'utilità di pubblicazione dei pacchetti Python con. CodeArtifact

Argomenti

- Configura e usa pip con CodeArtifact
- Configura e usa twine con CodeArtifact
- Normalizzazione dei nomi dei pacchetti in Python
- Compatibilità con Python
- Richiesta di pacchetti Python da upstream e connessioni esterne

Configura e usa pip con CodeArtifact

pip è l'installatore di pacchetti per i pacchetti Python. Per usare pip per installare i pacchetti Python dal CodeArtifact tuo repository, devi prima configurare il client pip con le informazioni e le credenziali del CodeArtifact tuo repository.

pip può essere usato solo per installare pacchetti Python. Per pubblicare pacchetti Python, puoi usare twine. Per ulteriori informazioni, consulta Configura e usa twine con CodeArtifact.

Configura pip con il comando **`login`**

Innanzitutto, configura AWS le tue credenziali da utilizzare con AWS CLI, come descritto in. Guida introduttiva con CodeArtifact Quindi, utilizzate il CodeArtifact `login` comando per recuperare le credenziali e `pip` configurarle con esse.

> **Note**
>
> Se accedi a un repository in un dominio di tua proprietà, non è necessario includerlo. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

Per configurare pip, esegui il comando seguente.

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

`login`recupera un token di autorizzazione CodeArtifact utilizzando le tue AWS credenziali. Il `login` comando verrà configurato `pip` per essere utilizzato con CodeArtifact modificando `~/.config/pip/pip.conf` per impostare il `index-url` repository specificato dall'opzione. `--repository`

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

Configura pip senza il comando login

Se non puoi usare il `login` comando per configurare`pip`, puoi usare`pip config`.

1. Utilizzate il AWS CLI per recuperare un nuovo token di autorizzazione.

   > **Note**
   >
   > Se accedi a un repository in un dominio di tua proprietà, non è necessario includere il. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

   ```
   CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

2. Utilizzare `pip config` per impostare l'URL e le credenziali del CodeArtifact registro. Il comando seguente aggiornerà solo il file di configurazione dell'ambiente corrente. Per aggiornare il file di configurazione a livello di sistema, sostituirlo `site` con. `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**
   >
   > Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

> **Important**
>
> L'URL del registro deve terminare con una barra (/). In caso contrario, non è possibile connettersi al repository.

Esempio di file di configurazione pip

Di seguito è riportato un esempio di `pip.conf` file dopo aver impostato l'URL e le credenziali del CodeArtifact registro.

```
[global]
index-url = https://aws:eyJ2ZX...@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/pypi/my_repo/simple/
```

Esegui pip

Per eseguire `pip` i comandi, è necessario configurare `pip` con CodeArtifact. Per ulteriori informazioni, consulta la seguente documentazione:

1. Segui i passaggi indicati nella Configurazione con AWS CodeArtifact sezione per configurare l' AWS account, gli strumenti e le autorizzazioni.
2. Configura `twine` seguendo la procedura riportata di seguito. Configura e usa twine con CodeArtifact

Supponendo che un pacchetto sia presente nel tuo repository o in uno dei suoi repository upstream, puoi installarlo con. `pip install` Ad esempio, utilizzate il comando seguente per installare il pacchetto. `requests`

```
pip install requests
```

Usa l'`-i`opzione per tornare temporaneamente all'installazione dei pacchetti da https://pypi.org anziché dal tuo CodeArtifact repository.

```
pip install -i https://pypi.org/simple requests
```

Configura e usa twine con CodeArtifact

twine è un'utilità di pubblicazione di pacchetti per pacchetti Python. Per usare twine per pubblicare pacchetti Python nel CodeArtifact tuo repository, devi prima configurare twine con le informazioni e le credenziali del CodeArtifact tuo repository.

twine può essere usato solo per pubblicare pacchetti Python. Per installare i pacchetti Python, puoi usare pip. Per ulteriori informazioni, consulta Configura e usa pip con CodeArtifact.

Configura twine con il comando **`login`**

Innanzitutto, configura AWS le tue credenziali da utilizzare con AWS CLI, come descritto in. Guida introduttiva con CodeArtifact Quindi, usa il CodeArtifact `login` comando per recuperare le credenziali e configurare twine con esse.

> **Note**
>
> Se accedi a un repository in un dominio di tua proprietà, non è necessario includerlo. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

Per configurare twine, esegui il seguente comando.

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

`login`recupera un token di autorizzazione CodeArtifact utilizzando le tue AWS credenziali. Il `login` comando configura twine da utilizzare con modificandolo per `~/.pypirc` aggiungere il repository specificato CodeArtifact dall'opzione con credenziali. `--repository`

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

Configura twine senza il comando **`login`**

Se non è possibile utilizzare il `login` comando per configurare twine, è possibile utilizzare le variabili di `~/.pypirc` file o di ambiente. Per utilizzare il `~/.pypirc` file, aggiungete le seguenti voci. La password deve essere un token di autenticazione acquisito dall'`get-authorization-token`API.

```
[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**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

Per utilizzare le variabili di ambiente, procedi come segue.

> **Note**
>
> Se accedi a un repository in un dominio di tua proprietà, non è necessario includere il `--domain-owner`. Per ulteriori informazioni, consulta Domini con più account.

```
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`
```

Esegui twine

Prima di usare twine per pubblicare le risorse del pacchetto Python, devi prima CodeArtifact configurare i permessi e le risorse.

1. Segui i passaggi indicati nella Configurazione con AWS CodeArtifact sezione per configurare l' AWS account, gli strumenti e le autorizzazioni.
2. Configura twine seguendo i passaggi in Configura twine con il comando `login` o. Configura twine senza il comando `login`

Dopo aver configurato twine, puoi eseguire `twine` i comandi. Usa il seguente comando per pubblicare gli asset del pacchetto Python.

```
twine upload --repository codeartifact mypackage-1.0.tgz
```

Per informazioni su come creare e impacchettare la tua applicazione Python, vedi Generating Distribution Archives sul sito Web Python Packaging Authority.

Normalizzazione dei nomi dei pacchetti in Python

CodeArtifact normalizza i nomi dei pacchetti prima di memorizzarli, il che significa che i nomi dei pacchetti CodeArtifact possono essere diversi dal nome fornito quando il pacchetto è stato pubblicato.

Per i pacchetti Python, quando si esegue la normalizzazione il nome del pacchetto è in minuscolo e tutte le istanze dei caratteri vengono sostituite con un `.` singolo `-` carattere. `_` `-` Quindi i nomi dei pacchetti e sono normalizzati `pigeon_cli` e `pigeon.cli` memorizzati come. `pigeon-cli` Il nome non normalizzato può essere utilizzato da pip e twine, ma il nome normalizzato deve essere utilizzato nelle richieste CodeArtifact CLI o API (come) e in. `list-package-versions` ARNs Per ulteriori informazioni sulla normalizzazione dei nomi dei pacchetti Python, vedere PEP 503 nella documentazione di Python.

Compatibilità con Python

CodeArtifact non supporta PyPI o. `XML-RPC` `JSON` APIs

CodeArtifact supporta PyPI `Legacy` APIs, ad eccezione dell'`simple`API. Sebbene CodeArtifact non supporti l'endpoint `/simple/` API, supporta l'`/simple/<project>/`endpoint.

Per ulteriori informazioni, vedere quanto segue nel repository della Python Packaging Authority. GitHub

- API XML-RPC
- API JSON
- API legacy

supporto per i comandi pip

Le sezioni seguenti riepilogano i comandi pip supportati dai CodeArtifact repository, oltre ai comandi specifici che non sono supportati.

Argomenti

- Comandi supportati che interagiscono con un repository
- Comandi lato client supportati

Comandi supportati che interagiscono con un repository

Questa sezione elenca `pip` i comandi in cui il `pip` client effettua una o più richieste al registro con cui è stato configurato. È stato verificato che questi comandi funzionino correttamente quando vengono richiamati su un CodeArtifact repository.

Comando	Descrizione
installare	Installa pacchetti.
scaricare	Scarica i pacchetti.

CodeArtifact non implementa`pip search`. Se hai configurato `pip` con un CodeArtifact repository, l'esecuzione `pip search` cercherà e mostrerà i pacchetti da PyPI.

Comandi lato client supportati

Questi comandi non richiedono alcuna interazione diretta con un repository, quindi CodeArtifact non è necessario fare nulla per supportarli.

Comando	Descrizione
disinstallare	Disinstalla i pacchetti.
congelare	Visualizza i pacchetti installati nel formato dei requisiti.
elenco	Elenca i pacchetti installati.

Comando	Descrizione
show	Mostra informazioni sui pacchetti installati.
controlla	Verifica che i pacchetti installati abbiano dipendenze compatibili.
config	Gestisci la configurazione locale e globale.
ruota	Costruisci ruote in base alle tue esigenze.
cancelletto	Calcola gli hash degli archivi dei pacchetti.
completamento	Aiuta a completare i comandi.
debug	Mostra informazioni utili per il debug.
Aiuto	Mostra l'aiuto per i comandi.

Richiesta di pacchetti Python da upstream e connessioni esterne

Quando si importa una versione del pacchetto Python da pypi.org CodeArtifact , importerà tutte le risorse in quella versione del pacchetto. Sebbene la maggior parte dei pacchetti Python contenga un numero limitato di risorse, alcuni ne contengono più di 100, in genere per supportare più architetture hardware e interpreti Python.

È normale che nuove risorse vengano pubblicate su pypi.org per una versione del pacchetto esistente. Ad esempio, alcuni progetti pubblicano nuove risorse quando vengono rilasciate nuove versioni di Python. Quando un pacchetto Python viene installato da CodeArtifact con`pip install`, le versioni del pacchetto conservate nel CodeArtifact repository vengono aggiornate per riflettere l'ultimo set di risorse di pypi.org.

Allo stesso modo, se sono disponibili nuove risorse per una versione del pacchetto in un CodeArtifact repository upstream che non sono presenti nel CodeArtifact repository corrente, verranno conservate nel repository corrente quando vengono eseguite. `pip install`

Versioni del pacchetto modificate

Alcune versioni dei pacchetti in pypi.org sono contrassegnate come cancellate, il che comunica all'installatore del pacchetto (come pip) che la versione non deve essere installata a meno che non sia l'unica che corrisponde a un identificatore di versione (usando uno o). == === Vedi PEP_592 per maggiori informazioni.

Se una versione del pacchetto in CodeArtifact è stata originariamente recuperata da una connessione esterna a pypi.org, quando installi la versione del pacchetto da un CodeArtifact repository, CodeArtifact assicura che i metadati rimossi aggiornati della versione del pacchetto vengano recuperati da pypi.org.

Come sapere se una versione del pacchetto è stata rimossa

Per verificare se la versione di un pacchetto è stata inserita CodeArtifact, puoi provare a installarla con. `pip install packageName===packageVersion` Se la versione del pacchetto viene rimossa, riceverai un messaggio di avviso simile al seguente:

```
WARNING: The candidate selected for download or install is a yanked version
```

Per verificare se una versione del pacchetto è stata rimossa in pypi.org, puoi visitare l'elenco pypi.org della versione del pacchetto all'indirizzo. `https://pypi.org/project/packageName/packageVersion/`

Impostare lo status di «annullato» sui pacchetti privati

CodeArtifact non supporta l'impostazione di metadati rimossi per i pacchetti pubblicati direttamente negli archivi. CodeArtifact

Perché CodeArtifact non sta recuperando gli ultimi metadati o risorse eliminati per una versione del pacchetto?

Normalmente, CodeArtifact assicura che quando una versione del pacchetto Python viene recuperata da un CodeArtifact repository, i metadati rimossi abbiano il valore più recente su up-to-date pypi.org. Inoltre, l'elenco delle risorse nella versione del pacchetto viene aggiornato anche con il set più recente su pypi.org e su qualsiasi repository upstream. CodeArtifact Questo è vero sia che tu stia installando la versione del pacchetto per la prima volta e la CodeArtifact importi da pypi.org nel tuo CodeArtifact repository, sia se hai già installato il pacchetto. Tuttavia, ci sono casi in cui il client di

gestione dei pacchetti, come pip, non estrae gli ultimi metadati estratti da pypi.org o dagli archivi upstream. Invece, CodeArtifact restituirà i dati già archiviati nel tuo repository. Questa sezione descrive i tre modi in cui ciò può avvenire:

Configurazione upstream: se la connessione esterna a pypi.org viene rimossa dal repository o dai relativi flussi upstream utilizzando disassociate-external-connection, i metadati rimossi non verranno più aggiornati da pypi.org. Allo stesso modo, se rimuovi un repository upstream, le risorse dal repository rimosso e dagli upstream del repository rimosso non saranno più disponibili per il repository corrente. Lo stesso vale se si utilizzano i controlli di origine dei CodeArtifact pacchetti per impedire che vengano recuperate nuove versioni di un pacchetto specifico: l'impostazione `upstream=BLOCK` bloccherà l'aggiornamento dei metadati eliminati.

Stato della versione del pacchetto: se imposti lo stato di una versione del pacchetto su un valore diverso da `Published` o`Unlisted`, i metadati e le risorse della versione del pacchetto eliminati non verranno aggiornati. Allo stesso modo, se state recuperando una versione specifica del pacchetto (ad esempio`torch 2.0.1`) e la stessa versione del pacchetto è presente in un repository upstream con uno stato diverso `Published` oppure`Unlisted`, ciò bloccherà anche la propagazione dei metadati e delle risorse sottratti dal repository upstream al repository corrente. Questo perché gli altri stati delle versioni del pacchetto indicano che le versioni non sono più pensate per essere utilizzate in nessun repository.

Pubblicazione diretta: se pubblichi una versione specifica del pacchetto direttamente in un CodeArtifact repository, ciò impedirà l'eliminazione dei metadati e dell'aggiornamento delle risorse per la versione del pacchetto dai suoi repository originali e da pypi.org. Ad esempio, supponiamo che tu scarichi una risorsa dalla versione del pacchetto`torch 2.0.1`, ad esempio usando un browser web`torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`, e poi la pubblichi nel tuo repository usando twine as. CodeArtifact `torch 2.0.1` CodeArtifact traccia che la versione del pacchetto è entrata nel dominio mediante pubblicazione diretta nel tuo repository, non da una connessione esterna a pypi.org o da un repository upstream. In questo caso, CodeArtifact non mantiene i metadati eliminati sincronizzati con i repository upstream o pypi.org. Lo stesso vale se pubblicate `torch 2.0.1` in un repository upstream: la presenza della versione del pacchetto bloccherà la propagazione dei metadati e delle risorse rimossi nei repository più in basso nel grafico a monte.

Usare CodeArtifact con Ruby

Questi argomenti descrivono come utilizzare gli strumenti RubyGems e Bundler per installare e pubblicare le gemme CodeArtifact di Ruby.

> **Note**
>
> CodeArtifact consiglia Ruby 3.3 o versioni successive e non funziona con Ruby 2.6 o versioni precedenti.

Argomenti

- Configura e usa RubyGems e Bundler con CodeArtifact
- RubyGems supporto ai comandi
- compatibilità con Bundler

Configura e usa RubyGems e Bundler con CodeArtifact

Dopo aver creato un repository in CodeArtifact, puoi usare RubyGems (`gem`) e Bundler (`bundle`) per installare e pubblicare gemme. Questo argomento descrive come configurare i gestori di pacchetti per l'autenticazione e l'utilizzo di un repository. CodeArtifact

Configura RubyGems (**`gem`**) e Bundler () con **`bundle`** CodeArtifact

Per utilizzare RubyGems (`gem`) o Bundler (`bundle`) per pubblicare gemme o da cui consumare gemme AWS CodeArtifact, devi prima configurarli con le informazioni del tuo CodeArtifact repository, comprese le credenziali per accedervi. Segui i passaggi di una delle seguenti procedure per configurare gli strumenti e la `bundle` CLI con le informazioni `gem` e le credenziali dell'endpoint CodeArtifact del repository.

Configura RubyGems e raggruppa utilizzando le istruzioni della console

Puoi usare le istruzioni di configurazione nella console per connettere i tuoi gestori di pacchetti Ruby al tuo CodeArtifact repository. Le istruzioni della console forniscono comandi personalizzati che è possibile eseguire per configurare i gestori di pacchetti senza dover trovare e inserire le informazioni. CodeArtifact

1. Apri la AWS CodeArtifact console all'indirizzo https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Nel pannello di navigazione, scegli Repository, quindi scegli il repository che desideri utilizzare per installare o inviare le gemme Ruby.
3. Scegli Visualizza le istruzioni di connessione.
4. Scegli il tuo sistema operativo.
5. Scegli il client di gestione pacchetti Ruby che desideri configurare con il tuo CodeArtifact repository.
6. Segui le istruzioni generate per configurare il client di gestione dei pacchetti da cui installare o pubblicare le gemme di Ruby nel repository.

Configura e raggruppa manualmente RubyGems

Se non puoi o non vuoi usare le istruzioni di configurazione della console, puoi usare le seguenti istruzioni per connetterti manualmente ai tuoi gestori di pacchetti Ruby al tuo CodeArtifact repository.

1. In una riga di comando, usa il seguente comando per recuperare un token di CodeArtifact autorizzazione e memorizzarlo in una variabile di ambiente.

   - *my_domain*Sostituiscilo con il tuo nome di CodeArtifact dominio.
   - Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.

   macOS and Linux

   ```
   export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

   Windows

   - Windows (utilizzando la shell dei comandi predefinita):

```
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. Per pubblicare le gemme di Ruby nel tuo repository, usa il seguente comando per recuperare l'endpoint del tuo CodeArtifact repository e memorizzarlo nella variabile di ambiente. `RUBYGEMS_HOST` La `gem` CLI utilizza questa variabile di ambiente per determinare dove vengono pubblicate le gemme.

> **Note**
>
> In alternativa, invece di utilizzare la variabile di `RUBYGEMS_HOST` ambiente, è possibile fornire all'endpoint del repository l'`--host`opzione quando si utilizza il comando. `gem push`

- Sostituiscilo *my_domain* con il tuo CodeArtifact nome di dominio.
- Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
- *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

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

I seguenti comandi recuperano l'endpoint del repository, tagliano il finale e lo memorizzano in una / variabile di ambiente.

- Windows (utilizzando la shell dei comandi predefinita):

```
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("/")
```

L'URL seguente è un esempio di endpoint del repository:

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/
```

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

3. Per pubblicare le gemme di Ruby nel tuo repository, devi autenticarti su CodeArtifact with RubyGems modificando il file per includere il tuo token di autenticazione. `~/.gem/credentials` Crea una `~/.gem/` directory e un `~/.gem/credentials` file se la directory o il file non esiste.

   macOS and Linux

```
echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/
credentials
```

Windows

- Windows (utilizzando la shell di comando predefinita):

```
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. `gem`Per installare le gemme Ruby dal tuo repository, devi aggiungere le informazioni sull'endpoint del repository e il token di autenticazione al tuo file. `.gemrc` Puoi aggiungerlo al file globale () `~/.gemrc` o al file di progetto. `.gemrc` Le CodeArtifact informazioni da aggiungere a sono una combinazione dell'`.gemrc`endpoint del repository e del token di autenticazione. È formattata come segue:

```
https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/
```

- Per il token di autenticazione, è possibile utilizzare la variabile di `CODEARTIFACT_AUTH_TOKEN` ambiente impostata in un passaggio precedente.
- Per recuperare l'endpoint del repository, puoi leggere il valore della variabile di `RUBYGEMS_HOST` ambiente impostata in precedenza oppure puoi usare il seguente `get-repository-endpoint` comando, sostituendo i valori se necessario:

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint
 --output text
```

Dopo aver ottenuto l'endpoint, utilizzate un editor di testo per aggiungerlo `aws:${CODEARTIFACT_AUTH_TOKEN}@` nella posizione appropriata. Una volta creati l'endpoint del repository e la stringa del token di autenticazione, aggiungili alla `:sources:` sezione del `.gemrc` file con il `echo` comando seguente:

> **Warning**
>
> CodeArtifact non supporta l'aggiunta di repository come sorgenti utilizzando il comando. `gem sources -add` È necessario aggiungere la fonte direttamente al file.

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 (utilizzando la shell dei comandi predefinita):

```
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. Per utilizzare Bundler, devi configurare Bundler con l'URL dell'endpoint del repository e il token di autenticazione eseguendo il seguente comando: `bundle config`

   macOS and Linux

   ```
   bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
   ```

   Windows

   - Windows (utilizzando la shell dei comandi predefinita):

```
bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
```

- Windows PowerShell:

```
bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
```

Ora che hai configurato RubyGems (`gem`) e Bundler (`bundle`) con il tuo CodeArtifact repository, puoi usarli per pubblicare e utilizzare le gemme di Ruby da e verso di esso.

Installazione di Ruby gems da CodeArtifact

Usa le seguenti procedure per installare le gemme Ruby da un CodeArtifact repository con gli strumenti o `gem` CLI`bundle`.

Installa Ruby gems con **`gem`**

Puoi usare la CLI RubyGems (`gem`) per installare rapidamente una versione specifica di una gemma Ruby dal tuo repository. CodeArtifact

Per installare Ruby gems da un repository con CodeArtifact **`gem`**

1. In caso contrario, segui i passaggi indicati Configura RubyGems (`gem`) e Bundler () con `bundle` CodeArtifact per configurare la `gem` CLI per utilizzare il tuo CodeArtifact repository con le credenziali appropriate.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Usa il seguente comando per installare Ruby gems da: CodeArtifact

   ```
   gem install my_ruby_gem --version 1.0.0
   ```

Installa Ruby gems con **`bundle`**

Puoi usare la CLI Bundler (`bundle`) per installare le gemme Ruby configurate nel tuo. `Gemfile`

Per installare Ruby gems da un repository con CodeArtifact **`bundle`**

1. In caso contrario, segui i passaggi indicati Configura RubyGems (`gem`) e Bundler () con `bundle` CodeArtifact per configurare la `bundle` CLI per utilizzare il tuo CodeArtifact repository con le credenziali appropriate.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Aggiungi l'URL CodeArtifact dell'endpoint del tuo repository al tuo annuncio per `source` installare `Gemfile` le gemme Ruby configurate dal tuo CodeArtifact repository e dai suoi upstream.

   ```
   source "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   ruby/my_repo/"

   gem 'my_ruby_gem'
   ```

3. Usa il seguente comando per installare le gemme Ruby come specificato nel tuo: `Gemfile`

   ```
   bundle install
   ```

Pubblicazione delle gemme di Ruby su CodeArtifact

Usa la seguente procedura per pubblicare le gemme di Ruby in un CodeArtifact repository utilizzando la CLI. `gem`

1. In caso contrario, segui i passaggi indicati Configura RubyGems (`gem`) e Bundler () con `bundle` CodeArtifact per configurare la `gem` CLI per utilizzare il tuo CodeArtifact repository con le credenziali appropriate.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Usa il seguente comando per pubblicare le gemme di Ruby in un CodeArtifact repository. Nota che se non hai impostato la variabile di `RUBYGEMS_HOST` ambiente, devi fornire l'endpoint del CodeArtifact repository nell'opzione. `--host`

```
gem push --key codeartifact_api_key my_ruby_gem-0.0.1.gem
```

RubyGems supporto ai comandi

CodeArtifact supporta i `gem push` comandi `gem install` and. CodeArtifact non supporta i seguenti `gem` comandi:

- `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`

compatibilità con Bundler

Questa guida contiene informazioni sulla compatibilità CodeArtifact di Bundler.

Compatibilità con Bundler

AWS CodeArtifact consiglia Bundler 2.4.11 o versione successiva. Se riscontri problemi con l'installazione, aggiorna la CLI di Bundler alla versione più recente.

Supporto della versione Bundler

Nelle versioni di Bundler precedenti alla 2.4.11, esiste un limite di 500 dipendenze che possono essere definite nel Gemfile prima che Bundler decida di interrogare l'indice completo,. `specs.4.8.gz` Poiché non CodeArtifact supporta l'indice completo, non è possibile specificare più di 500 dipendenze quando si utilizzano versioni di Bundler inferiori alla 2.4.11. CodeArtifact

Per definire più di 500 dipendenze nel tuo Gemfile con CodeArtifact, aggiorna Bundler alla versione 2.4.11 o successiva.

Supporto operativo Bundler

CodeArtifactil supporto per RubyGems non include il Bundler Compact Index APIs (l'`/versions`API non è supportata). CodeArtifact supporta solo l'API Dependencies.

Inoltre, CodeArtifact non supporta le varie specifiche APIs, come. `specs.4.8.gz`

Utilizzo CodeArtifact con Swift

Questi argomenti descrivono come utilizzare Swift Package Manager CodeArtifact per installare e pubblicare pacchetti Swift.

> **Note**
>
> CodeArtifact supporta Swift 5.8 e versioni successive e Xcode 14.3 e versioni successive. CodeArtifact consiglia Swift 5.9 e versioni successive e Xcode 15 e versioni successive.

Argomenti

- Configura Swift Package Manager con CodeArtifact
- Consumo e pubblicazione di pacchetti Swift
- Normalizzazione del nome e dello spazio dei nomi del pacchetto Swift
- Risoluzione dei problemi Swift

Configura Swift Package Manager con CodeArtifact

Per utilizzare Swift Package Manager per pubblicare o utilizzare pacchetti da cui utilizzare AWS CodeArtifact, devi prima configurare le credenziali per accedere al tuo CodeArtifact repository. Il metodo consigliato per configurare la CLI di Swift Package Manager con CodeArtifact le credenziali e l'endpoint del repository consiste nell'utilizzare il comando. `aws codeartifact login` Puoi anche configurare Swift Package Manager manualmente.

Configura Swift con il comando login

Usa il `aws codeartifact login` comando per configurare Swift Package Manager con CodeArtifact.

> **Note**
>
> Per utilizzare il comando login, è necessario Swift 5.8 o successivo e Swift 5.9 o successivo è consigliato.

Il `aws codeartifact login` comando eseguirà le seguenti operazioni:

1. Recupera un token di autenticazione CodeArtifact e memorizzalo nel tuo ambiente. La modalità di archiviazione delle credenziali dipende dal sistema operativo dell'ambiente:

   a. macOS: viene creata una voce nell'applicazione macOS Keychain.

   b. Linux e Windows: viene creata una voce nel file. `~/.netrc`

   In tutti i sistemi operativi, se esiste un'immissione di credenziali, questo comando sostituisce tale voce con un nuovo token.

2. Recupera l'URL CodeArtifact dell'endpoint del repository e aggiungilo al file di configurazione Swift. Il comando aggiunge l'URL dell'endpoint del repository al file di configurazione a livello di progetto che si trova in. `/path/to/project/.swiftpm/configuration/registries.json`

> **Note**
>
> Il `aws codeartifact login` comando richiama `swift package-registry` i comandi che devono essere eseguiti dalla directory che contiene il `Package.swift` file. Per questo motivo, `aws codeartifact login` il comando deve essere eseguito dall'interno del progetto Swift.

Per configurare Swift con il comando login

1. Vai alla directory del progetto Swift che contiene il file del `Package.swift` tuo progetto.
2. Eseguire il seguente comando `aws codeartifact login`.

   Se accedi a un repository in un dominio di tua proprietà, non è necessario includerlo. `--domain-owner` Per ulteriori informazioni, consulta Domini con più account.

```
aws codeartifact login --tool swift --domain my_domain \
--domain-owner 111122223333 --repository my_repo \
[--namespace my_namespace]
```

L'`--namespace`opzione configura l'applicazione in modo che utilizzi solo i pacchetti del tuo CodeArtifact repository se si trovano nello spazio dei nomi designato. CodeArtifact i namespace sono

sinonimo di ambiti e vengono utilizzati per organizzare il codice in gruppi logici e per prevenire le collisioni di nomi che possono verificarsi quando la base di codice include più librerie.

Il periodo di autorizzazione predefinito dopo la chiamata `login` è di 12 ore e `login` deve essere chiamato per aggiornare periodicamente il token. Per ulteriori informazioni sul token di autorizzazione creato con il `login` comando, vedereToken creati con il comando `login`.

Configura Swift senza il comando di login

Sebbene sia consigliabile configurare Swift con il `aws codeartifact login` comando, è possibile configurare Swift Package Manager anche senza il comando login aggiornando manualmente la configurazione di Swift Package Manager.

Nella procedura seguente, utilizzerai il AWS CLI per effettuare le seguenti operazioni:

1. Recupera un token di autenticazione CodeArtifact e memorizzalo nel tuo ambiente. La modalità di archiviazione delle credenziali dipende dal sistema operativo dell'ambiente:
   a. macOS: viene creata una voce nell'applicazione macOS Keychain.
   b. Linux e Windows: viene creata una voce nel file. `~/.netrc`
2. Recupera l'URL dell'endpoint CodeArtifact del tuo repository.
3. Nel file di `~/.swiftpm/configuration/registries.json` configurazione, aggiungi una voce con l'URL dell'endpoint del repository e il tipo di autenticazione.

Per configurare Swift senza il comando login

1. In una riga di comando, usa il seguente comando per recuperare un token di CodeArtifact autorizzazione e memorizzarlo in una variabile di ambiente.
   - *`my_domain`*Sostituiscilo con il tuo nome di CodeArtifact dominio.
   - Sostituiscilo *`111122223333`* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.

macOS and Linux

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

Windows

- Windows (utilizzando la shell dei comandi predefinita):

```
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. Ottieni l'endpoint del tuo CodeArtifact repository eseguendo il comando seguente. L'endpoint del tuo repository viene utilizzato per indirizzare Swift Package Manager al tuo repository per consumare o pubblicare pacchetti.

   - Sostituiscilo *my_domain* con il tuo nome di dominio. CodeArtifact
   - Sostituiscilo *111122223333* con l'ID dell' AWS account del proprietario del dominio. Se accedi a un repository in un dominio di tua proprietà, non è necessario `--domain-owner` includerlo. Per ulteriori informazioni, consulta Domini con più account.
   - *my_repo*Sostituiscilo con il nome del tuo CodeArtifact repository.

   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 (utilizzando la shell di comando predefinita):

```
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
```

L'URL seguente è un esempio di endpoint del repository.

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
swift/my_repo/
```

> **Note**
>
> Per utilizzare un endpoint dualstack, usa l'endpoint. `codeartifact.region.on.aws`

> **Important**
>
> È necessario aggiungere alla fine `login` dell'endpoint URL del repository quando viene utilizzato per configurare Swift Package Manager. Questa operazione viene eseguita automaticamente nei comandi di questa procedura.

3. Con questi due valori memorizzati nelle variabili di ambiente, passali a Swift usando il `swift package-registry login` comando seguente:

   macOS and Linux

```
swift package-registry login ${CODEARTIFACT_REPO}login --token
 ${CODEARTIFACT_AUTH_TOKEN}
```

Windows

- Windows (utilizzando la shell dei comandi predefinita):

```
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. Successivamente, aggiorna il registro dei pacchetti utilizzato dall'applicazione in modo che qualsiasi dipendenza venga estratta dal CodeArtifact repository. Questo comando deve essere eseguito nella directory del progetto in cui state cercando di risolvere la dipendenza del pacchetto:

   macOS and Linux

   ```
   $ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
   ```

   Windows

   - Windows (utilizzando la shell dei comandi predefinita):

   ```
   $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
   ```

   - Windows PowerShell:

   ```
   $ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
   ```

   L'`--scope`opzione configura l'applicazione in modo che utilizzi solo i pacchetti del CodeArtifact repository se rientrano nell'ambito designato. Gli ambiti sono sinonimo di CodeArtifact namespace e vengono utilizzati per organizzare il codice in gruppi logici e per prevenire le collisioni di nomi che possono verificarsi quando la base di codice include più librerie.

5. È possibile verificare che la configurazione sia stata impostata correttamente visualizzando il contenuto del `.swiftpm/configuration/registries.json` file a livello di progetto eseguendo il comando seguente nella directory del progetto:

```
$ 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
}
```

Ora che hai configurato Swift Package Manager con il tuo CodeArtifact repository, puoi usarlo per pubblicare e utilizzare pacchetti Swift da e verso di esso. Per ulteriori informazioni, consulta Consumo e pubblicazione di pacchetti Swift.

Consumo e pubblicazione di pacchetti Swift

Consumo di pacchetti Swift da CodeArtifact

Usa la seguente procedura per consumare pacchetti Swift da un AWS CodeArtifact repository.

Per utilizzare pacchetti Swift da un repository CodeArtifact

1. In caso contrario, segui i passaggi indicati Configura Swift Package Manager con CodeArtifact per configurare Swift Package Manager per utilizzare il tuo CodeArtifact repository con le credenziali corrette.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Modifica il `Package.swift` file nella cartella del progetto dell'applicazione per aggiornare le dipendenze dei pacchetti che verranno utilizzate dal progetto.

   a. Se il `Package.swift` file non contiene una `dependencies` sezione, aggiungine una.

b. Nella `dependencies` sezione del `Package.swift` file, aggiungi il pacchetto che desideri utilizzare aggiungendo il relativo identificatore del pacchetto. L'identificatore del pacchetto è costituito dall'ambito e dal nome del pacchetto separati da un punto. Per un esempio, vedi il frammento di codice che segue un passaggio successivo.

> **Tip**
>
> Per trovare l'identificatore del pacchetto, puoi usare la console. CodeArtifact Trova la versione specifica del pacchetto che desideri utilizzare e consulta le istruzioni rapide di installazione nella pagina della versione del pacchetto.

c. Se il `Package.swift` file non contiene una `targets` sezione, aggiungine una.

d. Nella `targets` sezione, aggiungi gli obiettivi che dovranno utilizzare la dipendenza.

Il frammento seguente è un frammento di esempio che mostra le sezioni configurate `dependencies` e `targets` le sezioni in un file: `Package.swift`

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
    ],
...
```

3. Ora che tutto è configurato, usate il seguente comando per scaricare le dipendenze del pacchetto da. CodeArtifact

```
swift package resolve
```

Consumo di pacchetti Swift da Xcode CodeArtifact

Usa la seguente procedura per consumare pacchetti Swift da un CodeArtifact repository in Xcode.

Per utilizzare pacchetti Swift da un repository in Xcode CodeArtifact

1. In caso contrario, segui i passaggi indicati Configura Swift Package Manager con CodeArtifact per configurare Swift Package Manager per utilizzare il tuo CodeArtifact repository con le credenziali corrette.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Aggiungi i pacchetti come dipendenza nel tuo progetto in Xcode.

   a. Scegli File > Aggiungi pacchetti.

   b. Cerca il tuo pacchetto utilizzando la barra di ricerca. La tua ricerca deve essere inserita nel modulo`package_scope.package_name`.

   c. Una volta trovato, scegli il pacchetto e scegli Aggiungi pacchetto.

   d. Una volta verificato il pacchetto, scegli i prodotti del pacchetto che desideri aggiungere come dipendenza e scegli Aggiungi pacchetto.

Se riscontri problemi nell'utilizzo del tuo CodeArtifact repository con Xcode, consulta i problemi più comuni e Risoluzione dei problemi Swift le possibili correzioni.

Pubblicazione di pacchetti Swift su CodeArtifact

CodeArtifact consiglia Swift 5.9 o versione successiva e utilizza il `swift package-registry publish` comando per pubblicare pacchetti Swift. Se stai usando una versione precedente, devi usare un comando Curl per pubblicare i pacchetti Swift su. CodeArtifact

Pubblicazione di CodeArtifact pacchetti con il comando **`swift package-registry publish`**

Usa la seguente procedura con Swift 5.9 o versione successiva per pubblicare pacchetti Swift in un CodeArtifact repository con Swift Package Manager.

1. In caso contrario, segui i passaggi indicati Configura Swift Package Manager con CodeArtifact per configurare Swift Package Manager per utilizzare il tuo CodeArtifact repository con le credenziali corrette.

> **Note**
>
> Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla sua creazione.

2. Vai alla directory del progetto Swift che contiene il `Package.swift` file del tuo pacchetto.
3. Esegui il `swift package-registry publish` comando seguente per pubblicare il pacchetto. Il comando crea un archivio sorgente del pacchetto e lo pubblica nel tuo CodeArtifact repository.

```
swift package-registry publish packageScope.packageName packageVersion
```

Per esempio:

```
swift package-registry publish myScope.myPackage 1.0.0
```

4. È possibile confermare che il pacchetto è stato pubblicato ed esiste nel repository effettuando il check-in nella console o utilizzando il `aws codeartifact list-packages` comando seguente:

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

È possibile elencare la versione singola del pacchetto utilizzando il `aws codeartifact list-package-versions` comando seguente:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Pubblicazione di CodeArtifact pacchetti con Curl

Sebbene sia consigliabile utilizzare il `swift package-registry publish` comando fornito con Swift 5.9 o versioni successive, puoi anche usare Curl per pubblicare pacchetti Swift su. CodeArtifact

Usa la seguente procedura per pubblicare pacchetti Swift in un repository con Curl. AWS CodeArtifact

1. In caso contrario, crea e aggiorna le `CODEARTIFACT_AUTH_TOKEN` variabili e di `CODEARTIFACT_REPO` ambiente seguendo i passaggi riportati di seguito. Configura Swift Package Manager con CodeArtifact

   > **Note**
   >
   > Il token di autorizzazione è valido per 12 ore. Dovrai aggiornare la variabile di `CODEARTIFACT_AUTH_TOKEN` ambiente con nuove credenziali se sono trascorse 12 ore dalla sua creazione.

2. Innanzitutto, se non hai creato un pacchetto Swift, puoi farlo eseguendo i seguenti comandi:

   ```
   mkdir testDir && cd testDir
   swift package init
   git init .
   swift package archive-source
   ```

3. Usa il seguente comando Curl per pubblicare il tuo pacchetto Swift su: CodeArtifact

   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. Puoi confermare che il pacchetto è stato pubblicato ed esiste nel repository effettuando il check-in nella console o utilizzando il `aws codeartifact list-packages` comando seguente:

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

   È possibile elencare la versione singola del pacchetto utilizzando il `aws codeartifact list-package-versions` comando seguente:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Recupero di pacchetti Swift GitHub e ripubblicazione su CodeArtifact

Usa la seguente procedura per recuperare un pacchetto Swift da GitHub e ripubblicarlo in un repository. CodeArtifact

Per recuperare un pacchetto Swift da e ripubblicarlo in GitHub CodeArtifact

1. In caso contrario, segui i passaggi indicati Configura Swift Package Manager con CodeArtifact per configurare Swift Package Manager per utilizzare il tuo CodeArtifact repository con le credenziali corrette.

   > **Note**
   >
   > Il token di autorizzazione generato è valido per 12 ore. Dovrai crearne uno nuovo se sono trascorse 12 ore dalla creazione del token.

2. Clona il repository git del pacchetto Swift che desideri recuperare e ripubblicare usando il seguente comando. `git clone` Per informazioni sulla clonazione dei repository, consulta Clonazione di un GitHub repository nei Documenti. GitHub

   ```
   git clone repoURL
   ```

3. Passa al repository che hai appena clonato:

   ```
   cd repoName
   ```

4. Crea il pacchetto e pubblicalo su. CodeArtifact

   a. Consigliato: se utilizzi Swift 5.9 o versioni successive, puoi usare il seguente `swift package-registry publish` comando per creare il pacchetto e pubblicarlo nel tuo repository configurato CodeArtifact .

      ```
      swift package-registry publish packageScope.packageName versionNumber
      ```

      Per esempio:

```
swift package-registry publish myScope.myPackage 1.0.0
```

b. Se utilizzi una versione di Swift precedente alla 5.9, devi usare il `swift archive-source` comando per creare il pacchetto e poi usare un comando Curl per pubblicarlo.

   i. Se non hai configurato le variabili `CODEARTIFACT_AUTH_TOKEN` e di `CODEARTIFACT_REPO` ambiente, o sono passate più di 12 ore da quando l'hai fatto, segui i passaggi indicati. Configura Swift senza il comando di login

   ii. Crea il pacchetto Swift usando il `swift package archive-source` comando:

   ```
   swift package archive-source
   ```

   In caso di successo, lo vedrai `Created` *package_name*`.zip` nella riga di comando.

   iii. Usa il seguente comando Curl per pubblicare il pacchetto Swift su: CodeArtifact

   macOS and Linux

   ```
   curl -X PUT  --user "aws:$CODEARTIFACT_AUTH_TOKEN" \
   -H "Accept: application/vnd.swift.registry.v1+json" \
   -F source-archive="@package_name.zip" \
   "${CODEARTIFACT_REPO}my_scope/package_name/packageVersion"
   ```

   Windows

   ```
   curl -X PUT  --user "aws:%CODEARTIFACT_AUTH_TOKEN%" \
   -H "Accept: application/vnd.swift.registry.v1+json" \
   -F source-archive="@package_name.zip" \
   "%CODEARTIFACT_REPO%my_scope/package_name/packageVersion"
   ```

5. Puoi confermare che il pacchetto è stato pubblicato ed esiste nel repository effettuando il check-in nella console o utilizzando il `aws codeartifact list-packages` comando seguente:

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

È possibile elencare la versione singola del pacchetto utilizzando il `aws codeartifact list-package-versions` comando seguente:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
```

```
--format swift --namespace my_scope --package package_name
```

Normalizzazione del nome e dello spazio dei nomi del pacchetto Swift

CodeArtifact normalizza i nomi e gli spazi dei nomi dei pacchetti prima di memorizzarli, il che significa che i nomi contenuti CodeArtifact possono essere diversi da quelli forniti quando il pacchetto è stato pubblicato.

Normalizzazione del nome del pacchetto e dello spazio dei nomi: CodeArtifact normalizza i nomi e gli spazi dei nomi dei pacchetti Swift convertendo tutte le lettere in minuscolo.

Normalizzazione della versione del pacchetto: CodeArtifact non normalizza le versioni dei pacchetti Swift. Nota che supporta CodeArtifact solo i modelli di versione Semantic Versioning 2.0, per ulteriori informazioni sul Semantic Versioning, vedi Semantic Versioning 2.0.0.

Il nome e lo spazio dei nomi del pacchetto non normalizzati possono essere utilizzati con le richieste API e CLI perché CodeArtifact eseguono la normalizzazione degli input per tali richieste. Ad esempio, gli input di `--package myPackage` e `--namespace myScope` verrebbero normalizzati e restituirebbero un pacchetto con un nome di pacchetto e uno spazio dei nomi normalizzati di. `mypackage` `myscope`

È necessario utilizzare nomi normalizzati, ad esempio nelle ARNs politiche IAM.

Per trovare il nome normalizzato di un pacchetto, usa il `aws codeartifact list-packages` comando. Per ulteriori informazioni, consulta Elenca i nomi dei pacchetti.

Risoluzione dei problemi Swift

Le seguenti informazioni potrebbero aiutarti a risolvere i problemi più comuni relativi a Swift e. CodeArtifact

Ricevo un errore 401 in Xcode anche dopo aver configurato Swift Package Manager

Problema: quando cerchi di aggiungere un pacchetto dal tuo CodeArtifact repository come dipendenza al tuo progetto Swift in Xcode, ricevi un errore 401 non autorizzato anche dopo aver seguito le istruzioni per connettere Swift a. CodeArtifact

Possibili correzioni: ciò può essere causato da un problema con l'applicazione macOS Keychain, in cui sono archiviate CodeArtifact le credenziali. Per risolvere questo problema, ti consigliamo di aprire l'applicazione Keychain ed eliminare tutte le CodeArtifact voci e di configurare nuovamente Swift Package Manager con il tuo CodeArtifact repository seguendo le istruzioni riportate in. Configura Swift Package Manager con CodeArtifact

Xcode si blocca sulla macchina CI a causa della richiesta di password del portachiavi

Problema: quando si tenta di estrarre pacchetti Swift da CodeArtifact una build Xcode su un server di integrazione continua (CI), ad esempio con GitHub Actions, l'autenticazione con CodeArtifact può bloccarsi e alla fine fallire con un messaggio di errore simile al seguente:

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

Possibili correzioni: ciò è causato dal fatto che le credenziali non vengono salvate nel portachiavi sulle macchine CI e Xcode supporta solo le credenziali salvate in Keychain. Per risolvere questo problema, consigliamo di creare manualmente la voce del portachiavi utilizzando i seguenti passaggi:

1. Prepara il portachiavi.

```
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. Ottieni un token di CodeArtifact autenticazione e l'endpoint del tuo repository.

```
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. Crea manualmente la voce Keychain.

```
SERVER=$(echo $CODEARTIFACT_REPO | sed  's/https:\/\///g' | sed 's/.com.*$/.com/g')
AUTHORIZATION=(-T /usr/bin/security -T /usr/bin/codesign -T /usr/bin/xcodebuild -
T /usr/bin/swift \
               -T /Applications/Xcode-15.2.app/Contents/Developer/usr/bin/
xcodebuild)

security delete-internet-password -a token -s $SERVER -r htps "${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}"
```

Per ulteriori informazioni su questo errore e sulla soluzione, vedere https://github.com/apple/swift-package-manager/issues/7236.

Utilizzo CodeArtifact con pacchetti generici

Questi argomenti mostrano come utilizzare e pubblicare pacchetti generici utilizzando AWS CodeArtifact.

Argomenti

- Panoramica dei pacchetti generici
- Comandi supportati per pacchetti generici
- Pubblicazione e utilizzo di pacchetti generici

Panoramica dei pacchetti generici

Utilizzando il formato del `generic` pacchetto, è possibile caricare qualsiasi tipo di file per creare un pacchetto in un CodeArtifact repository. I pacchetti generici non sono associati a nessun linguaggio di programmazione, tipo di file o ecosistema di gestione dei pacchetti specifico. Questo può essere utile per archiviare e controllare le versioni di artefatti di build arbitrari, come programmi di installazione di applicazioni, modelli di apprendimento automatico, file di configurazione e altro.

Un pacchetto generico è costituito da un nome di pacchetto, uno spazio dei nomi, una versione e uno o più asset (o file). I pacchetti generici possono esistere insieme a pacchetti di altri formati in un unico CodeArtifact repository.

È possibile utilizzare l'SDK AWS CLI o l'SDK per lavorare con pacchetti generici. Per un elenco completo dei AWS CLI comandi che funzionano con i pacchetti generici, consultaComandi supportati per pacchetti generici.

Vincoli relativi ai pacchetti generici

- Non vengono mai recuperati dai repository originali. Possono essere ottenuti solo dal repository in cui sono stati pubblicati.
- Non possono dichiarare dipendenze da restituire ListPackageVersionDependencieso visualizzare in. Console di gestione AWS
- Possono archiviare file README e LICENSE, ma non vengono interpretati da. CodeArtifact Le informazioni contenute in questi file non vengono restituite da GetPackageVersionReadmeo DescribePackageVersione non vengono visualizzate in. Console di gestione AWS

- Come tutti i pacchetti CodeArtifact, esistono dei limiti alla dimensione delle risorse e al numero di risorse per pacchetto. Per ulteriori informazioni sui limiti e le quote in CodeArtifact, vedereQuote in AWS CodeArtifact.
- I nomi delle risorse che contengono devono rispettare queste regole:
  - I nomi delle risorse possono utilizzare lettere e numeri Unicode. In particolare, sono consentite queste categorie di caratteri Unicode: Lowercase Letter (`Ll`), Modifier Letter (`Lm`), Other Letter (), Titlecase Letter (`Lo`), Uppercase Letter (`Lt`), Letter Number (`Lu`) e Decimal Number (). `Nl` `Nd`
  - I seguenti caratteri speciali sono consentiti: `~!@^&()-_+[]{};,.`
  - `.`Le risorse non possono essere denominate o `..`
  - Gli spazi sono l'unico carattere di spazio bianco consentito. I nomi delle risorse non possono iniziare o terminare con uno spazio o includere spazi consecutivi.

Comandi supportati per pacchetti generici

È possibile utilizzare AWS CLI o SDK per lavorare con pacchetti generici. I seguenti CodeArtifact comandi funzionano con pacchetti generici:

- copy-package-versions(vedereCopia i pacchetti tra i repository)
- delete-package (vedi) Eliminazione di un pacchetto (AWS CLI)
- delete-package-versions(vedi) Eliminazione di una versione del pacchetto ()AWS CLI
- descrivi il pacchetto
- describe-package-version(vedi) Visualizza e aggiorna i dettagli e le dipendenze della versione del pacchetto
- dispose-package-versions(vedereEliminazione delle versioni dei pacchetti)
- get-package-version-asset(vedereScarica gli asset della versione del pacchetto)
- list-package-version-assets(vedereElenca le risorse della versione del pacchetto)
- list-package-versions(vedereElenca le versioni dei pacchetti)
- list-packages (vedi) Elenca i nomi dei pacchetti
- publish-package-version(vedi) Pubblicazione di un pacchetto generico
- put-package-origin-configuration(vedereModifica dei controlli di origine dei pacchetti)

> **Note**
>
> È possibile utilizzare l'impostazione di controllo dell'`publish`origine per consentire o bloccare la pubblicazione di un nome di pacchetto generico in un repository. Tuttavia, l'`upstream`impostazione non si applica ai pacchetti generici perché non possono essere recuperati da un repository upstream.

- update-package-versions-status(vedi) Aggiornamento dello stato della versione del pacchetto

Pubblicazione e utilizzo di pacchetti generici

Per pubblicare una versione generica del pacchetto e le relative risorse, utilizzate il `publish-package-version` comando. È possibile elencare le risorse di un pacchetto generico utilizzando il `list-package-version-asset` comando e scaricarle utilizzando`get-package-version-asset`. L'argomento seguente contiene step-by-step istruzioni per pubblicare pacchetti generici o scaricare risorse di pacchetti generici utilizzando questi comandi.

Pubblicazione di un pacchetto generico

Un pacchetto generico è costituito da un nome di pacchetto, uno spazio dei nomi, una versione e uno o più asset (o file). Questo argomento dimostra come pubblicare un pacchetto denominato`my-package`, con lo spazio dei nomi`my-ns`, la versione `1.0.0` e contenente una risorsa denominata. `asset.tar.gz`

Prerequisiti:

- Imposta e configura il AWS Command Line Interface with CodeArtifact (vedi) Configurazione con AWS CodeArtifact
- Avere un CodeArtifact dominio e un repository (vediIniziare a usare il AWS CLI)

Per pubblicare un pacchetto generico

1. Utilizzate il comando seguente per generare l' SHA256 hash per ogni file che desiderate caricare in una versione del pacchetto e inserite il valore in una variabile di ambiente. Questo valore viene utilizzato come controllo di integrità per verificare che il contenuto del file non sia cambiato dopo l'invio originale.

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. Chiama `publish-package-version` per caricare la risorsa e creare una nuova versione del pacchetto.

> **Note**
>
> Se il pacchetto contiene più di una risorsa, puoi chiamare `publish-package-version` una volta per ogni risorsa da caricare. `--unfinished`Includete l'argomento per ogni chiamata a`publish-package-version`, tranne quando caricate la risorsa finale. L'omissione `--unfinished` imposterà lo stato della versione del pacchetto su e impedirà il caricamento di risorse aggiuntive. `Published`
> In alternativa, includi `--unfinished` per ogni chiamata a`publish-package-version`, quindi imposta lo stato della versione del pacchetto sull'`Published`utilizzo del `update-package-versions-status` comando.

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%
```

Di seguito è riportato l'output.

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

Elenco delle risorse del pacchetto generico

Per elencare gli asset contenuti in un pacchetto generico, utilizzate il `list-package-version-assets` comando. Per ulteriori informazioni, consulta Elenca le risorse della versione del pacchetto.

L'esempio seguente elenca gli asset della versione `1.0.0` del pacchetto`my-package`.

Per elencare gli asset della versione del pacchetto

- Chiama `list-package-version-assets` per elencare le risorse contenute in un pacchetto generico.

  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
  ```

  Di seguito è riportato l'output.

  ```
  {
      "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"
  ```

```
}
```

Scaricamento delle risorse del pacchetto generico

Per scaricare le risorse da un pacchetto generico, utilizzate il `get-package-version-asset` comando. Per ulteriori informazioni, consulta Scarica gli asset della versione del pacchetto.

L'esempio seguente scarica la risorsa `asset.tar.gz` dalla versione `1.0.0` del pacchetto `my-package` alla directory di lavoro corrente in un file denominato anch'essoasset.tar.gz.

Per scaricare gli asset della versione del pacchetto

- Chiama `get-package-version-asset` per scaricare risorse da un pacchetto generico.

  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
  ```

  Di seguito è riportato l'output.

  ```
  {
      "assetName": "asset.tar.gz",
      "packageVersion": "1.0.0",
      "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
  }
  ```

Utilizzo CodeArtifact con CodeBuild

Questi argomenti descrivono come utilizzare i pacchetti in un CodeArtifact repository in un progetto di AWS CodeBuild compilazione.

Argomenti

- Utilizzo dei pacchetti npm in CodeBuild
- Usare i pacchetti Python in CodeBuild
- Usare i pacchetti Maven in CodeBuild
- Utilizzo di pacchetti in NuGet CodeBuild
- Memorizzazione nella cache delle dipendenze

Utilizzo dei pacchetti npm in CodeBuild

I seguenti passaggi sono stati testati con i sistemi operativi elencati nelle immagini Docker fornite da. CodeBuild

Configura le autorizzazioni con i ruoli IAM

Questi passaggi sono necessari quando si utilizzano pacchetti npm da CodeArtifact in. CodeBuild

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, seleziona Ruoli. Nella pagina Ruoli, modifica il ruolo utilizzato dal tuo progetto di CodeBuild build. Questo ruolo deve avere le seguenti autorizzazioni.

   JSON

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

```
                    "codeartifact:ReadFromRepository"
                    ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Important**
>
> Se desideri utilizzarlo anche CodeBuild per pubblicare pacchetti, aggiungi l'**`codeartifact:PublishPackageVersion`**autorizzazione.

Per informazioni, consulta Modifying a Role nella IAM User Guide.

Accedi e usa npm

Per usare i pacchetti npm da CodeBuild, esegui il `login` comando dalla `pre-build` sezione del tuo progetto da `buildspec.yaml` cui configurare `npm` per il recupero dei pacchetti. CodeArtifact Per ulteriori informazioni, consulta Autenticazione con npm.

Dopo averlo `login` eseguito correttamente, puoi eseguire `npm` i comandi dalla `build` sezione per installare i pacchetti npm.

Linux

> **Note**
>
> È necessario eseguire l'aggiornamento AWS CLI con solo `pip3 install awscli --upgrade --user` se si utilizza un' CodeBuild immagine precedente. Se utilizzi le versioni più recenti dell'immagine, puoi rimuovere quella riga.

```
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
```

Usare i pacchetti Python in CodeBuild

I seguenti passaggi sono stati testati con i sistemi operativi elencati nelle immagini Docker fornite da. CodeBuild

Configura le autorizzazioni con i ruoli IAM

Questi passaggi sono necessari quando si utilizzano pacchetti Python dall' CodeArtifact interno. CodeBuild

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, seleziona Ruoli. Nella pagina Ruoli, modifica il ruolo utilizzato dal tuo progetto di CodeBuild build. Questo ruolo deve avere le seguenti autorizzazioni.

   JSON

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

> **Important**
>
> Se desideri utilizzarlo anche CodeBuild per pubblicare pacchetti, aggiungi l'**`codeartifact:PublishPackageVersion`**autorizzazione.

Per informazioni, consulta Modifying a Role nella IAM User Guide.

Accedi e usa pip o twine

Per usare i pacchetti Python da CodeBuild, esegui il `login` comando dalla `pre-build` sezione del `buildspec.yaml` file del tuo progetto da cui configurare `pip` per il recupero dei pacchetti. CodeArtifact Per ulteriori informazioni, consulta Usare CodeArtifact con Python.

Dopo `login` che è stato eseguito correttamente, puoi eseguire `pip` i comandi dalla `build` sezione per installare o pubblicare pacchetti Python.

Linux

> **Note**
>
> È necessario aggiornare il AWS CLI with solo `pip3 install awscli --upgrade --user` se si utilizza un' CodeBuild immagine precedente. Se utilizzi le versioni più recenti dell'immagine, puoi rimuovere quella riga.

Per installare i pacchetti Python usando: `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
```

Per pubblicare pacchetti Python usando: `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

Per installare i pacchetti Python usando: `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
```

Per pubblicare pacchetti Python usando: `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
```

Usare i pacchetti Maven in CodeBuild

Configura le autorizzazioni con i ruoli IAM

Questi passaggi sono necessari quando si utilizzano pacchetti Maven da in. CodeArtifact CodeBuild

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel riquadro di navigazione, seleziona Ruoli. Nella pagina Ruoli, modifica il ruolo utilizzato dal tuo progetto di CodeBuild build. Questo ruolo deve avere le seguenti autorizzazioni.

   JSON

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

> **Important**
>
> Se desideri utilizzarlo anche per CodeBuild pubblicare pacchetti, aggiungi i **`codeartifact:PutPackageMetadata`** permessi **`codeartifact:PublishPackageVersion`** and.

Per informazioni, consulta Modifying a Role nella IAM User Guide.

Usa gradle o mvn

Per utilizzare i pacchetti Maven con `gradle` o`mvn`, memorizza il token di CodeArtifact autenticazione in una variabile di ambiente, come descritto in Passare un token di autenticazione in una variabile di ambiente. Di seguito è riportato un esempio di :

> **Note**
>
> È necessario aggiornare il AWS CLI with solo `pip3 install awscli --upgrade --user` se si utilizza un'immagine precedente. CodeBuild Se utilizzi le versioni più recenti dell'immagine, puoi rimuovere quella riga.

```
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`
```

Per usare Gradle:

Se hai fatto riferimento alla `CODEARTIFACT_AUTH_TOKEN` variabile nel tuo `build.gradle` file Gradle come descritto in Uso CodeArtifact con Gradle, puoi richiamare la tua build Gradle dalla sezione. `buildspec.yaml` `build`

```
build:
  commands:
```

```
    - gradle build
```

Per usare mvn:

È necessario configurare i file di configurazione di Maven (`settings.xml`e`pom.xml`) seguire le istruzioni in Uso con CodeArtifact mvn.

Utilizzo di pacchetti in NuGet CodeBuild

I seguenti passaggi sono stati testati con i sistemi operativi elencati nelle immagini Docker fornite da CodeBuild.

Argomenti

- Configura le autorizzazioni con i ruoli IAM
- Consumare pacchetti NuGet
- Compila con NuGet pacchetti
- Pubblica NuGet pacchetti

Configura le autorizzazioni con i ruoli IAM

Questi passaggi sono necessari quando si utilizzano NuGet pacchetti da CodeArtifact in CodeBuild.

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, seleziona Ruoli. Nella pagina Ruoli, modifica il ruolo utilizzato dal tuo progetto di CodeBuild build. Questo ruolo deve avere le seguenti autorizzazioni.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [ "codeartifact:GetAuthorizationToken",
                      "codeartifact:GetRepositoryEndpoint",
                      "codeartifact:ReadFromRepository"
```

```
                    ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Important**
>
> Se desideri utilizzarlo anche CodeBuild per pubblicare pacchetti, aggiungi l'**`codeartifact:PublishPackageVersion`**autorizzazione.

Per informazioni, consulta Modifying a Role nella IAM User Guide.

Consumare pacchetti NuGet

Per utilizzare NuGet i pacchetti da CodeBuild, includi quanto segue nel `buildspec.yaml` file del progetto.

1. Nella `install` sezione, installa il CodeArtifact Credential Provider per configurare strumenti da riga di comando come `msbuild` e `dotnet` su cui creare e pubblicare CodeArtifact pacchetti.
2. Nella `pre-build` sezione, aggiungi il tuo CodeArtifact repository alla tua NuGet configurazione.

Vedi i seguenti `buildspec.yaml` esempi. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con NuGet.

Dopo aver installato il provider di credenziali e aggiunto l'origine del repository, puoi eseguire i comandi dello strumento NuGet CLI dalla `build` sezione per consumare i pacchetti. NuGet

Linux

Per utilizzare i pacchetti utilizzando: NuGet `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

Per consumare NuGet pacchetti utilizzando`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
```

Compila con NuGet pacchetti

Per creare con NuGet i pacchetti di CodeBuild, includi quanto segue nel `buildspec.yaml` file del progetto.

1. Nella `install` sezione, installa il CodeArtifact Credential Provider per configurare strumenti da riga di comando come `msbuild` e `dotnet` su cui creare e pubblicare CodeArtifact pacchetti.
2. Nella `pre-build` sezione, aggiungi il tuo CodeArtifact repository alla tua NuGet configurazione.

Vedi i seguenti `buildspec.yaml` esempi. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con NuGet.

Dopo aver installato il provider di credenziali e aggiunto l'origine del repository, puoi eseguire i comandi dello strumento NuGet CLI come `dotnet build` nella sezione. `build`

Linux

Per creare pacchetti usando NuGet : `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
```

Per creare NuGet pacchetti usando`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

Per creare NuGet pacchetti usando`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
```

Per creare NuGet pacchetti usando`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
```

Pubblica NuGet pacchetti

Per pubblicare NuGet pacchetti da CodeBuild, includi quanto segue nel `buildspec.yaml` file del progetto.

1. Nella `install` sezione, installa il CodeArtifact Credential Provider per configurare strumenti da riga di comando come `msbuild` e `dotnet` su cui creare e pubblicare CodeArtifact pacchetti.
2. Nella `pre-build` sezione, aggiungi il tuo CodeArtifact repository alla tua NuGet configurazione.

Vedi i seguenti `buildspec.yaml` esempi. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con NuGet.

Dopo aver installato il provider di credenziali e aggiunto l'origine del repository, puoi eseguire i comandi dello strumento NuGet CLI dalla `build` sezione e pubblicare i pacchetti. NuGet

Linux

Per pubblicare pacchetti utilizzando NuGet : `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

Per pubblicare NuGet pacchetti utilizzando`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
```

Memorizzazione nella cache delle dipendenze

Puoi abilitare la memorizzazione nella cache locale CodeBuild per ridurre il numero di dipendenze da cui è necessario recuperare per ogni build. CodeArtifact Per informazioni, consulta Build Caching AWS CodeBuild nella Guida per l'utente.AWS CodeBuild Dopo aver abilitato una cache locale personalizzata, aggiungi la directory della cache al `buildspec.yaml` file del progetto.

Ad esempio, se stai usando`mvn`, usa quanto segue.

```
cache:
  paths:
    - '/root/.m2/**/*'
```

Per altri strumenti, utilizzate le cartelle della cache mostrate in questa tabella.

Strumento	Directory della cache
mvn	/root/.m2/**/*
gradle	/root/.gradle/caches/**/*
pip	/root/.cache/pip/**/*
npm	/root/.npm/**/*
nuget	/root/.nuget/**/*
yarn (classic)	/root/.cache/yarn/**/*

Monitoraggio CodeArtifact

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle CodeArtifact altre AWS soluzioni. AWS fornisce i seguenti strumenti di monitoraggio per osservare CodeArtifact, segnalare quando qualcosa non va e intraprendere azioni automatiche quando necessario:

- Puoi utilizzare Amazon EventBridge per automatizzare AWS i tuoi servizi e rispondere automaticamente agli eventi di sistema, come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi AWS relativi ai servizi vengono forniti quasi EventBridge in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali operazioni automatizzate intraprendere quando un evento corrisponde a una regola. Per ulteriori informazioni, consulta Amazon EventBridge User Guide eCodeArtifact formato ed esempio dell'evento.
- Puoi utilizzare i CloudWatch parametri di Amazon per visualizzare CodeArtifact l'utilizzo per operazione. CloudWatch le metriche includono tutte le richieste fatte a e CodeArtifact le richieste vengono mostrate per account. Puoi visualizzare queste metriche nelle CloudWatch metriche accedendo allo spazio dei nomi Usage/By AWS Resource. AWS Per ulteriori informazioni, consulta Use Amazon CloudWatch metrics nella Amazon CloudWatch User Guide.

Argomenti

- Monitoraggio degli eventi CodeArtifact
- Usa un evento per avviare un'esecuzione CodePipeline
- Utilizzare un evento per eseguire una funzione Lambda

Monitoraggio degli eventi CodeArtifact

CodeArtifact è integrato con Amazon EventBridge, un servizio che automatizza e risponde agli eventi, comprese le modifiche in un CodeArtifact repository. Puoi creare regole per gli eventi e configurare cosa succede quando un evento corrisponde a una regola. EventBridge in precedenza si CloudWatch chiamava Events.

Le seguenti azioni possono essere attivate da un evento:

- Invocare una funzione AWS Lambda .

- Attivazione di una macchina a AWS Step Functions stati.
- Notifica di un argomento di Amazon SNS o di una coda Amazon SQS.
- Avvio di una pipeline in. AWS CodePipeline

CodeArtifact crea un evento quando viene creata, modificata o eliminata una versione del pacchetto. Di seguito sono riportati alcuni esempi di CodeArtifact eventi:

- Pubblicazione di una nuova versione del pacchetto (ad esempio, eseguendo`npm publish`).
- Aggiungere una nuova risorsa a una versione del pacchetto esistente (ad esempio, inserendo un nuovo file JAR in un pacchetto Maven esistente).
- Copiare una versione del pacchetto da un repository a un altro utilizzando. `copy-package-versions` Per ulteriori informazioni, consulta Copia i pacchetti tra i repository.
- Eliminazione delle versioni del pacchetto utilizzando. `delete-package-versions` Per ulteriori informazioni, consulta Eliminare un pacchetto o una versione del pacchetto.
- Eliminazione delle versioni di un pacchetto utilizzando. `delete-package` Verrà pubblicato un evento per ogni versione del pacchetto eliminato. Per ulteriori informazioni, consulta Eliminare un pacchetto o una versione del pacchetto.
- Conservazione della versione del pacchetto in un repository downstream quando è stata recuperata da un repository upstream. Per ulteriori informazioni, consulta Lavorare con i repository upstream in CodeArtifact.
- Inserimento di una versione del pacchetto da un repository esterno in un repository. CodeArtifact Per ulteriori informazioni, consulta Connect un CodeArtifact repository a un repository pubblico.

Gli eventi vengono recapitati sia all'account proprietario del dominio sia all'account che amministra il repository. Ad esempio, supponiamo che l'account `111111111111` possieda il dominio. `my_domain` L'account `222222222222` crea un repository in `my_domain` called. `repo2` Quando viene pubblicata una nuova versione del pacchetto su`repo2`, entrambi gli account ricevono gli EventBridge eventi. L'account proprietario del dominio (`111111111111`) riceve gli eventi per tutti gli archivi del dominio. Se un singolo account possiede sia il dominio che il repository al suo interno, viene consegnato un solo evento.

I seguenti argomenti descrivono il formato CodeArtifact dell'evento. Ti mostrano come configurare CodeArtifact gli eventi e come utilizzarli con altri AWS servizi. Per ulteriori informazioni, consulta la sezione Getting Started with Amazon EventBridge nella Amazon EventBridge User Guide.

CodeArtifact formato ed esempio dell'evento

Di seguito sono riportati i campi e le descrizioni degli eventi insieme a un esempio di CodeArtifact evento.

CodeArtifact formato dell'evento

Tutti CodeArtifact gli eventi includono i seguenti campi.

Campo Evento	Descrizione
versione	La versione del formato dell'evento . Attualmente esiste una sola versione,`0`.
id	Un identificatore univoco per l'evento.
detail-type (tipo di dettaglio)	Tipo di evento. Questo determina i campi dell'`detail`oggetto. Quello `detail-type` attualmente supportato è`CodeArtifact Package Version State Change`.
source	L'origine dell'evento. Perché CodeArtifact lo saràaws.codeartifact .
account	L' AWS ID dell'account che riceve l'evento.
time	L'ora esatta in cui è stato attivato l'evento.
Regione	La regione in cui è stato attivato l'evento.
risorse	Un elenco che contiene l'ARN del pacchetto modificato. L'elenco contiene una voce. Per informazioni sul formato ARN del pacchetto , vedere. Concedi l'accesso in scrittura ai pacchetti
domainName	Il dominio che contiene il repository che contiene il pacchetto.

Campo Evento	Descrizione
Proprietario del dominio	L'ID dell' AWS account del proprietario del dominio.
Nome del repository	Il repository che contiene il pacchetto.
Amministratore del repository	L'ID AWS dell'account dell'amministratore del repository.
Formato del pacchetto	Il formato del pacchetto che ha attivato l'evento.
PackageNameSpace	Lo spazio dei nomi del pacchetto che ha attivato l'evento.
Nome del pacchetto	Il nome del pacchetto che ha attivato l'evento.
Versione del pacchetto	La versione del pacchetto che ha attivato l'evento.
packageVersionState	Lo stato della versione del pacchetto al momento dell'attivazione dell'evento. I valori possibili sono `Unfinished` , `Published` , `Unlisted`, `Archived` e `Disposed`.
packageVersionRevision	Un valore che identifica in modo univoco lo stato delle risorse e dei metadati della versione del pacchetto al momento dell'attivazione dell'evento. Se la versione del pacchetto viene modificata (ad esempio, aggiungendo un altro file JAR a un pacchetto Maven), le modifiche vengono apportate. `packageVersionRevision`
Modifiche.Assets Added	Il numero di risorse aggiunte a un pacchetto che ha attivato un evento. Esempi di una risorsa sono un file JAR Maven o una ruota Python.

Campo Evento	Descrizione
Modifiche. Risorse rimosse	Il numero di risorse rimosse da un pacchetto che ha attivato un evento.
Changes.Assets (aggiornato)	Il numero di asset modificati nel pacchetto che ha attivato l'evento.
Modifiche.MetadataUpdated	Un valore booleano impostato su se l'evento include metadati modificati a livello di pacchetto `true`. Ad esempio, un evento potrebbe modificare un file Maven. `pom.xml`
changes.statusChanged	Un valore booleano impostato su `true` se l'evento `packageVersionStatus` viene modificato (ad esempio, se cambia da a). `packageVersionStatus` `Unfinished` `Published`
Tipo di operazione	Descrive il tipo di alto livello della modifica della versione del pacchetto. I valori possibili sono `Created`, `Updated` e `Deleted`.

Campo Evento	Descrizione
SequenceNumber	Un numero intero che specifica un numero di evento per un pacchetto. Ogni evento su un pacchetto incrementa la dimensione in modo che gli eventi `sequenceNumber` possano essere disposti in sequenza. Un evento può incrementarlo di qualsiasi `sequenceNumber` numero intero. **Note** EventBridge gli eventi potrebbero essere ricevuti in modo errato. `sequenceNumber` possono essere utilizzati per determinarne l'ordine effettivo.
eventDeduplicationId	Un ID utilizzato per differenziare gli eventi duplicati. EventBridge In rari casi, EventBridge potrebbe attivare la stessa regola più di una volta per un singolo evento o orario programmato. In alternativa, potrebbe richiamare lo stesso obiettivo più di una volta per una determinata regola attivata.

CodeArtifact esempio di evento

Di seguito è riportato un esempio di CodeArtifact evento che potrebbe essere attivato quando viene pubblicato un pacchetto npm.

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


Usa un evento per avviare un'esecuzione CodePipeline

Questo esempio dimostra come configurare una EventBridge regola Amazon in modo che AWS CodePipeline l'esecuzione inizi quando una versione del pacchetto in un CodeArtifact repository viene pubblicata, modificata o eliminata.

Argomenti

- Configura le autorizzazioni EventBridge
- Crea la regola EventBridge
- Crea l'obiettivo della EventBridge regola

Configura le autorizzazioni EventBridge

È necessario aggiungere le autorizzazioni EventBridge da utilizzare CodePipeline per richiamare la regola creata. Per aggiungere queste autorizzazioni utilizzando AWS Command Line Interface (AWS CLI), segui il passaggio 1 in Creare una regola di CloudWatch eventi per una CodeCommit sorgente (CLI) nella Guida per AWS CodePipeline l'utente.

Crea la regola EventBridge

Per creare la regola, utilizzate il `put-rule` comando con i `--event-pattern` parametri `--name` and. Il modello di evento specifica i valori che vengono confrontati con il contenuto di ogni evento. Il bersaglio viene attivato se il pattern corrisponde all'evento. Ad esempio, lo schema seguente corrisponde CodeArtifact agli eventi del `myrepo` repository nel `my_domain` dominio.

```
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"]}}'
```

Crea l'obiettivo della EventBridge regola

Il comando seguente aggiunge un obiettivo alla regola in modo che quando un evento corrisponde alla regola, venga attivata un' CodePipeline esecuzione. Per il `RoleArn` parametro, specifica l'Amazon Resource Name (ARN) del ruolo creato in precedenza in questo argomento.

```
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'
```

Utilizzare un evento per eseguire una funzione Lambda

Questo esempio mostra come configurare una EventBridge regola che avvia una AWS Lambda funzione quando una versione di pacchetto in un CodeArtifact repository viene pubblicata, modificata o eliminata.

Per ulteriori informazioni, consulta Tutorial: Schedule AWS Lambda Functions Using EventBridge in the Amazon EventBridge User Guide.

Argomenti

- Crea la EventBridge regola
- Crea l'obiettivo della regola EventBridge
- Configura le autorizzazioni EventBridge

Crea la EventBridge regola

Per creare una regola che avvia una funzione Lambda, usa il `put-rule` comando con le opzioni `--name` e`--event-pattern`. Lo schema seguente specifica i pacchetti npm nell'`@types`ambito di qualsiasi repository del dominio. `my_domain`

```
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"]}}'
```

Crea l'obiettivo della regola EventBridge

Il comando seguente aggiunge una destinazione alla regola che esegue la funzione Lambda quando un evento corrisponde alla regola. Per il `arn` parametro, specifica l'Amazon Resource Name (ARN) della funzione Lambda.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

Configura le autorizzazioni EventBridge

Usa il `add-permission` comando per concedere le autorizzazioni alla regola per richiamare una funzione Lambda. Per il `--source-arn` parametro, specificate l'ARN della regola creata in precedenza in questo esempio.

```
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
```

Sicurezza in CodeArtifact

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità applicabili CodeArtifact, consulta AWS Services in Scope by Compliance Program .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa durante l'utilizzo CodeArtifact. I seguenti argomenti mostrano come configurare per CodeArtifact soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a usare altri servizi AWS che ti aiutano a monitorare e proteggere CodeArtifact le tue risorse.

Argomenti

- Protezione dei dati in AWS CodeArtifact
- Monitoraggio CodeArtifact
- Convalida della conformità per AWS CodeArtifact
- AWS CodeArtifact autenticazione e token
- Resilienza in AWS CodeArtifact
- Sicurezza dell'infrastruttura in AWS CodeArtifact
- Attacchi di sostituzione delle dipendenze
- Identity and Access Management per AWS CodeArtifact

Protezione dei dati in AWS CodeArtifact

Il modello di responsabilità AWS condivisa modello di si applica alla protezione dei dati in AWS CodeArtifact. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutto il Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori CodeArtifact o Servizi AWS utilizzi la console, l'API o. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Crittografia dei dati

La crittografia è una parte importante della CodeArtifact sicurezza. Alcune crittografie, ad esempio per i dati in transito, sono fornite di default e non richiedono alcuna operazione da parte dell'utente. Altre forme di crittografia, ad esempio per i dati inattivi, possono essere configurate durante la creazione del progetto o della build.

- Crittografia dei dati inattivi: tutte le risorse archiviate CodeArtifact vengono crittografate utilizzando AWS KMS keys (chiavi KMS). Ciò include tutte le risorse in tutti i pacchetti in tutti i repository. Una chiave KMS viene utilizzata per ogni dominio per crittografare tutte le sue risorse. Per impostazione predefinita, viene utilizzata una chiave KMS AWS gestita, quindi non è necessario creare una chiave KMS. Se lo desideri, puoi utilizzare una chiave KMS gestita dal cliente che crei e configuri. Per ulteriori informazioni, consulta Creating keys e AWS Key Management Service Concetti nella Guida per l'AWS Key Management Service utente. Puoi specificare una chiave KMS gestita dal cliente quando crei un dominio. Per ulteriori informazioni, consulta Lavorare con i domini in CodeArtifact.
- Crittografia dei dati in transito: tutte le comunicazioni tra i clienti e tra i clienti CodeArtifact e le relative dipendenze a valle sono protette mediante la crittografia TLS. CodeArtifact

Privacy del traffico

Puoi migliorare la sicurezza dei tuoi CodeArtifact domini e degli asset in essi contenuti CodeArtifact configurando l'uso di un endpoint VPC (Virtual Private Cloud) con interfaccia. A tale scopo, non è necessario un gateway Internet, un dispositivo NAT o un gateway privato virtuale. Per ulteriori informazioni, consulta Utilizzo degli endpoint Amazon VPC. Per ulteriori informazioni sugli endpoint VPC, consulta AWS PrivateLink e AWS PrivateLinkAccesso ai servizi AWS Through. PrivateLink

Monitoraggio CodeArtifact

Il monitoraggio è un elemento importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle AWS CodeArtifact vostre AWS soluzioni. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. AWS fornisce quanto segue per monitorare le CodeArtifact risorse e rispondere a potenziali incidenti:

Argomenti

- Registrazione delle chiamate API con CodeArtifact AWS CloudTrail

Registrazione delle chiamate API con CodeArtifact AWS CloudTrail

CodeArtifact è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, un ruolo o un AWS servizio in CodeArtifact. CloudTrail acquisisce tutte le chiamate API CodeArtifact come eventi, incluse le chiamate dai client del gestore di pacchetti.

Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon Simple Storage Service (Amazon S3), inclusi gli eventi per. CodeArtifact Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare a quale richiesta è stata inviata CodeArtifact, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

CodeArtifact informazioni in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in CodeArtifact, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi AWS di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi di CodeArtifact, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Puoi anche configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta i seguenti argomenti:

- Creazione di un percorso per il tuo account AWS
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail

Quando CloudTrail la registrazione è abilitata nel tuo AWS account, le chiamate API effettuate alle CodeArtifact azioni vengono tracciate nei file di CloudTrail registro, dove vengono scritte insieme ad

altri record di servizio. AWS CloudTrail determina quando creare e scrivere su un nuovo file in base al periodo di tempo e alle dimensioni del file.

Tutte CodeArtifact le azioni vengono registrate da CloudTrail. Ad esempio, le chiamate alle azioni `ListRepositories` (in AWS CLI,`aws codeartifact list-repositories`), `CreateRepository` (`aws codeartifact create-repository`) e `ListPackages` (`aws codeartifact list-packages`) generano voci nei file di CloudTrail registro, oltre ai comandi client del gestore di pacchetti. I comandi client del gestore di pacchetti in genere inviano più di una richiesta HTTP al server. Ogni richiesta genera un evento di CloudTrail registro separato.

Consegna dei log tra più CloudTrail account

Fino a tre account separati ricevono i CloudTrail log per una singola chiamata API:

- L'account che ha effettuato la richiesta, ad esempio l'account che ha chiamato. `GetAuthorizationToken`
- L'account dell'amministratore del repository, ad esempio l'account che amministra il repository a cui è stato effettuato l'accesso. `ListPackages`
- L'account del proprietario del dominio, ad esempio l'account proprietario del dominio che contiene l'archivio a cui è stata chiamata un'API.

Ad APIs esempio, `ListRepositoriesInDomain` si tratta di azioni contro un dominio e non un repository specifico, solo l'account chiamante e l'account del proprietario del dominio ricevono il registro. CloudTrail Perché APIs `ListRepositories` tali dati non sono autorizzati nei confronti di alcuna risorsa, solo l'account del chiamante riceve il CloudTrail registro.

Comprensione delle CodeArtifact voci dei file di registro

CloudTrail i file di registro possono contenere una o più voci di registro. Ogni voce elenca più eventi in formato JSON. Un evento di log rappresenta una singola richiesta inviata da un'origine e include informazioni sull'operazione richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. Le voci di log non sono una traccia di stack ordinata delle chiamate API pubbliche, pertanto non vengono visualizzate in un ordine specifico.

Argomenti

- Esempio: una voce di registro per chiamare l' GetAuthorizationToken API
- Esempio: una voce di registro per recuperare una versione del pacchetto npm

Esempio: una voce di registro per chiamare l' GetAuthorizationToken API

Una voce di registro creata da `GetAuthorizationToken` include il nome di dominio nel `requestParameters` campo.


```
{
  "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"
}
```

Esempio: una voce di registro per recuperare una versione del pacchetto npm

Le richieste effettuate da tutti i client di gestione pacchetti, incluso il **npm**client, hanno dati aggiuntivi registrati, tra cui il nome di dominio, il nome del repository e il nome del pacchetto nel campo. `requestParameters` Il percorso URL e il metodo HTTP vengono registrati nel campo. `additionalEventData`

```
{
   "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"
}
```


Convalida della conformità per AWS CodeArtifact

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta AWS la documentazione sulla sicurezza.

AWS CodeArtifact autenticazione e token

CodeArtifact richiede agli utenti di autenticarsi con il servizio per pubblicare o utilizzare le versioni dei pacchetti. È necessario autenticarsi al CodeArtifact servizio creando un token di autorizzazione utilizzando le proprie AWS credenziali. Per creare un token di autorizzazione, è necessario disporre delle autorizzazioni corrette. Per le autorizzazioni necessarie per creare un token di autorizzazione, vedere la `GetAuthorizationToken` voce in. AWS CodeArtifact riferimento alle autorizzazioni Per informazioni più generali sulle CodeArtifact autorizzazioni, vedere. Come AWS CodeArtifact funziona con IAM

Per recuperare un token di autorizzazione da CodeArtifact, devi chiamare l'GetAuthorizationToken API. Utilizzando AWS CLI, è possibile chiamare `GetAuthorizationToken` con il `get-authorization-token` comando `login` or.

> **Note**
>
> Gli utenti root non possono effettuare chiamate`GetAuthorizationToken`.

- `aws codeartifact login`: Questo comando semplifica la configurazione dei gestori di pacchetti comuni da utilizzare CodeArtifact in un unico passaggio. Calling `login` recupera un token `GetAuthorizationToken` e configura il gestore di pacchetti con il token e l'endpoint del CodeArtifact repository corretto. I gestori dei pacchetti di supporto sono i seguenti:
  - dotnet
  - npm
  - pepita
  - pip
  - veloce
  - spago
- `aws codeartifact get-authorization-token`: Per i gestori di pacchetti non supportati da`login`, è possibile chiamare `get-authorization-token` direttamente e quindi configurare il gestore di pacchetti con il token come richiesto, ad esempio aggiungendolo a un file di configurazione o memorizzandolo in una variabile di ambiente.

CodeArtifact i token di autorizzazione sono validi per un periodo predefinito di 12 ore. I token possono essere configurati con una durata compresa tra 15 minuti e 12 ore. Quando il periodo di validità scade, devi recuperare un altro token. La durata del token inizia dopo `login` o `get-authorization-token` è stato chiamato.

Se `login` o `get-authorization-token` viene chiamato mentre si assume un ruolo, è possibile configurare la durata del token in modo che sia uguale al tempo rimanente nella durata della sessione del ruolo impostando il valore `--duration-seconds` su to`0`. Altrimenti, la durata del token è indipendente dalla durata massima della sessione del ruolo. Ad esempio, supponiamo di chiamare `sts assume-role` e specificare una durata della sessione di 15 minuti, quindi di chiamare `login` per recuperare un token di CodeArtifact autorizzazione. In questo caso, il token è valido per l'intero

periodo di 12 ore, anche se è più lungo della durata della sessione di 15 minuti. Per informazioni sul controllo della durata della sessione, consulta Using IAM Roles nella IAM User Guide.

Token creati con il comando `login`

Il `aws codeartifact login` comando recupererà un token `GetAuthorizationToken` e configurerà il gestore di pacchetti con il token e l'endpoint del CodeArtifact repository corretto.

La tabella seguente descrive i parametri per il comando. `login`

Parametro	Obbligatorio	Descrizione
`--tool`	Sì	Il gestore di pacchetti con cui autenticarsi. I valori possibili sono`dotnet`,`npm`, `nugetpip`, `swift` `etwine`.
`--domain`	Sì	Il nome di dominio a cui appartiene il repository.
`--domain-owner`	No	L'ID del proprietario del dominio. Questo parametro è obbligatorio se si accede a un dominio di proprietà di un AWS account al quale non si è autenticati. Per ulteriori informazioni, consulta Domini con più account.
`--repository`	Sì	Il nome del repository in cui effettuare l'autenticazione.
`--duration-seconds`	No	Il tempo, in secondi, di validità delle informazioni di accesso. Il valore minimo è 900* e il valore massimo è 43200.
`--namespace`	No	Associa uno spazio dei nomi allo strumento di repository.

Parametro	Obbligatorio	Descrizione
`--dry-run`	No	Stampa solo i comandi che verrebbero eseguiti per connetter e lo strumento al repository senza apportare modifiche alla configura zione.

*Il valore 0 è valido anche quando si chiama `login` assumendo un ruolo. La chiamata `login` con `--duration-seconds 0` crea un token con una durata pari al tempo rimanente nella durata della sessione di un ruolo assunto.

L'esempio seguente mostra come recuperare un token di autorizzazione con il `login` comando.

```
aws codeartifact login \
    --tool dotnet | npm | nuget | pip | swift | twine \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo
```

Per indicazioni specifiche su come utilizzare il `login` comando con npm, vedere. Configura e usa npm con CodeArtifact Per Python, vedi. Usare CodeArtifact con Python

Autorizzazioni necessarie per chiamare l'API **`GetAuthorizationToken`**

Per chiamare l'`sts:GetServiceBearerToken`API sono necessarie sia `codeartifact:GetAuthorizationToken` le autorizzazioni che le autorizzazioni. CodeArtifact `GetAuthorizationToken`

Per utilizzare un gestore di pacchetti con un CodeArtifact repository, l'utente o il ruolo IAM deve consentirlo. `sts:GetServiceBearerToken` Sebbene `sts:GetServiceBearerToken` possa essere aggiunta a una politica delle risorse di CodeArtifact dominio, l'autorizzazione non avrà alcun effetto su tale politica.

Token creati con l'API **`GetAuthorizationToken`**

Puoi chiamare `get-authorization-token` per recuperare un token di autorizzazione da. CodeArtifact

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text
```

È possibile modificare la durata di validità di un token utilizzando l'`--duration-seconds`argomento. Il valore minimo è 900 e il valore massimo è 43200. L'esempio seguente crea un token che durerà 1 ora (3600 secondi).

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 3600
```

Se si chiama `get-authorization-token` mentre si assume un ruolo, la durata del token è indipendente dalla durata massima della sessione del ruolo. È possibile configurare il token in modo che scada alla scadenza della durata della sessione del ruolo assunto `--duration-seconds` impostandolo su 0.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 0
```

Per ulteriori informazioni, consulta la seguente documentazione:

- Per indicazioni sui token e sulle variabili di ambiente, vederePassa un token di autenticazione utilizzando una variabile di ambiente.
- Per gli utenti di Python, vedere Configura pip senza il comando login o. Configura e usa twine con CodeArtifact
- Per gli utenti Maven, vedi o. Uso CodeArtifact con Gradle Usare CodeArtifact con mvn
- Per gli utenti di npm, vedi. Configurazione di npm senza utilizzare il comando login

Passa un token di autenticazione utilizzando una variabile di ambiente

AWS CodeArtifact utilizza i token di autorizzazione forniti dall'`GetAuthorizationToken`API per autenticare e autorizzare le richieste provenienti da strumenti di compilazione come Maven e Gradle. Per ulteriori informazioni su questi token di autenticazione, consulta. [Token creati con l'API `GetAuthorizationToken`]

È possibile memorizzare questi token di autenticazione in una variabile di ambiente che può essere letta da uno strumento di compilazione per ottenere il token necessario per recuperare i pacchetti da un CodeArtifact repository o pubblicarvi pacchetti.

Per motivi di sicurezza, questo approccio è preferibile alla memorizzazione del token in un file in cui potrebbe essere letto da altri utenti o processi o verificato accidentalmente nel controllo del codice sorgente.

1. Configura le tue AWS credenziali come descritto in. [Installa o aggiorna e quindi configura il AWS CLI]
2. Imposta la variabile di ambiente `CODEARTIFACT_AUTH_TOKEN`:

   > **Note**
   >
   > In alcuni scenari, non è necessario includere l'`--domain-owner`argomento. Per ulteriori informazioni, consulta [Domini con più account].

   - macOS o Linux:

     ```
     export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
     domain my_domain --domain-owner 111122223333 --query authorizationToken --output
      text`
     ```

   - Windows (utilizzando la shell di comando predefinita):

     ```
     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
```

Revoca CodeArtifact dei token di autorizzazione

Quando un utente autenticato crea un token per accedere alle CodeArtifact risorse, tale token dura fino al termine del periodo di accesso personalizzabile. Il periodo di accesso predefinito è di 12 ore. In alcune circostanze, potresti voler revocare l'accesso a un token prima della scadenza del periodo di accesso. Puoi revocare l'accesso alle CodeArtifact risorse seguendo queste istruzioni.

Se hai creato il token di accesso utilizzando credenziali di sicurezza temporanee, come ruoli assunti o accesso utente federato, puoi revocare l'accesso aggiornando una policy IAM per negare l'accesso. Per informazioni, consulta Disabling Permissions for Temporary Security Credentials nella IAM User Guide.

Se hai utilizzato credenziali utente IAM a lungo termine per creare il token di accesso, devi modificare la policy dell'utente per negare l'accesso o eliminare l'utente IAM. Per ulteriori informazioni, consulta Modifica delle autorizzazioni per un utente IAM o Eliminazione di un utente IAM.

Resilienza in AWS CodeArtifact

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. AWS CodeArtifact opera in più zone di disponibilità e archivia dati e metadati relativi agli artefatti in Amazon S3 e Amazon DynamoDB. I tuoi dati crittografati vengono archiviati in modo ridondante su più strutture e più dispositivi in ogni struttura, il che li rende altamente disponibili e altamente durevoli.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta AWS Global Infrastructure.

Sicurezza dell'infrastruttura in AWS CodeArtifact

In quanto servizio gestito, AWS CodeArtifact è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzate chiamate API AWS pubblicate per accedere CodeArtifact attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Attacchi di sostituzione delle dipendenze

I Package Manager semplificano il processo di imballaggio e condivisione del codice riutilizzabile. Questi pacchetti possono essere pacchetti privati sviluppati da un'organizzazione per essere utilizzati nelle proprie applicazioni, oppure possono essere pacchetti pubblici, in genere pacchetti open source sviluppati all'esterno di un'organizzazione e distribuiti da archivi di pacchetti pubblici. Quando richiedono pacchetti, gli sviluppatori si affidano al proprio gestore di pacchetti per recuperare nuove versioni delle proprie dipendenze. Gli attacchi di sostituzione delle dipendenze, noti anche come attacchi basati sulla confusione delle dipendenze, sfruttano il fatto che un gestore di pacchetti in genere non ha modo di distinguere le versioni legittime di un pacchetto da quelle dannose.

Gli attacchi di sostituzione delle dipendenze appartengono a un sottoinsieme di attacchi noti come attacchi alla catena di fornitura del software. Un attacco alla catena di fornitura del software è un attacco che sfrutta le vulnerabilità in qualsiasi punto della catena di fornitura del software.

Un attacco basato sulla sostituzione delle dipendenze può colpire chiunque utilizzi sia pacchetti sviluppati internamente sia pacchetti recuperati da archivi pubblici. Gli aggressori identificano i nomi interni dei pacchetti e quindi posizionano strategicamente il codice dannoso con lo stesso nome negli archivi pubblici dei pacchetti. In genere, il codice dannoso viene pubblicato in un pacchetto con un numero di versione elevato. I gestori di pacchetti recuperano il codice dannoso da questi feed pubblici perché ritengono che i pacchetti dannosi siano le versioni più recenti del pacchetto. Ciò causa una «confusione» o una «sostituzione» tra il pacchetto desiderato e il pacchetto dannoso, con conseguente compromissione del codice.

Per prevenire attacchi di sostituzione delle dipendenze, fornisce controlli sull'origine dei pacchetti. AWS CodeArtifact I controlli di origine dei pacchetti sono impostazioni che controllano il modo in cui i pacchetti possono essere aggiunti ai repository. I controlli possono essere utilizzati per garantire che le versioni dei pacchetti non possano essere sia pubblicate direttamente nel repository sia importate da fonti pubbliche, proteggendovi dagli attacchi di sostituzione delle dipendenze. I controlli di origine

possono essere impostati su pacchetti singoli e su più pacchetti impostando i controlli di origine sui gruppi di pacchetti. Per ulteriori informazioni sui controlli di origine dei pacchetti e su come modificarli, consulta Modifica dei controlli di origine dei pacchetti eControlli dell'origine dei gruppi di pacchetti.

Identity and Access Management per AWS CodeArtifact

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse. CodeArtifact IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell’accesso tramite policy
- Come AWS CodeArtifact funziona con IAM
- Esempi di policy basate sull'identità per AWS CodeArtifact
- Utilizzo dei tag per controllare l'accesso alle risorse CodeArtifact
- AWS CodeArtifact riferimento alle autorizzazioni
- Risoluzione dei problemi relativi a identità e accesso AWS CodeArtifact

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all’amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi relativi a identità e accesso AWS CodeArtifact)
- Amministratore del servizio: determina l’accesso degli utenti e invia le richieste di autorizzazione (consulta Come AWS CodeArtifact funziona con IAM)
- Amministratore IAM: scrivi policy per gestire l’accesso (consulta Esempi di policy basate sull'identità per AWS CodeArtifact)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la

federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso federato degli utenti, le autorizzazioni utente IAM temporanee, l'accesso tra account, l'accesso tra servizi e le applicazioni in esecuzione su Amazon. EC2 Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e collegandole a identità o risorse. AWS Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come

creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come AWS CodeArtifact funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a CodeArtifact, scopri con quali funzionalità IAM è disponibile l'uso CodeArtifact.

Funzionalità IAM che puoi utilizzare con AWS CodeArtifact

Funzionalità IAM	CodeArtifact supporto
Policy basate sull'identità	Sì
Policy basate su risorse	Sì
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	No
ACLs	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	No
Ruoli collegati al servizio	No

Per avere una panoramica di alto livello su come CodeArtifact e altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Politiche basate sull'identità per CodeArtifact

Supporta le policy basate sull'identità: sì

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Esempi di politiche basate sull'identità per CodeArtifact

Per visualizzare esempi di politiche basate sull' CodeArtifact identità, vedere. Esempi di policy basate sull'identità per AWS CodeArtifact

Politiche basate sulle risorse all'interno CodeArtifact

Supporta le policy basate sulle risorse: sì

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni politiche per CodeArtifact

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di CodeArtifact azioni, vedere Azioni definite da AWS CodeArtifact nel Service Authorization Reference.

Le azioni politiche in CodeArtifact uso utilizzano il seguente prefisso prima dell'azione:

```
codeartifact
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "codeartifact:action1",
      "codeartifact:action2"
         ]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "codeartifact:Describe*"
```

Per visualizzare esempi di politiche CodeArtifact basate sull'identità, vedere. Esempi di policy basate sull'identità per AWS CodeArtifact

Risorse politiche per CodeArtifact

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon

(ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di CodeArtifact risorse e relativi ARNs, vedere Resources defined by AWS CodeArtifact nel Service Authorization Reference. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta la sezione Operazioni definite da AWS CodeArtifact. Per vedere esempi di specificazione CodeArtifact delle risorse ARNs nelle politiche, vedereAWS CodeArtifact risorse e operazioni.

Chiavi relative alle condizioni delle politiche per CodeArtifact

Supporta le chiavi di condizione delle policy specifiche del servizio: No

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

> **Note**
>
> AWS CodeArtifact non supporta le seguenti chiavi di contesto della condizione AWS globale:
>
> - Referente
> - UserAgent

Per visualizzare un elenco di chiavi di CodeArtifact condizione, consulta Condition keys for AWS CodeArtifact nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, vedi Azioni definite da AWS CodeArtifact.

Per visualizzare esempi di politiche CodeArtifact basate sull'identità, vedere. Esempi di policy basate sull'identità per AWS CodeArtifact

ACLs in CodeArtifact

Supporti: No ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con CodeArtifact

Supporta ABAC (tag nelle policy): parzialmente

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi, chiamati tag. Puoi allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Per ulteriori informazioni sull'etichettatura CodeArtifact delle risorse, inclusi esempi di politiche basate sull'identità per limitare l'accesso a una risorsa in base ai tag presenti su quella risorsa, consulta. Utilizzo dei tag per controllare l'accesso alle risorse CodeArtifact

Utilizzo di credenziali temporanee con CodeArtifact

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare

dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Autorizzazioni principali multiservizio per CodeArtifact

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Esistono due azioni CodeArtifact API che richiedono al principale chiamante di disporre delle autorizzazioni per altri servizi:

1. `GetAuthorizationToken`richiede `sts:GetServiceBearerToken` insieme `codeartifact:GetAuthorizationToken` a.
2. `CreateDomain`, quando si fornisce una chiave di crittografia non predefinita, richiede entrambe le chiavi `kms:DescribeKey` e `kms:CreateGrant` insieme alla chiave KMS. `codeartifact:CreateDomain`

Per ulteriori informazioni sulle autorizzazioni e sulle risorse necessarie per le azioni in CodeArtifact, consulta. AWS CodeArtifact riferimento alle autorizzazioni

Ruoli di servizio per CodeArtifact

Supporta i ruoli di servizio: no

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità. CodeArtifact Modifica i ruoli di servizio solo quando viene CodeArtifact fornita una guida in tal senso.

Ruoli collegati ai servizi per CodeArtifact

Supporta i ruoli collegati ai servizi: no

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate sull'identità per AWS CodeArtifact

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse CodeArtifact. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle azioni e sui tipi di risorse definiti da CodeArtifact, incluso il formato di ARNs per ogni tipo di risorsa, vedere Azioni, risorse e chiavi di condizione AWS CodeArtifact nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console di CodeArtifact
- Policy (predefinite) gestite da AWS per AWS CodeArtifact
- Consenti a un utente di visualizzare le proprie autorizzazioni
- Consenti a un utente di ottenere informazioni su repository e domini
- Consenti a un utente di ottenere informazioni su domini specifici
- Consenti a un utente di ottenere informazioni su repository specifici
- Limita la durata del token di autorizzazione

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare CodeArtifact risorse nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consultare Best practice di sicurezza in IAM nella Guida per l'utente IAM.

Utilizzo della console di CodeArtifact

Per accedere alla AWS CodeArtifact console, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle CodeArtifact risorse del tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console agli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano ancora utilizzare la CodeArtifact console, allega anche la policy `AWSCodeArtifactAdminAccess` o la policy `AWSCodeArtifactReadOnlyAccess` AWS gestita alle entità. Per maggiori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente di IAM.

Policy (predefinite) gestite da AWS per AWS CodeArtifact

AWS affronta molti casi d'uso comuni fornendo policy IAM autonome create e amministrate da AWS. Queste policy AWS gestite concedono le autorizzazioni necessarie per i casi d'uso comuni, in modo da evitare di dover esaminare quali autorizzazioni sono necessarie. Per ulteriori informazioni, consulta Policy gestite AWS nella Guida per gli utenti di IAM.

Le seguenti politiche AWS gestite, che puoi allegare agli utenti del tuo account, sono specifiche per. AWS CodeArtifact

- `AWSCodeArtifactAdminAccess`— Fornisce l'accesso completo all' CodeArtifact inclusione delle autorizzazioni per amministrare CodeArtifact i domini.

  JSON

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

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

- `AWSCodeArtifactReadOnlyAccess`— Fornisce accesso in sola lettura a. CodeArtifact

  JSON

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

```
    ]
}
```

Per creare e gestire i ruoli CodeArtifact di servizio, è inoltre necessario allegare la policy AWS gestita denominata. `IAMFullAccess`

È inoltre possibile creare policy IAM personalizzate per concedere le autorizzazioni per operazioni e risorse CodeArtifact. Puoi associare queste policy personalizzate agli utenti o ai gruppi IAM che richiedono tali autorizzazioni.

Consenti a un utente di visualizzare le proprie autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
```

```
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Consenti a un utente di ottenere informazioni su repository e domini

La seguente policy consente a un utente o a un ruolo IAM di elencare e descrivere qualsiasi tipo di CodeArtifact risorsa, inclusi domini, repository, pacchetti e asset. La policy include anche l'`codeArtifact:ReadFromRepository`autorizzazione, che consente al principale di recuperare i pacchetti da un repository. CodeArtifact Non consente la creazione di nuovi domini o repository e non consente la pubblicazione di nuovi pacchetti.

Le `sts:GetServiceBearerToken` autorizzazioni `codeartifact:GetAuthorizationToken` e sono necessarie per chiamare l'API. `GetAuthorizationToken`

JSON

```
{
   "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"
```

```
            }
         }
      }
   ]
}
```

Consenti a un utente di ottenere informazioni su domini specifici

Di seguito viene illustrato un esempio di politica di autorizzazioni che consente a un utente di elencare i domini solo nella `us-east-2` regione in cui si riferisce l'account `123456789012` di qualsiasi dominio che inizia con quel nome. `my`

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codeartifact:ListDomains",
         "Resource": "arn:aws:codeartifact:us-east-2:111122223333:domain/my*"

      }
   ]
}
```

Consenti a un utente di ottenere informazioni su repository specifici

Di seguito viene illustrato un esempio di politica di autorizzazioni che consente a un utente di ottenere informazioni sui repository che terminano con `test`, incluse informazioni sui pacchetti in essi contenuti. L'utente non sarà in grado di pubblicare, creare o eliminare risorse.

Le `sts:GetServiceBearerToken` autorizzazioni `codeartifact:GetAuthorizationToken` e sono necessarie per chiamare l'`GetAuthorizationToken`API.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codeartifact:List*",
            "codeartifact:Describe*",
            "codeartifact:Get*",
            "codeartifact:Read*"
         ],
         "Resource": "arn:aws:codeartifact:*:*:repository/*/*test"
      },
      {
         "Effect": "Allow",
         "Action": [
            "codeartifact:List*",
            "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": "*"
      }
   ]
}
```

Limita la durata del token di autorizzazione

Gli utenti devono autenticarsi CodeArtifact con token di autorizzazione per pubblicare o utilizzare le versioni dei pacchetti. I token di autorizzazione sono validi solo per il periodo di vita configurato. I

token hanno una durata predefinita di 12 ore. Per ulteriori informazioni sui token di autorizzazione, vedere. AWS CodeArtifact autenticazione e token

Quando recuperano un token, gli utenti possono configurare la durata del token. I valori validi per la durata di un token di autorizzazione sono `0` e qualsiasi numero compreso tra `900` (15 minuti) e `43200` (12 ore). Un valore di `0` creerà un token con una durata pari alle credenziali temporanee del ruolo dell'utente.

Gli amministratori possono limitare i valori validi per la durata di un token di autorizzazione utilizzando la chiave di `sts:DurationSeconds` condizione nella politica di autorizzazione allegata all'utente o al gruppo. Se l'utente tenta di creare un token di autorizzazione con una durata superiore ai valori validi, la creazione del token avrà esito negativo.

Le politiche di esempio seguenti limitano le possibili durate di un token di autorizzazione creato dagli CodeArtifact utenti.

Politica di esempio: limita la durata del token esattamente a 12 ore (43200 secondi)

Con questa politica, gli utenti potranno creare solo token di autorizzazione con una durata di 12 ore.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codeartifact:*",
            "Resource": "*"
        },
        {
            "Sid": "sts",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "NumericEquals": {
                    "sts:DurationSeconds": 43200
                },
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

Politica di esempio: limita la durata del token tra 15 minuti e 1 ora o uguale al periodo temporaneo delle credenziali dell'utente

Con questa politica, gli utenti saranno in grado di creare token validi tra 15 minuti e 1 ora. Gli utenti potranno anche creare un token che duri per la durata delle credenziali temporanee del loro ruolo `0` specificando for. `--durationSeconds`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codeartifact:*",
            "Resource": "*"
        },
        {
            "Sid": "sts",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "NumericLessThanEquals": {
                    "sts:DurationSeconds": 3600
                },
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

Utilizzo dei tag per controllare l'accesso alle risorse CodeArtifact

Le condizioni nelle istruzioni relative alle policy degli utenti di IAM fanno parte della sintassi utilizzata per specificare le autorizzazioni per le risorse richieste dalle azioni. CodeArtifact L'utilizzo di tag nelle condizioni è un modo per controllare l'accesso alle risorse e alle richieste. Per informazioni sull'etichettatura delle CodeArtifact risorse, consulta. Applicazione di tag alle risorse In questo argomento viene illustrato il controllo degli accessi basato su tag.

Durante la progettazione di policy IAM, potrebbe essere necessario impostare autorizzazioni granulari concedendo l'accesso a risorse specifiche. Poiché il numero di risorse che puoi gestire cresce, questa operazione diventa più difficile. Il tagging delle risorse e l'utilizzo di tag in condizioni di dichiarazione di policy possono semplificare questa attività. Puoi concedere l'accesso in blocco a qualsiasi risorsa con un determinato tag. Quindi applicare ripetutamente questo tag a risorse pertinenti, durante la creazione o in seguito.

I tag possono essere collegati alla risorsa o trasferiti nella richiesta verso servizi che supportano il tagging. In CodeArtifact, le risorse possono avere tag e alcune azioni possono includere tag. Quando si crea una policy IAM, è possibile utilizzare le chiavi di condizione di tag per controllare:

- Quali utenti possono eseguire azioni su un dominio o su una risorsa del repository, in base ai tag già presenti.
- Quali tag possono essere passati in una richiesta di operazione.
- Se delle chiavi di tag specifiche possono essere utilizzate in una richiesta.

Per la sintassi completa e la semantica delle chiavi di condizione di tag, consulta Controllo degli accessi tramite tag nella Guida per l'utente di IAM.

> **Important**
>
> Quando si utilizzano tag sulle risorse per limitare le azioni, i tag devono riferirsi alla risorsa su cui opera l'azione. Ad esempio, per negare `DescribeRepository` le autorizzazioni con i tag, i tag devono trovarsi su ogni repository e non sul dominio. Consulta AWS CodeArtifact riferimento alle autorizzazioni l'elenco delle azioni CodeArtifact e delle risorse su cui operano.

Esempi di controllo degli accessi basato su tag

I seguenti esempi mostrano come specificare le condizioni dei tag nelle policy per gli utenti CodeArtifact .

Example 1: Limitare le operazioni in base ai tag nella richiesta

La politica degli utenti `AWSCodeArtifactAdminAccess` gestiti offre agli utenti il permesso illimitato di eseguire qualsiasi CodeArtifact azione su qualsiasi risorsa.

La seguente politica limita questo potere e nega agli utenti non autorizzati l'autorizzazione a creare repository a meno che la richiesta non contenga determinati tag. Per fare ciò, nega l'`CreateRepository`azione se la richiesta non specifica un tag denominato `costcenter` con uno dei valori o. 1 2 L'amministratore di un cliente deve collegare questa policy IAM a utenti IAM non autorizzati oltre alla policy gestita dall'utente.

JSON

```
{
  "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: Limitare le operazioni in base ai tag delle risorse

La politica degli utenti `AWSCodeArtifactAdminAccess` gestiti offre agli utenti il permesso illimitato di eseguire qualsiasi CodeArtifact azione su qualsiasi risorsa.

La seguente politica limita questo potere e nega agli utenti non autorizzati l'autorizzazione a eseguire azioni sui repository in domini specifici. A tale scopo, rifiuta alcune operazioni se la risorsa ha un tag denominato `Key1` con uno dei valori `Value1` o `Value2`. La chiave di condizione `aws:ResourceTag` viene utilizzata per controllare l'accesso alle risorse in base ai tag su tali risorse. L'amministratore di un cliente deve collegare questa policy IAM a utenti IAM non autorizzati oltre alla policy gestita dall'utente.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codeartifact:TagResource",
        "codeartifact:UntagResource",
        "codeartifact:DescribeDomain",
        "codeartifact:DescribeRepository",
        "codeartifact:PutDomainPermissionsPolicy",
        "codeartifact:PutRepositoryPermissionsPolicy",
        "codeartifact:ListRepositoriesInDomain",
        "codeartifact:UpdateRepository",
        "codeartifact:ReadFromRepository",
        "codeartifact:ListPackages",
        "codeartifact:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/Key1": ["Value1", "Value2"]
        }
      }
    }
  ]
}
```

Example 3: Consenti azioni basate sui tag delle risorse

La seguente politica concede agli utenti il permesso di eseguire azioni e ottenere informazioni su repository e pacchetti in essi contenuti. CodeArtifact

A tal fine, consente azioni specifiche se il repository ha un tag denominato `Key1` con lo stesso valore. `Value1` La chiave di condizione `aws:RequestTag` viene utilizzata per controllare quali tag possono essere passati in una richiesta IAM. La condizione `aws:TagKeys` garantisce che la chiave tag rileva la distinzione tra maiuscole e minuscole. Questa policy è utile per gli utenti IAM che non dispongono della policy gestita dall'utente `AWSCodeArtifactAdminAccess` collegata. La politica gestita offre agli utenti il permesso illimitato di eseguire qualsiasi CodeArtifact azione su qualsiasi risorsa.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact:ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

Example 4: Consenti azioni basate sui tag nella richiesta

La seguente politica concede agli utenti il permesso di creare repository in domini specifici in. CodeArtifact

A tale scopo, consente `TagResource` le azioni `CreateRepository` and se l'API di creazione della risorsa nella richiesta specifica un tag denominato `Key1` con il valore. `Value1` La chiave di condizione `aws:RequestTag` viene utilizzata per controllare quali tag possono essere passati in una richiesta IAM. La condizione `aws:TagKeys` garantisce che la chiave tag rileva la distinzione tra maiuscole e minuscole. Questa policy è utile per gli utenti IAM che non dispongono della policy gestita dall'utente `AWSCodeArtifactAdminAccess` collegata. La politica gestita offre agli utenti il permesso illimitato di eseguire qualsiasi CodeArtifact azione su qualsiasi risorsa.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:CreateRepository",
        "codeartifact:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

AWS CodeArtifact riferimento alle autorizzazioni

AWS CodeArtifact risorse e operazioni

Nel AWS CodeArtifact, la risorsa principale è un dominio. In una policy, devi utilizzare un Amazon Resource Name (ARN) per identificare la risorsa a cui si applica la policy. I repository sono anche

risorse a cui sono ARNs associati. Per ulteriori informazioni, consulta Amazon Resource Names (ARNs) nel Riferimenti generali di Amazon Web Services.

Tipo di risorsa	Formato ARN
Dominio	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:domain/` *`my_domain`*
Repository	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:repository/` *`my_domain`* `/` *`my_repo`*
Gruppo di pacchetti	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:package-group/` *`my_domain`* `/` *`encoded_package_group_pattern`*
Package con un namespace	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:package/` *`my_domain`* `/` *`my_repo`* `/` *`package-format`* `/` *`namespace`* `/` *`my_package`*
Package senza namespace	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:package/` *`my_domain`* `/` *`my_repo`* `/` *`package-format`* `//` *`my_package`*
Tutte le risorse CodeArtifact	`arn:aws:codeartifact:*`
Tutte CodeArtifact le risorse di proprietà dell'account specificato nella regione AWS specificata	`arn:aws:codeartifact:` *`region-ID`* `:` *`account-ID`* `:*`

L'ARN della risorsa specificato dipende dall'azione o dalle azioni a cui si desidera controllare l'accesso.

Puoi indicare un dominio specifico (*`myDomain`*) nella tua dichiarazione utilizzando il relativo ARN come segue.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

Puoi indicare un repository specifico (*myRepo*) nella tua dichiarazione utilizzando il relativo ARN come segue.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

Per specificare più risorse in una singola istruzione, separale ARNs con virgole. La seguente dichiarazione si applica a tutti i pacchetti e i repository in un dominio specifico.

```
"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**
>
> Molti AWS servizi considerano i due punti (:) o una barra (/) come lo stesso carattere in. ARNs Tuttavia, CodeArtifact utilizza una corrispondenza esatta nei modelli e nelle regole delle risorse. Assicurati di utilizzare i caratteri ARN corretti durante la creazione di modelli di eventi, facendo in modo che corrispondano alla sintassi ARN nella risorsa.

AWS CodeArtifact Operazioni e autorizzazioni dell'API

Puoi utilizzare la tabella seguente come riferimento quando configuri il controllo degli accessi e scrivi politiche di autorizzazione da allegare a un'identità IAM (politiche basate sull'identità).

Puoi utilizzare le chiavi di condizione AWS-wide nelle tue AWS CodeArtifact politiche per esprimere condizioni. Per un elenco, consulta IAM JSON Policy Elements Reference nella IAM User Guide.

Puoi specificare le operazioni nel campo `Action` della policy. Per specificare un'operazione, utilizza il prefisso `codeartifact:` seguito dal nome dell'operazione API (ad esempio, `codeartifact:CreateDomain` and `codeartifact:AssociateExternalConnection`). Per specificare più operazioni in una sola istruzione, separa ciascuna di esse con una virgola (ad esempio, `"Action": [ "codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection" ]`).

Utilizzo di caratteri jolly

Puoi specificare un ARN, con o senza un carattere jolly (*), come valore della risorsa nel campo `Resource` della policy. È possibile utilizzare un carattere jolly per specificare più operazioni o risorse. Ad esempio, `codeartifact:*` specifica tutte le CodeArtifact azioni e `codeartifact:Describe*` specifica tutte le CodeArtifact azioni che iniziano con la parola. `Describe`

Gruppo di pacchetti ARNs

> **Note**
>
> Questa sezione spiega come la codifica dei gruppi di pacchetti ARNs e dei pattern sia informativa. Si consiglia di copiare ARNs dalla console o eseguire il recupero ARNs utilizzando l'`DescribePackageGroup`API invece di codificare modelli e costruire. ARNs

Le policy IAM utilizzano il carattere jolly,*, per abbinare più azioni IAM o più risorse. Anche i modelli di gruppi di pacchetti utilizzano il * carattere. Per scrivere più facilmente policy IAM che corrispondano a un singolo gruppo di pacchetti, il formato ARN del gruppo di pacchetti utilizza una versione codificata del pattern del gruppo di pacchetti.

In particolare, il formato ARN del gruppo di pacchetti è il seguente:

```
arn:aws:codeartifact:region:account-ID:package-
group/my_domain/encoded_package_group_pattern
```

Dove lo schema codificato del gruppo di pacchetti è lo schema del gruppo di pacchetti, con alcuni caratteri speciali sostituiti con i relativi valori codificati in percentuale. L'elenco seguente contiene i caratteri e i corrispondenti valori codificati in percentuale:

- * : %2a
- $: %24
- % : %25

Ad esempio, l'ARN per un gruppo di pacchetti root di un dominio, (/*), sarebbe:

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

Nota che i caratteri non inclusi nell'elenco non possono essere codificati e ARNs fanno distinzione tra maiuscole e minuscole, quindi `*` devono essere codificati come e non. `%2a` `%2A`

Risoluzione dei problemi relativi a identità e accesso AWS CodeArtifact

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un CodeArtifact IAM.

Argomenti

- Non sono autorizzato a eseguire alcuna azione in CodeArtifact
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie CodeArtifact risorse

Non sono autorizzato a eseguire alcuna azione in CodeArtifact

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `codeartifact:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codeartifact:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `codeartifact:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie CodeArtifact risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se CodeArtifact supporta queste funzionalità, consulta. Come AWS CodeArtifact funziona con IAM
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Utilizzo degli endpoint Amazon VPC

Puoi configurare l'utilizzo CodeArtifact di un endpoint VPC (Virtual Private Cloud) con interfaccia per migliorare la sicurezza del tuo VPC.

Gli endpoint VPC utilizzano AWS PrivateLink, un servizio che consente l'accesso CodeArtifact APIs tramite indirizzi IP privati. AWS PrivateLink limita tutto il traffico di rete tra il tuo VPC CodeArtifact e la rete AWS. Quando si utilizza un endpoint VPC con interfaccia, non è necessario un gateway Internet, un dispositivo NAT o un gateway privato virtuale. Per ulteriori informazioni, consulta Endpoint VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Important

- Gli endpoint VPC non supportano le richieste interregionali.AWS Assicurati di creare l'endpoint nella stessa AWS regione verso cui intendi effettuare le chiamate API. CodeArtifact
- Gli endpoint VPC supportano solo il DNS fornito da Amazon tramite Amazon Route 53. Se si desidera utilizzare il proprio DNS, è possibile usare l'inoltro condizionale sul DNS. Per ulteriori informazioni, consulta DHCP Option Sets nella Amazon Virtual Private Cloud User Guide.
- Il gruppo di sicurezza collegato all'endpoint VPC deve consentire le connessioni in entrata sulla porta 443 dalla sottorete privata del VPC.

Argomenti

- Crea endpoint VPC per CodeArtifact
- Creare l'endpoint gateway Amazon S3
- Utilizzo CodeArtifact da un VPC
- Creazione di una policy di endpoint VPC per CodeArtifact.

Crea endpoint VPC per CodeArtifact

Per creare endpoint di cloud privato virtuale (VPC) per CodeArtifact, usa il comando Amazon. EC2 `create-vpc-endpoint` AWS CLI Per ulteriori informazioni, consulta Interface VPC Endpoints (AWS PrivateLink) nella Amazon Virtual Private Cloud User Guide.

Sono necessari due endpoint VPC in modo che tutte le richieste si trovino nella CodeArtifact rete. AWS Il primo endpoint viene utilizzato per chiamare CodeArtifact APIs (ad esempio, `GetAuthorizationToken` and). `CreateRepository`

```
com.amazonaws.region.codeartifact.api
```

Il secondo endpoint viene utilizzato per accedere ai CodeArtifact repository utilizzando gestori di pacchetti e strumenti di compilazione (ad esempio, npm e Gradle).

```
com.amazonaws.region.codeartifact.repositories
```

Il comando seguente crea un endpoint per accedere ai repository. 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
```

Il comando seguente crea un endpoint per accedere ai gestori di pacchetti e agli strumenti di compilazione.

```
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**
>
> Quando si crea un `codeartifact.repositories` endpoint, è necessario creare un hostname DNS privato utilizzando l'opzione. `--private-dns-enabled` Se non puoi o non vuoi creare un nome host DNS privato quando crei l'`codeartifact.repositories`endpoint, devi seguire un passaggio di configurazione aggiuntivo per utilizzare il tuo gestore di pacchetti da CodeArtifact un VPC. Per ulteriori informazioni, consulta Usa l'`codeartifact.repositories`endpoint senza DNS privato.

Dopo aver creato gli endpoint VPC, potrebbe essere necessario eseguire ulteriori configurazioni con le regole del gruppo di sicurezza con cui utilizzare gli endpoint. CodeArtifact Per ulteriori informazioni sui gruppi di sicurezza in Amazon VPC, consulta Gruppi di sicurezza.

Se riscontri problemi di connessione a CodeArtifact, puoi utilizzare lo strumento VPC Reachability Analyzer per eseguire il debug del problema. Per ulteriori informazioni, consulta Cos'è VPC Reachability Analyzer?

Creare l'endpoint gateway Amazon S3

CodeArtifact utilizza Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) per archiviare gli asset dei pacchetti. Per estrarre i pacchetti CodeArtifact, devi creare un endpoint gateway per Amazon S3. Quando il processo di compilazione o distribuzione scarica pacchetti da CodeArtifact, deve accedere CodeArtifact per ottenere i metadati del pacchetto e Amazon S3 per scaricare le risorse del pacchetto (ad esempio, i file `.jar` Maven).

> **Note**
>
> Non è necessario un endpoint Amazon S3 quando si utilizzano i formati di pacchetto Python o Swift.

Per creare l'endpoint gateway Amazon S3 per CodeArtifact, usa il comando Amazon. EC2 `create-vpc-endpoint` AWS CLI Quando crei l'endpoint, devi selezionare le tabelle di routing per il tuo VPC. Per ulteriori informazioni, consulta Gateway VPC Endpoints nella Amazon Virtual Private Cloud User Guide.

Il comando seguente crea un endpoint Amazon S3.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
  --route-table-ids routetableid
```

Autorizzazioni minime per i bucket Amazon S3 per AWS CodeArtifact

L'endpoint del gateway Amazon S3 usa un documento di policy IAM per limitare l'accesso al servizio. Per consentire solo le autorizzazioni minime per il bucket Amazon S3 CodeArtifact, limita l'accesso al bucket Amazon S3 CodeArtifact utilizzato quando crei il documento di policy IAM per l'endpoint.

La tabella seguente descrive i bucket Amazon S3 a cui dovresti fare riferimento nelle tue politiche per consentire l'accesso CodeArtifact in ciascuna regione.

Regione	ARN per bucket Amazon S3
us-east-1	arn:aws:s3: :assets-193858265520-us-east-1
us-east-2	arn:aws:s3: :assets-250872398865-us-east-2
us-west-2	arn:aws:s3: :assets-787052242323-us-west-2
eu-west-1	arn:aws:s3: :assets-438097961670-eu-west-1
eu-west-2	arn:aws:s3: :assets-247805302724-eu-west-2
eu-west-3	arn:aws:s3: ::assets-762466490029-eu-west-3
eu-north-1	arn:aws:s3: :assets-611884512288-eu-nord-1
eu-south-1	arn:aws:s3: :assets-484130244270-eu-sud-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

Puoi usare il `aws codeartifact describe-domain` comando per recuperare il bucket Amazon S3 usato da un dominio. CodeArtifact

```
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"
  }
}
```

Esempio

L'esempio seguente illustra come fornire l'accesso ai bucket Amazon S3 necessari CodeArtifact per le operazioni nella regione. `us-east-1` Per le altre regioni, aggiorna la `Resource` voce con l'ARN di autorizzazione corretto per la tua regione in base alla tabella precedente.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]
    }
  ]
}
```

Utilizzo CodeArtifact da un VPC

Se non puoi o non vuoi abilitare il DNS privato sull'endpoint `com.amazonaws.region.codeartifact.repositories` VPC Crea endpoint VPC per CodeArtifact in cui hai creato, devi utilizzare una configurazione diversa per l'endpoint dei repository da utilizzare da un VPC. CodeArtifact Segui le istruzioni Usa l'`codeartifact.repositories`endpoint senza DNS privato per configurare CodeArtifact se

sull'com.amazonaws.*region*.codeartifact.repositoriesendpoint non è abilitato il DNS privato.

Usa l'**codeartifact.repositories**endpoint senza DNS privato

Se non puoi o non vuoi abilitare il DNS privato sull'endpoint `com.amazonaws.`*region*`.codeartifact.repositories` VPC Crea endpoint VPC per CodeArtifact in cui hai creato, devi seguire queste istruzioni per configurare il tuo gestore di pacchetti con l'URL corretto. CodeArtifact

1. Esegui il comando seguente per trovare un endpoint VPC da utilizzare per sovrascrivere il nome host.

   ```
   $ aws ec2 describe-vpc-endpoints --filters Name=service-
   name,Values=com.amazonaws.region.codeartifact.repositories \
     --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
   ```

   L'output sarà simile al seguente.

   ```
   [
     [
       "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"
     ]
   ]
   ```

2. Aggiorna il percorso dell'endpoint VPC per includere il formato del pacchetto, il nome di CodeArtifact dominio e CodeArtifact il nome del repository. Guarda l'esempio seguente.

   ```
   https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-
   west-2.vpce.amazonaws.com/format/d/domain_name-domain_owner/repo_name
   ```

   Sostituisci i seguenti campi dall'endpoint di esempio.

   - *`format`*: Sostituisci con un formato di CodeArtifact pacchetto valido, ad esempio `npm` o`pypi`.
   - *`domain_name`*: Sostituisci con il CodeArtifact dominio che contiene il CodeArtifact repository che ospita i pacchetti.
   - *`domain_owner`*: Sostituisci con l'ID del proprietario del CodeArtifact dominio, `111122223333` ad esempio.
   - *`repo_name`*: Sostituiscilo con il CodeArtifact repository che ospita i pacchetti.

L'URL seguente è un esempio di endpoint del repository npm.

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. Configura il tuo gestore di pacchetti per utilizzare l'endpoint VPC aggiornato del passaggio precedente. È necessario configurare il gestore di pacchetti senza utilizzare il CodeArtifact `login` comando. Per le istruzioni di configurazione per ogni formato di pacchetto, consultate la seguente documentazione.

   - npm: Configurazione di npm senza utilizzare il comando login
   - nuget: configura nuget o dotnet senza il comando di login
   - pip: Configura pip senza il comando login
   - spago: Configura e usa twine con CodeArtifact
   - Gradle: Uso CodeArtifact con Gradle
   - mvn: Usare CodeArtifact con mvn

Creazione di una policy di endpoint VPC per CodeArtifact.

Per creare una policy per gli endpoint VPC per CodeArtifact, specifica quanto segue:

- Il principale che può eseguire azioni.
- Le operazioni che possono essere eseguite.
- Le risorse su cui è possibile eseguire le operazioni.

La seguente politica di esempio specifica che i principali dell'account 123456789012 possono chiamare l'API e recuperare i `GetAuthorizationToken` pacchetti da un repository. CodeArtifact

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
```

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

Creare CodeArtifact risorse con AWS CloudFormation

CodeArtifact è integrato con AWS CloudFormation, un servizio che consente di modellare e configurare le AWS risorse in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Crei un modello che descrive tutte le AWS risorse che desideri e si CloudFormation occupa del provisioning e della configurazione di tali risorse per te.

Quando lo utilizzi CloudFormation, puoi riutilizzare il modello per configurare le CodeArtifact risorse in modo coerente e ripetuto. Descrivi le tue risorse una sola volta e poi fornisci le stesse risorse più e più volte in più account e AWS regioni.

CodeArtifact e CloudFormation modelli

Per fornire e configurare le risorse CodeArtifact e i servizi correlati, è necessario conoscere CloudFormation i modelli. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse che desideri fornire negli CloudFormation stack. Se non conosci JSON o YAML, puoi usare CloudFormation Designer per iniziare a usare i modelli. CloudFormation Per ulteriori informazioni, consulta Cos'è AWS CloudFormation Designer? nella Guida AWS CloudFormation per l'utente.

CodeArtifact supporta la creazione di domini, repository e gruppi di pacchetti in. CloudFormationPer ulteriori informazioni, inclusi esempi di modelli JSON e YAML, consulta i seguenti argomenti nella Guida per l'utente:CloudFormation

- AWS::CodeArtifact::Domain
- AWS::CodeArtifact::Repository
- AWS::CodeArtifact::PackageGroup

Prevenzione dell'eliminazione delle risorse CodeArtifact

CodeArtifact i repository contengono dipendenze critiche tra le applicazioni che potrebbero non essere facili da ricreare in caso di perdita. Per proteggere CodeArtifact le risorse dall'eliminazione accidentale durante la gestione CodeArtifact delle risorse con CloudFormation, includi `UpdateRetainPolicy` gli attributi `DeletionPolicy` and con un valore di `Retain` su tutti i domini e i repository. Ciò impedirà l'eliminazione se la risorsa viene rimossa dal modello dello stack o se

l'intero stack viene eliminato accidentalmente. Il seguente frammento di codice YAML mostra un dominio e un repository di base con questi attributi:

```
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
```

Per ulteriori informazioni su questi attributi, consulta DeletionPolicye UpdateReplacePolicynella Guida per l'utente.AWS CloudFormation

Scopri di più su CloudFormation

Per ulteriori informazioni CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- AWS CloudFormation Guida per l'utente
- AWS CloudFormation Guida per l'utente dell'interfaccia a riga di comando

Risoluzione dei problemi AWS CodeArtifact

Le seguenti informazioni potrebbero aiutarti a risolvere i problemi più comuni con. CodeArtifact

Per informazioni sulla risoluzione di problemi specifici relativi ai formati, consultate i seguenti argomenti:

- Risoluzione dei problemi con Maven
- Risoluzione dei problemi Swift

Non riesco a visualizzare le notifiche

Problema: Quando ti trovi nella console Strumenti di sviluppo e selezioni Notifications (Notifiche) in Settings (Impostazioni), viene visualizzato un errore di autorizzazione.

Possibili correzioni: sebbene le notifiche siano una funzionalità della console Developer Tools, attualmente CodeArtifact non supporta le notifiche. Nessuna delle politiche gestite CodeArtifact include autorizzazioni che consentono agli utenti di visualizzare o gestire le notifiche. Se utilizzi altri servizi nella console Developer Tools e tali servizi supportano le notifiche, le politiche gestite per tali servizi includono le autorizzazioni necessarie per visualizzare e gestire le notifiche per tali servizi.

Applicazione di tag alle risorse

Un tag è un'etichetta di attributo personalizzata che l'utente o AWS assegna a una AWS risorsa. Ogni AWS tag è composto da due parti:

- Una chiave di tag (ad esempio, `CostCenter`, `Environment`, `Project` o `Secret`). Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole.
- Un campo facoltativo noto come valore del tag (ad esempio, `111122223333`, `Production` o un nome di team). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Analogamente alle chiavi dei tag, i valori dei tag prevedono una distinzione tra lettere maiuscole e minuscole.

Tutti questi sono noti come coppie chiave-valore.

I tag ti aiutano a identificare e organizzare AWS le tue risorse. Molti servizi AWS supportano il tagging, perciò è possibile assegnare lo stesso tag a risorse di diversi servizi per indicare che le risorse sono correlate. Ad esempio, puoi assegnare a un repository lo stesso tag che assegni a un progetto. AWS CodeBuild

Per suggerimenti e best practice per l'utilizzo dei tag, consultate il white paper sulle migliori pratiche per l' AWS etichettatura delle risorse.

È possibile aggiungere un tag ai seguenti tipi di risorse in CodeArtifact:

- Aggiungi un tag a un repository CodeArtifact
- Aggiungi un tag a un dominio in CodeArtifact

Puoi usare la console, AWS CLI, CodeArtifact APIs o per: AWS SDKs

- Aggiungi tag a un dominio o a un repository al momento della creazione*.
- Aggiungi, gestisci e rimuovi i tag per un dominio o un repository.

* Non è possibile aggiungere tag a un dominio o a un repository quando lo si crea nella console.

Oltre a identificare, organizzare e tracciare la risorsa con i tag, puoi utilizzare i tag nelle policy IAM per controllare chi può visualizzare e interagire con la tua risorsa. Per esempi di policy di accesso basate su tag, consulta Utilizzo dei tag per controllare l'accesso alle risorse CodeArtifact.

CodeArtifact allocazione dei costi con tag

È possibile utilizzare i tag per allocare sia i costi di archiviazione che quelli di richiesta. CodeArtifact

Allocazione dei costi di archiviazione dei dati in CodeArtifact

I costi di archiviazione dei dati sono legati ai domini, pertanto per allocare i costi di CodeArtifact archiviazione, puoi utilizzare qualsiasi tag applicato ai tuoi domini. Per informazioni sull'aggiunta di tag ai domini, consulta. Aggiungi un tag a un dominio in CodeArtifact

Allocazione dei costi delle richieste in CodeArtifact

La maggior parte dell'utilizzo delle richieste è legato ai repository, pertanto per allocare i costi CodeArtifact delle richieste, è possibile utilizzare qualsiasi tag applicato ai repository. Per informazioni sull'aggiunta di tag ai repository, consulta. Aggiungi un tag a un repository CodeArtifact

Alcuni tipi di richieste sono associati ai domini anziché ai repository, pertanto l'utilizzo delle richieste e i costi relativi alle richieste verranno allocati ai tag del dominio. Il modo migliore per determinare se un tipo di richiesta è associato a un dominio o a un repository consiste nell'utilizzare le azioni definite dalla AWS CodeArtifact tabella nel Service Authorization Reference. Trova il tipo di richiesta nella colonna Azioni e guarda il valore nella colonna Tipi di risorse corrispondente. Se il tipo di risorsa è dominio, le richieste di quel tipo verranno fatturate al dominio. Se il tipo di risorsa è repository o pacchetto, le richieste di quel tipo verranno fatturate al repository. Alcune azioni mostrano entrambi i tipi di risorse, per tali azioni la risorsa fatturata dipende dal valore passato nella richiesta.

Quote in AWS CodeArtifact

La tabella seguente descrive le quote di risorse in. CodeArtifact Per visualizzare le quote di risorse insieme all'elenco degli endpoint di servizio per CodeArtifact, consulta le quote di AWS servizio in. Riferimenti generali di Amazon Web Services

È possibile richiedere un aumento della quota di servizio per le seguenti CodeArtifact quote di risorse. Per ulteriori informazioni sulla richiesta di un aumento della quota di servizio, vedere AWS Service Quotas.

Name	Predefinita	Adattabile	Description
Dimensione del file degli asset	Ogni regione supportata: 5 GB	Sì	La dimensione massima del file per risorsa.
Risorse per versione del pacchetto	Ogni regione supportata: 150	No	Il numero massimo di risorse per versione del pacchetto.
CopyPackageVersions richieste al secondo	Ogni regione supportata: 5	Sì	Il numero massimo di chiamate che è possibile effettuare CopyPackageVersions al secondo.
Upstream diretti per repository	Ogni regione supportata: 10	No	Il numero massimo di repository upstream diretti per repository.
Domini per account AWS	Ogni regione supportata: 10	Sì	Il numero massimo di domini che possono essere creati per account. AWS
GetAuthorizationToken richieste al secondo	Ogni regione supportata: 40	Sì	Il numero massimo di token di autorizzazione recuperati al secondo.

Name	Predefinita	Adattabile	Description
GetPackageVersionAsset richieste al secondo	Ogni Regione supportata: 50	Sì	Il numero massimo di chiamate che è possibile effettuare GetPackageVersionAsset al secondo.
ListPackageVersionAssets richieste al secondo	Ogni Regione supportata: 200	Sì	Il numero massimo di chiamate che è possibile effettuare ListPackageVersionAssets al secondo.
ListPackageVersions richieste al secondo	Ogni Regione supportata: 200	Sì	Il numero massimo di chiamate che è possibile effettuare ListPackageVersions al secondo.
ListPackages richieste al secondo	Ogni Regione supportata: 200	Sì	Il numero massimo di chiamate che è possibile effettuare ListPackages al secondo.
PublishPackageVersion richieste al secondo	Ogni regione supportata: 10	Sì	Il numero massimo di chiamate che è possibile effettuare PublishPackageVersion al secondo.
Leggi le richieste al secondo da un singolo AWS account	Ogni regione supportata: 800	Sì	Il numero massimo di richieste di lettura da un AWS account al secondo.
Repository per dominio	Ogni regione supportata: 1.000	Sì	Il numero massimo di repository che puoi creare per dominio.

Name	Predefinita	Adatta e	Description
Richieste al secondo utilizzando un singolo token di autenticazione	Ogni regione supportata: 1.200	No	Il numero massimo di richieste al secondo tramite un singolo token di autenticazione.
Richieste senza token di autenticazione per indirizzo IP	Ogni regione supportata: 600	No	Il numero massimo di richieste al secondo senza token di autentica zione da parte di un singolo indirizzo IP.
Ricerca negli archivi upstream	Ogni regione supportata: 25	No	Il numero massimo di repository upstream in cui effettuare la ricerca durante la risoluzione di un pacchetto.
Richieste di scrittura al secondo da un singolo account AWS	Ogni regione supportata: 100	Sì	Il numero massimo di richieste di scrittura da un AWS account al secondo.

Note

In generale, ogni richiesta di lettura effettuata viene CodeArtifact conteggiata come una richiesta conteggiata ai fini di una quota. Tuttavia, per il formato del pacchetto Ruby, una singola richiesta di lettura all'`/api/v1/dependencies`operazione può richiedere dati su più pacchetti.

Ad esempio, la richiesta può essere simile `https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1,gem2.gem3`a. In questo esempio, la richiesta conta come tre richieste rispetto alla quota.

Tieni presente che le richieste multiple si applicano solo alle quote di servizio, non alla fatturazione. Nell'esempio, ti verrà addebitata una sola richiesta, anche se per la quota di servizio viene conteggiata come tre richieste.

AWS CodeArtifact cronologia dei documenti della guida per l'utente

La tabella seguente descrive le modifiche importanti alla documentazione per CodeArtifact.

Modifica	Descrizione	Data
È stata aggiunta documentazione per la configurazione e l'utilizzo di Cargo con CodeArtifact	CodeArtifact ora supporta le casse Cargo. È stata aggiunta documentazione con indicazioni sulla configurazione di Cargo per l'utilizzo CodeArtifact dei repository. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con Cargo.	20 giugno 2024
È stata aggiunta documentazione per la configurazione e l'utilizzo di Ruby con CodeArtifact	CodeArtifact ora supporta le gemme Ruby. È stata aggiunta documentazione con indicazioni sulla configurazione dei gestori di pacchetti Ruby per l'utilizzo dei repository. CodeArtifact Per ulteriori informazioni, consulta Usare CodeArtifact con Ruby.	30 aprile 2024
È stato aggiunto un esempio di politica chiave per la creazione di domini con una chiave gestita dal cliente AWS KMS	È stato aggiunto un esempio di policy chiave che può essere utilizzata per creare una chiave KMS gestita dal cliente per crittografare le risorse nei domini. CodeArtifact Per ulteriori informazioni, consulta Esempio di politica AWS KMS chiave.	18 aprile 2024

È stata aggiunta documentazione per supportare il lancio di gruppi di pacchetti.	È stata aggiunta documentazione sulla gestione e l'utilizzo dei gruppi di pacchetti in CodeArtifact. Per ulteriori informazioni, consulta Lavorare con i gruppi di pacchetti in CodeArtifact.	21 marzo 2024
Sono stati aggiunti altri gestori di pacchetti validi alla documentazione sul comando aws codeartifact login.	`dotnet`Aggiunto `nuget` e `swift` all'elenco dei gestori di pacchetti validi da utilizzare con il comando. `aws codeartifact login` Per ulteriori informazioni, consulta AWS CodeArtifact autenticazione e token.	18 febbraio 2024
È stata aggiunta una voce alla documentazione sulla risoluzione dei problemi di Swift relativa alla sospensione di Xcode sulle macchine CI	Sono state aggiunte informazioni, inclusa una soluzione, su un problema che può causare il blocco di Xcode sulle macchine CI a causa della richiesta di password nel portachiavi. Per ulteriori informazioni, consulta Xcode si blocca sulla macchina CI a causa della richiesta di password del portachiavi.	6 febbraio 2024

Sono state aggiunte informazioni sulla risoluzione dei problemi relativi ai tempi di installazione lenti dei pacchetti npm con npm 8.x o versioni successive	Sono state aggiunte informazioni su come aggirare i tempi di installazione dei pacchetti npm lenti CodeArtifact, che potrebbero causare tempi di compilazione lenti. Per ulteriori informazioni, consulta Risoluzione dei problemi di installazioni lente con npm 8.x o versioni successive.	29 dicembre 2023
Informazioni aggiornate sulle risorse del pacchetto Python e sul comportamento dei metadati in CodeArtifact	Informazioni aggiornate su come i CodeArtifact repository conservano e aggiornano le risorse e i metadati delle versioni dei pacchetti Python. Per ulteriori informazioni, consulta Richiesta di pacchetti Python da upstream e connessioni esterne.	14 dicembre 2023
Documentazione riorganizzata sul monitoraggio CodeArtifact	Informazioni riorganizzate sul monitoraggio CodeArtifact degli eventi e aggiunte informazioni sulla visualizzazione delle CodeArtifact richieste con i CloudWatch parametri di Amazon. Per ulteriori informazioni, consulta Monitoraggio CodeArtifact.	14 dicembre 2023

Sono state aggiunte ulteriori informazioni sulla gestione delle CodeArtifact risorse con CloudFormation	Sono stati aggiunti riferimenti e collegamenti alla documentazione sulla gestione CodeArtifact delle risorse con CloudFormation, inclusa una sezione sulla prevenzione dell'eliminazione delle CodeArtifact risorse gestite con CloudFormation. Per ulteriori informazioni, consulta Prevenzione dell'eliminazione delle risorse CodeArtifact .	7 dicembre 2023
È stata aggiunta una documentazione che descrive in dettaglio CodeArtifact il supporto di AWS KMS External Key Stores (XKS)	È stata aggiunta una sezione con informazioni sul supporto CodeArtifact delle chiavi KMS, incluso l'utilizzo delle chiavi XKS con. CodeArtifact Per ulteriori informazioni, consulta Tipi di AWS KMS chiavi supportati in CodeArtifact.	31 ottobre 2023
Documentazione per la risoluzione dei problemi aggiornata, esistente e aggiunta di nuova	È stato aggiunto un argomento sulla risoluzione dei problemi di Maven e sono stati inclusi collegamenti alla documentazione sulla risoluzione dei problemi di Swift e Maven nell'argomento generale sulla risoluzione dei problemi. Per ulteriori informazioni, consulta Risoluzione dei problemi AWS CodeArtifact.	28 settembre 2023

Documentazione aggiornata per includere il comando publish di Swift Package Manager	Swift 5.9 ha introdotto un `swift package-registry publish` comando per creare e pubblicare un pacchetto Swift in un repository di pacchetti. Ha aggiornato la documentazione di Swift per includere istruzioni per l'uso di quel comando. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con Swift.	25 settembre 2023
È stata aggiunta la documentazione per la configurazione CodeArtifact con Swift	CodeArtifact ora supporta i pacchetti Swift. È stata aggiunta documentazione con indicazioni sulla configurazione di Swift per l'utilizzo dei repository. CodeArtifact Per ulteriori informazioni, consulta Utilizzo CodeArtifact con Swift.	20 settembre 2023
Aggiunta una guida su come CodeArtifact gestisce le versioni dei pacchetti Python modificate	È stata aggiunta documentazione con informazioni su come stabilire se una versione di pacchetto Python è stata rimossa, come CodeArtifact gestire le versioni di pacchetto rimosse e risposte a domande comuni. Per ulteriori informazioni, consulta Versioni del pacchetto modificate.	2 agosto 2023

È stato corretto un comando da riga di comando errato nella documentazione di Yarn	È stato corretto un comando da riga di comando errato che recupera un token di CodeArtifact autorizzazione e lo memorizza in una variabile di ambiente nella documenta zione di Yarn.	20 luglio 2023
Aggiunte minori e piccole correzioni di bug nella documentazione di Python	Sono state aggiunte informazi oni su pip e twine nella rispettiva documentazione e corretto ciò che accade quando si utilizza il comando con twine. `codeartifact login` Per ulteriori informazi oni, consulta Configura e usa pip con CodeArtifact e Configura e usa twine con CodeArtifact.	14 luglio 2023
Sono stati corretti i comandi dotnet errati nella documenta zione CodeBuild	Sono stati corretti i `dotnet add package` comandi nella Utilizzo di pacchetti in NuGet CodeBuild documentazione.	13 luglio 2023
Documentazione AWS CodeArtifact e AWS Identity and Access Management aggiornamento	La CodeArtifact documenta zione IAM è stata revisionata per aggiungere chiarezza e coerenza con la documenta zione di altri servizi. AWS Per informazioni, consulta Identity and Access Management per AWS CodeArtifact.	24 maggio 2023

Aggiunte informazioni sulle versioni dei pacchetti Python cancellati	Sono state aggiunte informazioni su come CodeArtifact conservare i metadati della versione del pacchetto Python rimosso. Per ulteriori informazioni, vedere. Versioni del pacchetto modificate	11 aprile 2023
Sono state aggiunte informazioni sul supporto Clojure	Sono state aggiunte informazioni sul supporto Clojure, inclusa la gestione delle dipendenze per i progetti Clojure. Per ulteriori informazioni, consulta Utilizzare CodeArtifact con deps.edn.	21 marzo 2023
Sono state aggiunte informazioni sulla pubblicazione di pacchetti generici	Sono state aggiunte informazioni sui pacchetti generici e su come pubblicare e scaricare il contenuto del pacchetto con AWS CLI. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con pacchetti generici, Pubblicazione e utilizzo di pacchetti generici e Comandi supportati per pacchetti generici.	10 marzo 2023
Sono state aggiunte informazioni sui limiti delle dimensioni delle risorse per la pubblicazione	È stata aggiunta una sezione a Package publishing per spiegare i limiti delle dimensioni degli asset per la pubblicazione.	21 giugno 2022

È stata rifattorizzata la documentazione relativa alla connessione esterna	La documentazione relativa alla connessione esterna è stata spostata e l'ha riorganizzata in modo da concentrarsi sull'obiettivo finale dell'utente, ossia connettere il proprio CodeArtifact repository agli archivi pubblici di pacchetti. Sono state inoltre aggiunte ulteriori indicazioni e informazioni sui diversi metodi per raggiungere tale obiettivo. Per ulteriori informazioni, consulta Connect un CodeArtifact repository a un repository pubblico.	9 maggio 2022
Sono state aggiornate le informazioni sugli CodeArtifact eventi per Amazon CloudWatch Events	Sono state aggiunte ulteriori informazioni al `account` campo e aggiunto il `repositoryAdministrator` campo. Per ulteriori informazioni, consulta CodeArtifact formato ed esempio dell'evento.	7 marzo 2022

Sono state aggiunte istruzioni di configurazione per l'utilizzo CodeArtifact da un VPC senza DNS privato	Se non puoi o non vuoi abilitare il DNS privato sul tuo endpoint `codeartifact.repositories` VPC, devi utilizzare una configurazione diversa per l'endpoint dei repository da utilizzare da un VPC. CodeArtifact Per ulteriori informazioni, consulta Usa l'`codeartifact.repositories` endpoint senza DNS privato.	8 febbraio 2022
È stata aggiunta una documentazione approfondita per l'aggiornamento dello stato delle versioni dei pacchetti	La documentazione sullo stato della versione del pacchetto di aggiornamento è stata ampliata in un argomento a sé stante. È stata aggiunta la documentazione per l'aggiornamento dello stato di una versione del pacchetto, incluse le autorizzazioni IAM richieste, AWS CLI comandi di esempio per vari scenari e possibili errori. Per ulteriori informazioni, consulta Aggiorna lo stato della versione del pacchetto.	1 settembre 2021

È stata aggiornata la documentazione sulle versioni del pacchetto di copia con informazioni più approfondite sulle autorizzazioni	Sono state aggiunte ulteriori informazioni sulle autorizzazioni IAM e basate sulle politiche basate sulle risorse necessarie per richiamare il `aws codeartifact copy-package-versions` comando per copiare le versioni dei pacchetti da un repository a un altro all'interno dello stesso dominio in. CodeArtifact Oltre a ulteriori informazioni, ora sono disponibili esempi delle politiche basate sulle risorse richieste per il repository di origine e di destinazione. Per ulteriori informazioni, consulta Autorizzazioni IAM richieste per copiare i pacchetti.	25 agosto 2021

Documentazione aggiornata per l'esecuzione di una build Gradle in IntelliJ IDEA	È stata aggiornata la documentazione per l'esecuzione di una build Gradle in IntelliJ IDEA con i passaggi per configurare Gradle da cui recuperare i plugin. CodeArtifact È stata inoltre aggiunta un'opzione per creare un nuovo token di CodeArtifact autorizzazione per ogni nuova esecuzione con una chiamata in linea a. `aws codeartifact get-authorization-token` Per ulteriori informazioni, consulta Esegui una build Gradle in IntelliJ IDEA.	23 agosto 2021
Aggiunta documentazione per la configurazione e l'utilizzo di Yarn con AWS CodeArtifact	Aggiunta documentazione per la configurazione e l'utilizzo di Yarn 1.X e Yarn 2.X con cui gestire i pacchetti npm. CodeArtifact Per ulteriori informazioni, consulta Configura e usa Yarn con CodeArtifact.	30 luglio 2021

AWS CodeArtifact NuGet ora supporta i pacchetti	CodeArtifact gli utenti possono ora pubblicare e utilizzar e NuGet pacchetti. È stata aggiunta documentazione per la configurazione e l'utilizzo di Visual Studio e strumenti da riga di NuGet comando come `nuget` e `dotnet` con i CodeArtifact repository. Per ulteriori informazioni, consulta Utilizzo CodeArtifact con NuGet.	19 novembre 2020
Taggare le risorse in AWS CodeArtifact	È stata aggiunta documenta zione sull'etichettatura di repository e domini. AWS CodeArtifact Per informazioni, consulta Applicazione di tag alle risorse .	30 ottobre 2020
CodeArtifact ora supporta CloudFormation	CodeArtifact gli utenti possono ora utilizzare CloudFormation i modelli per creare CodeArtif act repository e domini. Vedi Creare CodeArtifact risorse con AWS CloudFormation per ulteriori informazioni e per iniziare.	8 ottobre 2020

Aggiungi informazioni sulla creazione di endpoint gateway Amazon S3 da utilizzare con CodeArtifact Amazon VPC	Sono state aggiunte informazioni sulla creazione di endpoint gateway Amazon S3 con il comando Amazon. EC2 AWS CLI Questa documentazione contiene anche informazioni sulle autorizzazioni specifiche CodeArtifact necessarie per essere utilizzate con gli ambienti Amazon VPC. Per informazioni, consulta Creare l'endpoint gateway Amazon S3.	12 agosto 2020
Pubblicazione di artefatti Maven con curl e pubblicazione di artefatti Maven di terze parti	È stata aggiunta Pubblicazione con curl Pubblica artefatti di terze parti una guida per e.	10 agosto 2020
Versione General Availability (GA)	Versione iniziale della Guida CodeArtifact per l'utente.	10 giugno 2020

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.