CodeArtifact 使用者指南

CodeArtifact

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

CodeArtifact: CodeArtifact 使用者指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS CodeArtifact？ 1
CodeArtifact 如何運作？ 1
概念 2
資產 2
網域 2
儲存庫 3
套件 3
套件群組 3
套件命名空間 3
套件版本 3
套件版本修訂 4
上游儲存庫 4
如何開始使用 CodeArtifact？ 4
設定 5
註冊 AWS 5
安裝或升級 ，然後設定 AWS CLI 6
佈建 IAM 使用者 7
安裝您的套件管理員或建置工具 8
後續步驟 8
開始使用 10
先決條件 10
開始使用主控台 10
開始使用 AWS CLI 13
使用儲存庫 20
建立 儲存庫 20
建立儲存庫 （主控台） 21
建立儲存庫 (AWS CLI) 22
使用上游儲存庫建立儲存庫 23
連接到儲存庫 24
使用套件管理員用戶端 24
刪除儲存庫 24
刪除儲存庫 （主控台） 25
刪除儲存庫 (AWS CLI) 25
防止儲存庫遭到刪除 25

列出儲存庫 27
列出 帳戶中的 AWS 儲存庫 27
列出網域中的儲存庫 28
檢視或修改儲存庫組態 30
檢視或修改儲存庫組態 （主控台） 30
檢視或修改儲存庫組態 (AWS CLI) 31
儲存庫政策 33
建立資源政策以授予讀取存取權 33
設定政策 35
讀取政策 36
刪除政策 36
將讀取存取權授予委託人 37
授予套件的寫入存取權 37
授予對儲存庫的寫入存取權 38
儲存庫與網域政策之間的互動 39
標記儲存庫 40
標籤儲存庫 (CLI) 40
標籤儲存庫 （主控台） 43
使用上游儲存庫 48
上游儲存庫和外部連線有何不同？ 48
新增或移除上游儲存庫 49
新增或移除上游儲存庫 （主控台） 49
新增或移除上游儲存庫 (AWS CLI) 50
將 CodeArtifact 儲存庫連線至公有儲存庫 52
連線至外部儲存庫 （主控台） 53
連線至外部儲存庫 (CLI) 54
支援的外部連線儲存庫 55
移除外部連線 (CLI) 56
使用上游儲存庫請求套件版本 56
來自上游儲存庫的套件保留 57
透過上游關係擷取套件 57
中繼儲存庫中的套件保留 59
從外部連線請求套件 60
從外部連線擷取套件 60
外部連線延遲 62
無法使用外部儲存庫時的 CodeArtifact 行為 62

新套件版本的可用性 63
匯入具有多個資產的套件版本 63
上游儲存庫優先順序 63
簡單優先順序範例 64
複雜優先順序範例 65
上游儲存庫的 API 行為 66
使用 套件 69
套件概觀 69
支援的套件格式 70
套件發佈 70
套件版本狀態 72
套件名稱、套件版本和資產名稱標準化 73
列出套件名稱 73
列出 npm 套件名稱 75
列出 Maven 套件名稱 76
列出 Python 套件名稱 77
依套件名稱字首篩選 77
支援的搜尋選項組合 78
格式輸出 78
預設值和其他選項 79
列出套件版本 79
列出 npm 套件版本 81
列出 Maven 套件版本 82
排序版本 82
預設顯示版本 83
格式輸出 83
列出套件版本資產 84
列出 npm 套件的資產 85
列出 Maven 套件的資產 85
下載套件版本資產 86
在儲存庫之間複製套件 87
複製套件所需的 IAM 許可 87
複製套件版本 88
從上游儲存庫複製套件 89
複製範圍 npm 套件 89
複製 Maven 套件版本 89

來源儲存庫中不存在的版本 90
已存在於目的地儲存庫的版本 91
指定套件版本修訂 92
複製 npm 套件 93
刪除套件或套件版本 93
刪除套件 (AWS CLI) 94
刪除套件 (主控台) 94
刪除套件版本 (AWS CLI) 95
刪除套件版本 (主控台) 96
刪除 npm 套件或套件版本 96
刪除 Maven 套件或套件版本 96
刪除套件或套件版本的最佳實務 97
檢視和更新套件版本詳細資訊和相依性 97
檢視套件版本詳細資訊 97
檢視 npm 套件版本詳細資訊 99
檢視 Maven 套件版本詳細資訊 100
檢視套件版本相依性 100
檢視套件版本 readme 檔案 101
更新套件版本狀態 102
更新套件版本狀態 102
更新套件版本狀態所需的 IAM 許可 104
更新範圍 npm 套件的狀態 104
更新 Maven 套件的狀態 104
指定套件版本修訂 105
使用預期的狀態參數 106
個別套件版本的錯誤 107
處置套件版本 108
編輯套件原始伺服器控制項 110
常見的套件存取控制案例 110
套件原始伺服器控制設定 112
預設套件原始伺服器控制設定 112
套件原始伺服器控制如何與套件群組原始伺服器控制互動 113
編輯套件原始伺服器控制項 114
發佈和上游儲存庫 115
使用套件群組 116
建立套件群組 116

建立套件群組 （主控台） 116
建立套件群組 (AWS CLI) 118
檢視或編輯套件群組 118
檢視或編輯套件群組 （主控台） 118
檢視或編輯套件群組 (AWS CLI) 119
刪除套件群組 120
刪除套件群組 （主控台） 120
刪除套件群組 (AWS CLI) 120
套件群組原始伺服器控制 121
限制設定 121
允許儲存庫清單 122
編輯套件群組原始伺服器控制設定 123
套件群組原始伺服器控制組態範例 124
套件群組原始伺服器控制設定如何與套件原始伺服器控制設定互動 126
套件群組定義語法和相符行為 126
套件群組定義語法和範例 126
套件群組階層和模式特異性 127
字詞、字詞邊界和字首比對 128
區分大小寫 129
強配對和弱配對 129
其他變化 129
標記套件群組 130
標籤套件群組 (CLI) 130
使用網域 134
網域概觀 134
跨帳戶網域 135
CodeArtifact 支援的 AWS KMS 金鑰類型 135
建立網域 136
建立網域 （主控台） 136
建立網域 (AWS CLI) 137
範例 AWS KMS 金鑰政策 138
刪除網域 139
刪除網域的限制 140
刪除網域 （主控台） 140
刪除網域 (AWS CLI) 140
網域政策 141

啟用網域的跨帳戶存取 141
網域政策範例 143
使用 的網域政策範例 AWS Organizations 144
設定網域政策 144
讀取網域政策 145
刪除網域政策 146
標記網域 146
標籤網域 (CLI) 146
標籤網域 （主控台） 149
使用貨運 153
設定和使用貨運 153
使用 CodeArtifact 設定貨運 153
安裝貨運箱 157
發佈貨運箱 158
貨運命令支援 158
需要存取登錄檔的支援命令 159
不支援的命令 159
使用 Maven 160
搭配 Gradle 使用 CodeArtifact 160
擷取相依性 161
擷取外掛程式 162
發佈成品 163
在 IntelliJ IDEA 中執行 Gradle 建置 164
將 CodeArtifact 與 mvn 搭配使用 168
擷取相依性 161
發佈成品 163
發佈第三方成品 173
限制將 Maven 相依性下載到 CodeArtifact 儲存庫 174
Apache Maven 專案資訊 176
搭配 deps.edn 使用 CodeArtifact 176
擷取相依性 176
發佈成品 177
使用 curl 發佈 178
使用 Maven 檢查總和 180
檢查總和儲存 180
發佈期間的檢查總和不相符 182

從檢查總和不相符復原 182
使用 Maven 快照 183
CodeArtifact 中的快照發佈 183
使用快照版本 185
刪除快照版本 186
使用 curl 發佈快照 186
快照和外部連線 188
快照和上游儲存庫 188
從上游和外部連線請求 Maven 套件 189
匯入標準資產名稱 189
匯入非標準資產名稱 189
檢查資產原始伺服器 190
在上游儲存庫中匯入新的資產和套件版本狀態 190
Maven 疑難排解 191
停用平行放置以修正錯誤 429：請求太多 191
使用 NPM 192
設定和使用 npm 192
使用登入命令設定 npm 192
不使用登入命令設定 npm 193
執行 npm 命令 195
驗證 npm 身分驗證和授權 196
變更回預設 npm 登錄檔 196
使用 npm 8.x 或更高版本對慢速安裝進行故障診斷 196
設定和使用 Yarn 197
使用 `aws codeartifact login`命令設定 Yarn 1.X 197
使用 `yarn config set`命令設定 Yarn 2.X 198
npm 命令支援 200
與儲存庫互動的支援命令 201
支援的用戶端命令 202
不支援的命令 159
npm 標籤處理 205
使用 npm 用戶端編輯標籤 205
npm 標籤和 CopyPackageVersions API 206
npm 標籤和上游儲存庫 206
支援 npm 相容套件管理員 207
使用 NuGet 209

搭配 Visual Studio 使用 CodeArtifact 209
使用 CodeArtifact 登入資料提供者設定 Visual Studio 210
使用 Visual Studio Package Manager 主控台 211
將 CodeArtifact 與 nuget 或 dotnet 搭配使用 211
設定 nuget 或 dotnet CLI 212
使用 NuGet 套件 216
發佈 NuGet 套件 218
CodeArtifact NuGet 登入資料提供者參考 218
CodeArtifact NuGet 登入資料提供者版本 219
NuGet 套件名稱、版本和資產名稱標準化 220
NuGet 相容性 221
一般 NuGet 相容性 221
NuGet 命令列支援 221
使用 Python 222
搭配 CodeArtifact 設定和使用 pip 222
使用 `login`命令設定 pip 222
設定不含登入命令的 pip 223
執行 pip 224
使用 CodeArtifact 設定和使用雙身 224
使用 `login`命令設定雙身 224
不使用 `login`命令設定雙身 225
執行雙身 226
Python 套件名稱標準化 226
Python 相容性 227
pip 命令支援 227
從上游和外部連線請求 Python 套件 228
Yanked 套件版本 229
為什麼 CodeArtifact 未擷取套件版本的最新偏擺中繼資料或資產？ 229
使用 Ruby 231
設定和使用 RubyGems 和 Bundler 231
使用 CodeArtifact 設定 RubyGems (`gem`) 和 Bundler (`bundle`) 231
安裝 Ruby Gem 套件 236
發佈 Ruby Gem 套件 237
RubyGems 命令支援 238
Bundler 相容性 239
Bundler 相容性 239

使用 Swift 240
使用 CodeArtifact 設定 Swift 240
使用登入命令設定 Swift 240
不使用登入命令設定 Swift 241
使用和發佈 Swift 套件 245
使用 Swift 套件 245
在 Xcode 中使用 Swift 套件 247
發佈 Swift 套件 247
從 GitHub 擷取 Swift 套件並重新發佈至 CodeArtifact 250
Swift 套件名稱和命名空間標準化 251
Swift 疑難排解 252
即使設定 Swift Package Manager 之後，我還是在 Xcode 中收到 401 錯誤 252
由於密碼的金鑰鏈提示，Xcode 掛在 CI 機器上 252
使用一般套件 255
一般套件概觀 255
一般套件限制 255
支援的命令 256
發佈和使用一般套件 256
發佈一般套件 257
列出一般套件資產 259
下載一般套件資產 260
搭配 CodeBuild 使用 CodeArtifact CodeBuild 262
在 CodeBuild 中使用 npm 套件 262
使用 IAM 角色設定許可 262
登入並使用 npm 263
在 CodeBuild 中使用 Python 套件 264
使用 IAM 角色設定許可 264
登入並使用 pip 或 Twine 265
在 CodeBuild 中使用 Maven 套件 267
使用 IAM 角色設定許可 267
使用 gradle 或 mvn 268
在 CodeBuild 中使用 NuGet 套件 269
使用 IAM 角色設定許可 269
使用 NuGet 套件 270
使用 NuGet 套件建置 271
發佈 NuGet 套件 274

相依性快取 ... 275
監控 CodeArtifact ... 277
監控 CodeArtifact 事件 ... 277
CodeArtifact 事件格式和範例 ... 278
使用事件啟動 CodePipeline 執行 ... 281
設定 EventBridge 許可 ... 282
建立 EventBridge 規則 ... 282
建立 EventBridge 規則目標 ... 282
使用 事件來執行 Lambda 函數 ... 282
建立 EventBridge 規則 ... 283
建立 EventBridge 規則目標 ... 283
設定 EventBridge 許可 ... 283
安全 ... 284
資料保護 ... 284
資料加密 ... 285
流量隱私權 ... 285
監控 ... 286
使用 記錄 CodeArtifact API 呼叫 AWS CloudTrail ... 286
法規遵循驗證 ... 290
身分驗證和字符 ... 291
使用 login命令建立的字符 ... 292
呼叫 GetAuthorizationToken API 所需的許可 ... 293
使用 GetAuthorizationToken API 建立的字符 ... 293
使用 環境變數傳遞身分驗證字符 ... 294
撤銷 CodeArtifact 授權字符 ... 295
恢復能力 ... 295
基礎架構安全 ... 296
相依性替代攻擊 ... 296
身分和存取權管理 ... 297
目標對象 ... 297
使用身分驗證 ... 298
使用政策管理存取權 ... 300
How AWS CodeArtifact 可與 IAM 搭配使用 ... 302
身分型政策範例 ... 308
使用標籤控制 CodeArtifact 資源的存取 ... 316
AWS CodeArtifact 許可參考 ... 320

故障診斷 323
使用 VPC 端點 325
建立 VPC 端點 325
建立 Amazon S3 閘道端點 326
AWS CodeArtifact 的最低 Amazon S3 儲存貯體許可 327
從 VPC 使用 CodeArtifact 329
在沒有私有 DNS 的情況下使用codeartifact.repositories端點 329
建立 VPC 端點政策 331
AWS CloudFormation 資源 332
CodeArtifact 和 AWS CloudFormation 範本 332
防止刪除 CodeArtifact 資源 332
進一步了解 AWS CloudFormation 333
故障診斷 334
我無法檢視通知 334
標記資源 335
CodeArtifact 成本分配與標籤 335
在 CodeArtifact 中分配資料儲存成本 336
在 CodeArtifact 中分配請求成本 336
AWS CodeArtifact 中的配額 337
文件歷史紀錄 340
..... cccxlviii

什麼是 AWS CodeArtifact？

AWS CodeArtifact 是一種安全、可擴展性高的受管成品儲存庫服務，可協助組織存放和共用軟體套件以進行應用程式開發。您可以將 CodeArtifact 與 NuGet CLI、Maven、Gradle、npm、Lunes、pip 和 Twine 等熱門建置工具和套件管理員搭配使用。CodeArtifact 可協助您減少管理自有成品儲存系統的需求，或擔心擴展其基礎設施。您可以存放在 CodeArtifact 儲存庫中的套件數量或總大小沒有限制。

您可以在私有 CodeArtifact 儲存庫與外部公有儲存庫之間建立連線，例如 npmjs.com 或 Maven Central。然後CodeArtifact 會在套件管理員請求時，從公有儲存庫隨需擷取和存放套件。這可讓您更方便地使用應用程式使用的開放原始碼相依性，並協助確保它們隨時可用於建置和開發。您也可以將私有套件發佈至 CodeArtifact 儲存庫。這可協助您在組織中的多個應用程式和開發團隊之間共用專屬軟體元件。

如需詳細資訊，請參閱 AWS CodeArtifact。

CodeArtifact 如何運作？

CodeArtifact 會將軟體套件存放在 儲存庫中。儲存庫是多邊形 - 單一儲存庫可以包含任何支援類型的套件。每個 CodeArtifact 儲存庫都是單一 CodeArtifact 網域的成員。我們建議您為組織使用一個生產網域與一或多個儲存庫。例如，您可以將每個儲存庫用於不同的開發團隊。然後，您可以探索儲存庫中的套件，並與開發團隊共用。

若要將套件新增至儲存庫，請設定套件管理員，例如 npm 或 Maven，以使用儲存庫端點 (URL)。然後，您可以使用套件管理員將套件發佈至儲存庫。您也可以使用外部連線將開放原始碼套件匯入儲存庫，例如 npmjs、NuGet Gallery、Maven Central 或 PyPI。如需詳細資訊，請參閱將 CodeArtifact 儲存庫連線至公有儲存庫。

您可以將一個儲存庫中的套件提供給相同網域中的另一個儲存庫。若要這樣做，請將一個儲存庫設定為另一個儲存庫的上游。上游儲存庫可用的所有套件版本也可供下游儲存庫使用。此外，透過與公有儲存庫的外部連線供上游儲存庫使用的所有套件，都可供下游儲存庫使用。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。

CodeArtifact 要求使用者向 服務進行身分驗證，以發佈或取用套件版本。您必須使用 AWS 憑證建立授權字符，以驗證 CodeArtifact 服務。CodeArtifact 儲存庫中的套件無法公開提供。如需 CodeArtifact 中身分驗證和存取的詳細資訊，請參閱 AWS CodeArtifact 身分驗證和字符。

AWS CodeArtifact 概念

以下是當您使用 CodeArtifact 時需要了解的一些概念和術語。

主題

- 資產
- 網域
- 儲存庫
- 套件
- 套件群組
- 套件命名空間
- 套件版本
- 套件版本修訂
- 上游儲存庫

資產

資產是存放在 CodeArtifact 中的個別檔案，與套件版本相關聯，例如 npm `.tgz` 檔案或 Maven POM 和 JAR 檔案。

網域

儲存庫會彙總到稱為網域的更高層級實體。所有套件資產和中繼資料都會存放在網域中，但會透過儲存庫使用。指定的套件資產，例如 Maven JAR 檔案，每個網域會儲存一次，無論它存在於多少儲存庫。網域中的所有資產和中繼資料都會使用存放在 AWS KMS key () 中的相同 AWS Key Management Service (KMS 金鑰） 進行加密AWS KMS。

每個儲存庫都是單一網域的成員，無法移至不同的網域。

使用網域，您可以在多個儲存庫中套用組織政策。透過此方法，您可以決定哪些帳戶可以存取網域中的儲存庫，以及哪些公有儲存庫可以用作套件的來源。

雖然組織可以有多個網域，但我們建議使用包含所有已發佈成品的單一生產網域。如此一來，團隊就可以在您的組織中尋找和共用套件。

儲存庫

CodeArtifact 儲存庫包含一組套件版本，每個版本都會對應到一組資產。儲存庫是多邊形 - 單一儲存庫可以包含任何支援類型的套件。每個儲存庫都會公開端點，以使用 nuget CLI、npm CLI、Maven CLI (mvn) 和 pip 等工具來擷取和發佈套件。每個網域最多可以建立 1，000 個儲存庫。

套件

套件是軟體和中繼資料的套件，這些中繼資料是解決相依性和安裝軟體所需的。在 CodeArtifact 中，套件包含套件名稱、選用的命名空間，例如 `@types`中的 `@types/node`、一組套件版本，以及套件層級中繼資料，例如 npm 標籤。

AWS CodeArtifact 支援貨運、一般、Maven、npm、NuGet、PyPI、Ruby、Swift 套件格式。

套件群組

套件群組可用來將組態套用至多個套件，這些套件符合使用套件格式、套件命名空間和套件名稱定義的模式。您可以使用套件群組，更方便地設定多個套件的套件原始伺服器控制。套件原始伺服器控制用於封鎖或允許擷取或發佈新的套件版本，以保護使用者免受稱為相依性替代攻擊的惡意動作。

套件命名空間

有些套件格式支援階層式套件名稱，可將套件組織成邏輯群組，並協助避免名稱衝突。例如，npm 支援範圍。如需詳細資訊，請參閱 npm 範圍文件。npm 套件的範圍`@types/node`為 `@types`，名稱為 `node`。`@types` 範圍內還有許多其他套件名稱。在 CodeArtifact 中，範圍（「類型」）稱為套件命名空間，而名稱（「節點」）稱為套件名稱。對於 Maven 套件，套件命名空間對應至 Maven groupID。Maven 套件`org.apache.logging.log4j:log4j`的 groupID（套件命名空間）為 `org.apache.logging.log4j`，而 artifactID（套件名稱）為 `log4j`。對於一般套件，需要命名空間。有些 PyPI 等套件格式不支援階層式名稱，其概念類似於 npm 範圍或 Maven groupID。如果沒有將套件名稱分組的方法，可能更難避免名稱衝突。

套件版本

套件版本可識別套件的特定版本，例如 `@types/node 12.6.9`。版本編號格式和語意因不同的套件格式而異。例如，npm 套件版本必須符合語意版本控制規格。在 CodeArtifact 中，套件版本包含版本識別符、套件版本層級中繼資料和一組資產。

套件版本修訂

套件版本修訂是可識別套件版本特定資產和中繼資料集的字串。每次更新套件版本時，都會建立新的套件版本修訂。例如，您可以發佈 Python 套件版本的來源分佈封存 (sdist)，稍後再將包含編譯程式碼的 Python 輪新增至相同版本。當您發佈 wheel 時，會建立新的套件版本修訂。

上游儲存庫

一個儲存庫位於另一個儲存庫的上游，此時可從下游儲存庫的儲存庫端點存取其中的套件版本。此方法會從用戶端的角度有效地合併兩個儲存庫的內容。使用 CodeArtifact，您可以在兩個儲存庫之間建立上游關係。

如何開始使用 CodeArtifact？

建議您完成下列步驟：

1. 閱讀 以進一步了解 CodeArtifactAWS CodeArtifact 概念。
2. 依照中的步驟設定您的 AWS 帳戶 AWS CLI、 和 IAM 使用者使用 AWS CodeArtifact 設定。
3. 遵循 中的指示使用 CodeArtifactCodeArtifact 入門。

使用 AWS CodeArtifact 設定

如果您已經註冊 Amazon Web Services (AWS)，您可以立即開始使用 AWS CodeArtifact。您可以開啟 CodeArtifact 主控台，選擇建立網域和儲存庫，然後遵循啟動精靈中的步驟來建立您的第一個網域和儲存庫。

如果您尚未註冊 AWS ，或需要協助建立第一個網域和儲存庫，請完成下列任務以設定使用 CodeArtifact：

主題

- 註冊 AWS
- 安裝或升級 ，然後設定 AWS CLI
- 佈建 IAM 使用者
- 安裝您的套件管理員或建置工具

註冊 AWS

當您註冊 Amazon Web Services (AWS) 時，您只需支付使用的服務和資源的費用，包括 AWS CodeArtifact。

如果您已有 AWS 帳戶，請跳到下一個任務 安裝或升級 ，然後設定 AWS CLI。如果您沒有 AWS 帳戶，請使用下列程序建立一個。

建立 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   註冊程序的一部分包括接聽電話或文字訊息，並在電話鍵盤上輸入驗證碼。

   當您註冊 時 AWS 帳戶，AWS 帳戶根使用者會建立 。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

安裝或升級 ，然後設定 AWS CLI

若要從本機開發機器上的 AWS Command Line Interface (AWS CLI) 呼叫 CodeArtifact 命令，您必須安裝 AWS CLI。

如果您 AWS CLI 已安裝舊版的 ，則必須進行升級，才能使用 CodeArtifact 命令。CodeArtifact 命令可在下列 AWS CLI 版本中使用：

1. AWS CLI 1：1.18.77 及更新版本
2. AWS CLI 2：2.0.21 及更新版本

若要檢查版本，請使用 `aws --version`命令。

安裝和設定 AWS CLI

1. AWS CLI 使用安裝 中的指示來安裝或升級 AWS Command Line Interface。
2. 使用 configure 命令設定 AWS CLI，如下所示。

   ```
   aws configure
   ```

   出現提示時，請指定您要與 CodeArtifact 搭配使用之 IAM 使用者的 AWS 存取金鑰和 AWS 私密存取金鑰。提示輸入預設 AWS 區域 名稱時，請指定您要建立管道的區域，例如 `us-east-2`。系統提示您輸入預設輸出格式時，請指定 `json`。

   > **Important**
   >
   > 當您設定 時 AWS CLI，系統會提示您指定 AWS 區域。在 中選擇區域和端點中列出的其中一個支援區域AWS 一般參考。

   如需詳細資訊，請參閱設定 AWS Command Line Interface 和管理 IAM 使用者的存取金鑰。
3. 若要驗證安裝或升級，請從 呼叫下列命令 AWS CLI。

   ```
   aws codeartifact help
   ```

   如果成功，此命令會顯示可用的 CodeArtifact 命令清單。

接著，您可以建立 IAM 使用者，並授予該使用者 CodeArtifact 的存取權。如需詳細資訊，請參閱佈建 IAM 使用者。

佈建 IAM 使用者

請遵循這些指示，準備 IAM 使用者使用 CodeArtifact。

佈建 anIAM 使用者

1. 建立 IAM 使用者，或使用與您的 相關聯的使用者 AWS 帳戶。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 使用者和 AWS IAM 政策概觀。
2. 授予 IAM 使用者 CodeArtifact 的存取權。

   - 選項 1：建立自訂 IAM 政策。透過自訂 IAM 政策，您可以提供所需的最低許可，並變更身分驗證字符的持續時間。如需詳細資訊和範例政策，請參閱AWS CodeArtifact 的身分型政策範例。
   - 選項 2：使用 `AWSCodeArtifactAdminAccess` AWS 受管政策。下列程式碼片段顯示此政策的內容。

     > **Important**
     >
     > 此政策會授予所有 CodeArtifact APIs存取權。建議您一律使用完成任務所需的最低許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 最佳實務。

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

```
                "sts:AWSServiceName": "codeartifact.amazonaws.com"
            }
        }
    }
  ]
}
```

> **Note**
>
> `sts:GetServiceBearerToken` 許可必須新增至 IAM 使用者或角色政策。雖然它可以新增到 CodeArtifact 網域或儲存庫資源政策，但許可不會對資源政策產生影響。

呼叫 CodeArtifact `GetAuthorizationToken` API 需要 `sts:GetServiceBearerToken`許可。此 API 會傳回權杖，在`pip`搭配 CodeArtifact 使用 npm或 等套件管理員時，必須使用該權杖。若要搭配 CodeArtifact 儲存庫使用套件管理員，您的 IAM 使用者或角色必須允許 `sts:GetServiceBearerToken`，如上述政策範例所示。

如果您尚未安裝計劃與 CodeArtifact 搭配使用的套件管理員或建置工具，請參閱 安裝您的套件管理員或建置工具。

安裝您的套件管理員或建置工具

若要從 CodeArtifact 發佈或使用套件，您必須使用套件管理員。每個套件類型都有不同的套件管理員。以下清單包含一些可與 CodeArtifact 搭配使用的套件管理員。如果您尚未安裝要使用的套件類型的套件管理員。

- 對於 npm，請使用 npm CLI 或 pnpm。
- 對於 Maven，請使用 Apache Maven (mvn) 或 Gradle。
- 對於 Python，請使用 pip 來安裝套件，並使用雙工來發佈套件。
- 對於 NuGet，請使用 Visual Studio 中的 Toolkit for Visual Studio 或 nuget 或 dotnet CLIs。
- 對於一般套件，請使用 AWS CLI或 SDK 來發佈和下載套件內容。

後續步驟

您的後續步驟將取決於您搭配 CodeArtifact 使用的套件類型或類型，以及 CodeArtifact 資源的狀態。

如果您是第一次為自己、團隊或組織開始使用 CodeArtifact，請參閱下列文件以取得一般入門資訊，並協助您建立所需的資源。

- 開始使用主控台
- 開始使用 AWS CLI

如果您的資源已建立，且您已準備好設定套件管理員將套件推送至 CodeArtifact 儲存庫或從 CodeArtifact 儲存庫安裝套件，請參閱對應至套件類型或套件管理員的文件。

- 搭配 npm 使用 CodeArtifact
- 搭配 Python 使用 CodeArtifact
- 搭配 Maven 使用 CodeArtifact
- 搭配 NuGet 使用 CodeArtifact
- 搭配一般套件使用 CodeArtifact

CodeArtifact 入門

在本入門教學課程中，您會使用 CodeArtifact 建立下列項目：

- 稱為 的網域`my-domain`。
- 包含在 中的名為 `my-repo`的儲存庫`my-domain`。
- 包含在 中的名為 `npm-store`的儲存庫`my-domain`。與 npm 公有儲存庫`npm-store`有外部連線。此連線用於將 npm 套件擷取至`my-repo`儲存庫。

在開始本教學課程之前，建議您檢閱 CodeArtifact AWS CodeArtifact 概念。

> **Note**
>
> 本教學要求您建立資源，可能會對您的 AWS 帳戶收費。如需詳細資訊，請參閱 CodeArtifact 定價。

主題

- 先決條件
- 開始使用主控台
- 開始使用 AWS CLI

先決條件

您可以使用 AWS Management Console 或 AWS Command Line Interface () 完成本教學課程AWS CLI。若要遵循教學課程，您必須先完成下列先決條件：

- 完成「使用 AWS CodeArtifact 設定」中的步驟。
- 安裝 npm CLI。如需詳細資訊，請參閱 npm 文件中的下載並安裝 Node.js 和 npm。

開始使用主控台

執行下列步驟，以使用 開始使用 CodeArtifact AWS Management Console。本指南使用 npm套件管理工具，如果您使用的是不同的套件管理工具，則需要修改下列部分步驟。

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/codesuite/codeartifact/start 開啟 AWS CodeArtifact 主控台。如需詳細資訊，請參閱使用 AWS CodeArtifact 設定。
2. 選擇建立儲存庫。
3. 在儲存庫名稱中，輸入 **my-repo**。
4. （選用）在儲存庫描述中，輸入儲存庫的選用描述。
5. 在公有上游儲存庫中，選取 npm-store 以建立連線至儲存庫上游 npmjs 的`my-repo`儲存庫。

   CodeArtifact 會為您指派`npm-store`此儲存庫的名稱。上游儲存庫中可用的所有套件`npm-store`也可供其下游儲存庫 使用`my-repo`。
6. 選擇 Next (下一步)。
7. 在 AWS 帳戶中，選擇此 AWS 帳戶。
8. 在網域名稱中，輸入 **my-domain**。
9. 展開 Additional configuration (其他組態)。
10. 您必須使用 AWS KMS key (KMS 金鑰） 來加密網域中的所有資產。您可以使用您管理的 AWS 受管金鑰 或 KMS 金鑰：

    - 如果您想要使用預設 ，請選擇 AWS 受管金鑰 AWS 受管金鑰。
    - 如果您想要使用您管理的 KMS 金鑰，請選擇客戶受管金鑰。若要使用您管理的 KMS 金鑰，請在客戶受管金鑰 ARN 中搜尋並選擇 KMS 金鑰。

    如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的 AWS 受管金鑰和客戶受管金鑰。
11. 選擇 Next (下一步)。
12. 在檢閱和建立中，檢閱為您建立的 CodeArtifact。

    - 套件流程顯示 `my-domain`、 `my-repo`和 `npm-store` 的關聯性。
    - 步驟 1：建立儲存庫會顯示 `my-repo`和 的詳細資訊`npm-store`。
    - 步驟 2：選取網域會顯示有關 的詳細資訊`my-domain`。

    當您準備好時，請選擇建立儲存庫。
13. 在 my-repo 頁面上，選擇檢視連線指示，然後選擇 npm。
14. 使用此CodeArtifact `login`命令 AWS CLI ，執行設定 npm 用戶端下顯示的命令。 AWS CLI CodeArtifact

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

您應該會收到確認登入成功的輸出。

```
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
```

如果您收到錯誤 `Could not connect to the endpoint URL`，請確定您的 AWS CLI 已設定，且您的預設區域名稱已設定為您建立儲存庫的相同區域，請參閱設定 AWS 命令列界面。

如需詳細資訊，請參閱 搭配 CodeArtifact 設定和使用 npm

15. 使用 npm CLI 來安裝 npm 套件。例如，若要安裝熱門的 npm 套件 `lodash`，請使用下列命令。

```
npm install lodash
```

16. 返回 CodeArtifact 主控台。如果您的 my-repo 儲存庫已開啟，請重新整理頁面。否則，在導覽窗格中，選擇儲存庫，然後選擇 my-repo。

    在套件下，您應該會看到已安裝的 npm 程式庫或套件。您可以選擇套件的名稱，以檢視其版本和狀態。您可以選擇其最新版本來檢視套件詳細資訊，例如相依性、資產等。

> **Note**
>
> 安裝套件時，以及擷取至儲存庫時，可能會有延遲。

17. 若要避免進一步 AWS 收費，請刪除您在本教學課程中使用的資源：

> **Note**
>
> 您無法刪除包含儲存庫的網域，因此您必須先刪除 `my-repo`和 ，`npm-store`再刪除 `my-domain`。

    a. 從導覽窗格中，選擇儲存庫。

    b. 選擇 npm-store，選擇刪除，然後依照步驟刪除儲存庫。

c. 選擇 my-repo，選擇刪除，然後依照步驟刪除儲存庫。

d. 從導覽窗格中，選擇網域。

e. 選擇 my-domain，選擇刪除，然後依照步驟刪除網域。

開始使用 AWS CLI

執行下列步驟，以使用 AWS Command Line Interface () 開始使用 CodeArtifact AWS CLI。如需詳細資訊，請參閱安裝或升級 ，然後設定 AWS CLI。本指南使用 npm套件管理工具，如果您使用不同的套件管理工具，則需要修改下列部分步驟。

1. 使用 AWS CLI 執行 create-domain命令。

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

JSON 格式的資料會顯示在輸出中，其中包含新網域的詳細資訊。

```
{
    "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
    }
}
```

如果您收到錯誤 `Could not connect to the endpoint URL`，請確定您的 AWS CLI 已設定，且您的預設區域名稱已設定為您建立儲存庫的相同區域，請參閱設定 AWS 命令列界面。

2. 使用 create-repository命令在您的網域中建立儲存庫。

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

JSON 格式的資料會顯示在輸出中，其中包含新儲存庫的詳細資訊。

```
{
    "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. 使用 create-repository命令為您的儲存庫建立上游my-repo儲存庫。

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

JSON 格式的資料會顯示在輸出中，其中包含新儲存庫的詳細資訊。

```
{
    "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. 使用 associate-external-connection命令將外部連線新增至 npm 公有儲存庫。 npm-store

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

JSON 格式的資料會顯示在輸出中，其中包含儲存庫及其新外部連線的詳細資訊。

```
{
```

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

如需詳細資訊，請參閱將 CodeArtifact 儲存庫連線至公有儲存庫。

5. 使用 update-repository命令將儲存npm-store庫做為上游儲存庫與my-repo儲存庫建立關聯。

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

JSON 格式的資料會顯示在輸出中，其中包含更新儲存庫的詳細資訊，包括其新的上游儲存庫。

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

```
}
```

如需詳細資訊，請參閱新增或移除上游儲存庫 (AWS CLI)。

6. 使用 login命令來設定 npm 套件管理員與my-repo儲存庫。

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

您應該會收到確認登入成功的輸出。

```
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
```

如需詳細資訊，請參閱搭配 CodeArtifact 設定和使用 npm。

7. 使用 npm CLI 來安裝 npm 套件。例如，若要安裝熱門的 npm 套件 lodash，請使用下列命令。

```
npm install lodash
```

8. 使用 list-packages命令來檢視您剛安裝在my-repo儲存庫中的套件。

> **Note**
>
> 安裝npm install命令完成與儲存庫中顯示套件之間可能會有延遲。如需從公有儲存庫擷取套件時一般延遲的詳細資訊，請參閱 外部連線延遲。

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

JSON 格式的資料會顯示在輸出中，其中包含您安裝的套件格式和名稱。

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

```
}
```

您現在有三個 CodeArtifact 資源：

- 網域 `my-domain`。
- 中包含`my-repo`的儲存庫`my-domain`。此儲存庫有可用的 npm 套件。
- 中包含`npm-store`的儲存庫`my-domain`。此儲存庫與公有 npm 儲存庫具有外部連線，並作為上游儲存庫與`my-repo`儲存庫建立關聯。

9. 若要避免進一步 AWS 收費，請刪除您在本教學課程中使用的資源：

> **Note**
>
> 您無法刪除包含儲存庫的網域，因此您必須先刪除 `my-repo`和 ，`npm-store`再刪除 `my-domain`。

a. 使用 delete-repository命令來刪除儲存`npm-store`庫。

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

JSON 格式的資料會顯示在輸出中，其中包含已刪除儲存庫的詳細資訊。

```
{
    "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. 使用 delete-repository命令來刪除儲存npm-store庫。

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

JSON 格式的資料會顯示在輸出中，其中包含已刪除儲存庫的詳細資訊。

```
{
    "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. 使用 delete-domain命令來刪除儲存my-domain庫。

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

JSON 格式的資料會顯示在輸出中，其中包含已刪除網域的詳細資訊。

```
{
    "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
    }
}
```

在 CodeArtifact 中使用儲存庫

這些主題說明如何使用 CodeArtifact 主控台 AWS CLI和 CodeArtifact APIs 來建立、列出、更新和刪除儲存庫。

主題

- 建立 儲存庫
- 連接到儲存庫
- 刪除儲存庫
- 列出儲存庫
- 檢視或修改儲存庫組態
- 儲存庫政策
- 在 CodeArtifact 中標記儲存庫

建立 儲存庫

由於 CodeArtifact 中的所有套件都存放在儲存庫中，若要使用 CodeArtifact，您必須建立一個。您可以使用 CodeArtifact 主控台、 AWS Command Line Interface (AWS CLI) 或 來建立儲存庫 AWS CloudFormation。每個儲存庫都會與您建立儲存庫時使用 AWS 的帳戶相關聯。您可以有多個儲存庫，它們會在網域中建立和分組。當您建立儲存庫時，它不包含任何套件。儲存庫是多邊形，這表示單一儲存庫可以包含任何支援類型的套件。

如需 CodeArtifact 服務限制的相關資訊，例如單一網域中允許的儲存庫數量上限，請參閱 AWS CodeArtifact 中的配額。如果您達到允許的儲存庫數量上限，您可以刪除儲存庫以騰出更多空間。

儲存庫可以有一或多個與其相關聯的 CodeArtifact 儲存庫做為上游儲存庫。這可讓套件管理員用戶端使用單一 URL 端點存取多個儲存庫中包含的套件。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。

如需使用 CloudFormation 管理 CodeArtifact 儲存庫的詳細資訊，請參閱 使用 建立 CodeArtifact 資源 AWS CloudFormation。

> **Note**
>
> 建立儲存庫之後，您無法變更其名稱、相關聯的 AWS 帳戶或網域。

主題

- 建立儲存庫（主控台）
- 建立儲存庫 (AWS CLI)
- 使用上游儲存庫建立儲存庫

建立儲存庫（主控台）

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇建立儲存庫。
3. 針對儲存庫名稱，輸入儲存庫的名稱。
4. （選用）在儲存庫描述中，輸入儲存庫的選用描述。
5. （選用）在發佈上游儲存庫中，新增中繼儲存庫，將儲存庫連線至套件授權單位，例如 Maven Central 或 npmjs.com。
6. 選擇下一步。
7. 在 AWS 帳戶中，如果您已登入擁有網域的帳戶，請選擇此 AWS 帳戶。如果另一個 AWS 帳戶擁有網域，請選擇不同的 AWS 帳戶。
8. 在網域中，選擇將在其中建立儲存庫的網域。

   如果帳戶中沒有網域，您必須建立一個網域。在網域名稱中輸入新網域的名稱。

   展開 Additional configuration (其他組態)。

   您必須使用 AWS KMS key (KMS 金鑰) 來加密網域中的所有資產。您可以使用您管理的 AWS 受管金鑰 或 KMS 金鑰：

   > **Important**
   >
   > CodeArtifact 僅支援對稱 KMS 金鑰。您無法使用非對稱 KMS 金鑰來加密 CodeArtifact 網域。如需判斷 KMS 金鑰為對稱或非對稱的說明，請參閱識別對稱和非對稱 KMS 金鑰。

   - 如果您想要使用預設 ，請選擇 AWS 受管金鑰 AWS 受管金鑰。
   - 如果您想要使用您管理的 KMS 金鑰，請選擇客戶受管金鑰。若要使用您管理的 KMS 金鑰，請在客戶受管金鑰 ARN 中搜尋並選擇 KMS 金鑰。

如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的 AWS 受管金鑰和客戶受管金鑰。

9. 選擇下一步。
10. 在檢閱和建立中，檢閱為您建立的 CodeArtifact。

    - 套件流程顯示網域和儲存庫的連線方式。
    - 步驟 1：建立儲存庫會顯示將建立之儲存庫和選用上游儲存庫的詳細資訊。
    - 步驟 2：選取網域會顯示有關 的詳細資訊my_domain。

    當您準備好時，請選擇建立儲存庫。

建立儲存庫 (AWS CLI)

使用 `create-repository`命令在您的網域中建立儲存庫。

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

輸出範例：

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

新的儲存庫不包含任何套件。每個儲存庫都會與您建立儲存庫時所驗證 AWS 的帳戶相關聯。

建立具有標籤的儲存庫

若要建立具有標籤的儲存庫，請將 --tags 參數新增至您的create-domain命令。

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

使用上游儲存庫建立儲存庫

您可以在建立儲存庫時指定一或多個上游儲存庫。

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

輸出範例：

```
{
    "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**
>
> 若要使用上游建立儲存庫，您必須具有上游儲存庫上AssociateWithDownstreamRepository動作的許可。

若要在建立儲存庫之後將上游新增至儲存庫，請參閱 新增或移除上游儲存庫 （主控台）和 新增或移除上游儲存庫 (AWS CLI)。

連接到儲存庫

在您設定設定檔和登入資料以向您的帳戶 AWS 進行身分驗證之後，請決定要在 CodeArtifact 中使用的儲存庫。您有下列選項：

- 建立 儲存庫。如需詳細資訊，請參閱建立儲存庫。
- 使用您帳戶中已存在的儲存庫。您可以使用 `list-repositories`命令來尋找在 AWS 帳戶中建立的儲存庫。如需詳細資訊，請參閱列出儲存庫。
- 使用不同帳戶中的儲存庫 AWS 。如需詳細資訊，請參閱儲存庫政策。

使用套件管理員用戶端

知道要使用哪個儲存庫之後，請參閱下列其中一個主題。

- 搭配 Maven 使用 CodeArtifact
- 搭配 npm 使用 CodeArtifact
- 搭配 NuGet 使用 CodeArtifact
- 搭配 Python 使用 CodeArtifact

刪除儲存庫

您可以使用 CodeArtifact 主控台或 刪除儲存庫 AWS CLI。刪除儲存庫之後，您就無法再推送套件或從中提取套件。儲存庫中的所有套件都會永久無法使用且無法還原。您可以建立同名的儲存庫，但其內容會是空的。

> **Important**
>
> 刪除儲存庫無法復原。刪除儲存庫之後，您就無法再將其復原，也無法還原。

主題

- 刪除儲存庫 （主控台）

- 刪除儲存庫 (AWS CLI)
- 防止儲存庫遭到刪除

刪除儲存庫 （主控台）

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇您要刪除的儲存庫。
3. 選擇刪除，然後依照步驟刪除網域。

刪除儲存庫 (AWS CLI)

使用 `delete-repository`命令來刪除儲存庫。

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

輸出範例：

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

防止儲存庫遭到刪除

您可以包含類似以下的網域政策，防止儲存庫意外刪除：

```
{
```

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

此政策可防止所有主體刪除儲存庫，但如果您稍後決定需要刪除儲存庫，您可以依照下列步驟執行此操作：

1. 在網域政策中，將政策更新為下列項目：

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

   將 *repository-arn* 取代為您要刪除之儲存庫的 ARN。

2. 在 AWS CodeArtifact 主控台中，選擇儲存庫並刪除您選擇的儲存庫。

3. 刪除儲存庫之後，您可以變更政策，以防止意外偏離。

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

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

或者，您可以在儲存庫政策中包含相同的拒絕陳述式。這可讓您更靈活地保護高價值儲存庫免於刪除。

列出儲存庫

使用本主題中的命令來列出 AWS 帳戶或網域中的儲存庫。

列出 帳戶中的 AWS 儲存庫

使用此命令列出您 AWS 帳戶中的所有儲存庫。

```
aws codeartifact list-repositories
```

輸出範例：

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

您可以使用 list-repositories--max-results和 --next-token 參數，從 分頁回應。針對 --max-results，指定從 1 到 1000 的整數，以指定單一頁面中傳回的結果數目。其預設值為 50。若要傳回後續頁面，請list-repositories再次執行 ，並將先前命令輸出中接收nextToken的值傳遞給--next-token。不使用 --next-token選項時，一律會傳回結果的第一頁。

列出網域中的儲存庫

使用 list-repositories-in-domain取得網域中所有儲存庫的清單。

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

輸出顯示某些儲存庫是由不同的 AWS 帳戶管理。

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

您可以使用 list-repositories-in-domain--max-results和 --next-token 參數，從 分頁回應。針對 --max-results，指定從 1 到 1000 的整數，以指定單一頁面中傳回的結果數目。其預設值為 50。若要傳回後續頁面，請list-repositories-in-domain再次執行 ，並將上一個命令輸出中收到nextToken的值傳遞給 --next-token。不使用 --next-token選項時，一律會傳回結果的第一頁。

若要輸出更精簡清單中的儲存庫名稱，請嘗試下列命令。

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

輸出範例：

```
repo1
repo2
repo3
```

除了儲存庫名稱之外，下列範例還會輸出帳戶 ID。

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

輸出範例：

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

如需 `--query` 參數的詳細資訊，請參閱 CodeArtifact API 參考中的 ListRepositories。

檢視或修改儲存庫組態

您可以使用 CodeArtifact 主控台或 AWS Command Line Interface () 檢視和更新儲存庫的詳細資訊 AWS CLI。

> **Note**
>
> 建立儲存庫之後，您無法變更其名稱、相關聯的 AWS 帳戶或網域。

主題

- 檢視或修改儲存庫組態 （ 主控台 ）
- 檢視或修改儲存庫組態 (AWS CLI)

檢視或修改儲存庫組態 （ 主控台 ）

您可以使用 CodeArtifact 主控台檢視和更新儲存庫的詳細資訊。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇您要檢視或修改的儲存庫名稱。
3. 展開詳細資訊以查看下列項目：
   - 儲存庫的網域。選擇網域名稱以進一步了解。
   - 儲存庫的資源政策。選擇套用儲存庫政策以新增儲存庫政策。

- 儲存庫的 Amazon Resource Name (ARN)。
- 如果您的儲存庫有外部連線，您可以選擇連線以進一步了解。儲存庫只能有一個外部連線。如需詳細資訊，請參閱將 CodeArtifact 儲存庫連線至公有儲存庫。
- 如果您的儲存庫有上游儲存庫，您可以選擇其中一個來查看其詳細資訊。儲存庫最多可以有 10 個直接上游儲存庫。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。

> **Note**
>
> 儲存庫可以具有外部連線或上游儲存庫，但不能同時擁有兩者。

4. 在套件中，您可以查看此儲存庫可用的任何套件。選擇套件以進一步了解。
5. 選擇檢視連線指示，然後選擇套件管理員，以了解如何使用 CodeArtifact 進行設定。
6. 選擇套用儲存庫政策，以更新或新增資源政策到您的儲存庫。如需詳細資訊，請參閱 儲存庫政策。
7. 選擇編輯以新增或更新下列項目。

   - 儲存庫描述。
   - 與儲存庫相關聯的標籤。
   - 如果您的儲存庫有外部連線，您可以變更其連線的公有儲存庫。否則，您可以將一或多個現有儲存庫新增為上游儲存庫。請求套件時，按您希望 CodeArtifact 排定優先順序的順序排列。如需詳細資訊，請參閱上游儲存庫優先順序。

檢視或修改儲存庫組態 (AWS CLI)

若要在 CodeArtifact 中檢視儲存庫目前的組態，請使用 `describe-repository`命令。

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

輸出範例：

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

修改儲存庫上游組態

上游儲存庫允許套件管理員用戶端使用單一 URL 端點存取多個儲存庫中包含的套件。若要新增或變更儲存庫的上游關係，請使用 update-repository命令。

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

輸出範例：

```
{
    "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**
>
> 若要新增上游儲存庫，您必須擁有上游儲存庫上 AssociateWithDownstreamRepository動作的許可。

若要移除儲存庫的上游關係，請使用空白清單做為 --upstreams選項的引數。

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

輸出範例：

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

儲存庫政策

CodeArtifact 使用資源型許可來控制存取。以資源為基礎的權限可讓您指定誰可存取儲存庫以及他們可以執行的動作。根據預設，只有儲存庫擁有者可存取儲存庫。您可以套用政策文件，允許其他 IAM 主體存取您的儲存庫。

如需詳細資訊，請參閱以資源為基礎的政策和以身分為基礎的政策和以資源為基礎的政策。

建立資源政策以授予讀取存取權

資源政策是 JSON 格式的文字檔案。檔案必須指定委託人（演員）、一或多個動作和效果 (Allow 或 Deny)。例如，下列資源政策會授予帳戶從儲存庫下載套件的123456789012許可。

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

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

由於政策只會針對其連接的儲存庫評估操作，因此您不需要指定資源。由於資源是隱含的，您可以將 Resource 設定為 *。為了讓套件管理員從此儲存庫下載套件，也需要建立跨帳戶存取的網域政策。網域政策必須至少將codeartifact:GetAuthorizationToken許可授予委託人。如需授予跨帳戶存取的完整網域政策範例，請參閱此 網域政策範例。

> **Note**
>
> codeartifact:ReadFromRepository 動作只能在儲存庫資源上使用。您無法使用 將套件的 Amazon Resource Name (ARN) 做為資源codeartifact:ReadFromRepository，做為允許讀取儲存庫中套件子集的動作。指定的委託人可以讀取儲存庫中的所有套件，也可以讀取它們。

由於儲存庫中指定的唯一動作是 ReadFromRepository，來自 帳戶的使用者和角色1234567890可以從儲存庫下載套件。不過，他們無法對其執行其他動作（例如，列出套件名稱和版本）。一般而言，除了 之外，您還會授予下列政策的許可，ReadFromRepository因為從儲存庫下載套件的使用者也需要以其他方式與其互動。

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

設定政策

建立政策文件後，請使用 put-repository-permissions-policy命令將其連接至儲存庫：

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

當您呼叫 時put-repository-permissions-policy，評估許可時，會忽略儲存庫上的資源政策。這可確保網域擁有者無法將自己鎖定在儲存庫之外，這會使他們無法更新資源政策。

> **Note**
>
> 您無法將許可授予另一個 AWS 帳戶，以使用資源政策更新儲存庫上的資源政策，因為在呼叫 put-repository-permissions-policy 時會忽略資源政策。

輸出範例：

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

命令的輸出包含儲存庫資源的 Amazon Resource Name (ARN)、政策文件的完整內容，以及修訂識別符。您可以使用 put-repository-permissions-policy --policy-revision選項將修訂識別符傳遞至 。這可確保文件的已知修訂被覆寫，而不是由另一個寫入器設定的較新版本。

讀取政策

使用 `get-repository-permissions-policy`命令讀取政策文件的現有版本。若要格式化輸出以提高可讀性，請使用 `--output`和 `--query policy.document`搭配 Python `json.tool`模組。

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

輸出範例：

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

刪除政策

使用 `delete-repository-permissions-policy`命令從儲存庫刪除政策。

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

```
        --repository my_repo
```

輸出的格式與 get-repository-permissions-policy命令的格式相同。

將讀取存取權授予委託人

當您將帳戶的根使用者指定為政策文件中的委託人時，您會授予該帳戶中所有使用者和角色的存取權。若要限制對所選使用者或角色的存取，請在政策的 Principal區段中使用其 ARN。例如，使用下列項目將讀取存取權授予帳戶 bob中的 IAM 使用者123456789012。

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

授予套件的寫入存取權

codeartifact:PublishPackageVersion 動作用於控制發佈新版本套件的許可。與此動作搭配使用的資源必須是套件。CodeArtifact 套件 ARNs的格式如下。

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

下列範例顯示網域 中my_repo儲存庫ui中範圍@parity和名稱為 npm 套件的 ARNmy_domain。

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

沒有範圍之 npm 套件的 ARN 具有命名空間欄位的空白字串。例如，以下是網域 中儲存my_repo庫react中沒有範圍和名稱之套件的 ARNmy_domain。

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

下列政策授予帳戶在 my_repo儲存庫@parity/ui中發佈 版本的123456789012許可。

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

Important

若要授予發佈 Maven 和 NuGet 套件版本的許可，除了 之外，請新增下列許可codeartifact:PublishPackageVersion。

1. NuGet： codeartifact:ReadFromRepository 並指定儲存庫資源
2. Maven： codeartifact:PutPackageMetadata

由於此政策指定網域和儲存庫做為資源的一部分，因此它僅允許在連接到該儲存庫時發佈。

授予對儲存庫的寫入存取權

您可以使用萬用字元來授予儲存庫中所有套件的寫入許可。例如，使用以下政策授予 帳戶寫入儲存my_repo庫中所有套件的許可。

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

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

儲存庫與網域政策之間的互動

CodeArtifact 支援網域和儲存庫的資源政策。資源政策是選用的。每個網域可能有一個政策，而網域中的每個儲存庫可能都有自己的儲存庫政策。如果網域政策和儲存庫政策都存在，則在決定是否允許或拒絕對 CodeArtifact 儲存庫的請求時，會評估兩者。網域和儲存庫政策正在評估使用下列規則：

- 執行 ListDomains 或 ListRepositories 等帳戶層級操作時，不會評估資源政策。
- 執行 DescribeDomain 或 ListRepositoriesInDomain 等網域層級操作時，不會評估儲存庫政策。
- 執行 PutDomainPermissionsPolicy 時，不會評估網域政策。請注意，此規則可防止鎖定。
- 執行 PutRepositoryPermissionsPolicy 時會評估網域政策，但不會評估儲存庫政策。
- 任何政策中的明確拒絕都會覆寫另一個政策中的允許。
- 一個資源政策只需要明確允許。如果網域政策允許動作，則從儲存庫政策省略動作不會導致隱含拒絕。
- 當資源政策不允許動作時，結果會是隱含拒絕，除非呼叫主體的帳戶是網域擁有者或儲存庫管理員帳戶，且身分型政策允許該動作。

資源政策在單一帳戶案例中用於授予存取權時為選用，其中用於存取儲存庫的發起人帳戶與網域擁有者和儲存庫管理員帳戶相同。在呼叫者帳戶與網域擁有者或儲存庫管理員帳戶不同的情況下，需要資源政策才能授予存取權。CodeArtifact 中的跨帳戶存取遵循跨帳戶存取的一般 IAM 規則，如 IAM 使用者指南中的判斷是否允許跨帳戶請求中所述。

- 網域擁有者帳戶中的主體可能會透過身分型政策獲得網域中任何儲存庫的存取權。請注意，在此情況下，網域或儲存庫政策不需要明確允許。
- 網域擁有者帳戶中的主體可以透過網域或儲存庫政策授予對任何儲存庫的存取權。請注意，在此情況下，身分型政策不需要明確允許。

- 儲存庫管理員帳戶中的主體可以透過身分型政策授予儲存庫的存取權。請注意，在此情況下，網域或儲存庫政策不需要明確允許。
- 只有在至少一個資源政策和至少一個以身分為基礎的政策允許時，沒有政策明確拒絕動作時，才會授予另一個帳戶中的委託人存取權。

在 CodeArtifact 中標記儲存庫

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在 CodeArtifact 中將標籤套用至儲存庫。如需 CodeArtifact 資源標記、使用案例、標籤索引鍵和值限制，以及支援的資源類型的詳細資訊，請參閱標記 資源。

您可以在建立儲存庫時使用 CLI 來指定標籤。您可以使用 主控台或 CLI 來新增或移除標籤，並更新儲存庫中標籤的值。每個儲存庫最多可新增 50 個標籤。

主題

- 標籤儲存庫 (CLI)
- 標籤儲存庫 （ 主控台 ）

標籤儲存庫 (CLI)

您可以使用 CLI 來管理儲存庫標籤。

主題

- 將標籤新增至儲存庫 (CLI)
- 檢視儲存庫的標籤 (CLI)
- 編輯儲存庫的標籤 (CLI)
- 從儲存庫移除標籤 (CLI)

將標籤新增至儲存庫 (CLI)

您可以使用 主控台或 AWS CLI 來標記儲存庫。

若要在建立儲存庫將標籤新增到儲存庫，請參閱建立 儲存庫。

在這些步驟中，我們假設您已經安裝新版 AWS CLI 或更新到最新版本。如需詳細資訊，請參閱安裝 AWS Command Line Interface。

在終端機或命令列，執行 tag-resource 命令，指定您要新增標籤之儲存庫的 Amazon Resource Name (ARN)，和您想新增之標籤的索引鍵和值。

> **Note**
>
> 若要取得儲存庫的 ARN，請執行 `describe-repository`命令：
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

您可以新增多個標籤到儲存庫。例如，若要在名為 *my_domain ####### my_repo* 的儲存庫加上兩個標籤、標籤鍵名為 *key1* 且標籤值為 *value1*，以及標籤鍵名為 *key2* 且標籤值為 *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
```

如果成功，此命令沒有輸出。

檢視儲存庫的標籤 (CLI)

請依照下列步驟使用 AWS CLI 來檢視儲存庫的 AWS 標籤。若未新增標籤，傳回的清單空白。

在終端機或命令列上執行 list-tags-for-resource 命令。

> **Note**
>
> 若要取得儲存庫的 ARN，請執行 `describe-repository`命令：
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

例如，若要在名為 *my_domain ### ARN ########## my_repo* 之儲存庫的標籤索引鍵和標籤值清單： `arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo`

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

若成功，此命令會傳回類似如下的資訊：

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

編輯儲存庫的標籤 (CLI)

請依照下列步驟，使用 AWS CLI 編輯儲存庫的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。

在終端機或命令列，執行 tag-resource命令，指定您要更新標籤的儲存庫 ARN，並指定標籤索引鍵和標籤值。

> **Note**
>
> 若要取得儲存庫的 ARN，請執行 `describe-repository`命令：
>
> ```
> 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
```

如果成功，此命令沒有輸出。

從儲存庫移除標籤 (CLI)

請依照下列步驟，使用 從儲存庫 AWS CLI 移除標籤。

> **Note**
>
> 如果您刪除 儲存庫，所有標籤關聯會從已刪除的儲存庫中移除。您不需要在刪除儲存庫之前移除標籤。

在終端機或命令列，執行 untag-resource命令，指定您要移除標籤之儲存庫的 ARN，以及您要移除之標籤的標籤索引鍵。

> **Note**
>
> 若要取得儲存庫的 ARN，請執行 describe-repository命令：
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

例如，若要在名為 *my_domain ############ key1 # key2 #### my_repo* 的儲存庫上的多個標籤：

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

如果成功，此命令沒有輸出。移除標籤後，您可以使用 list-tags-for-resource命令檢視儲存庫上剩餘的標籤。

標籤儲存庫（主控台）

您可以使用主控台或 CLI 以標記資源。

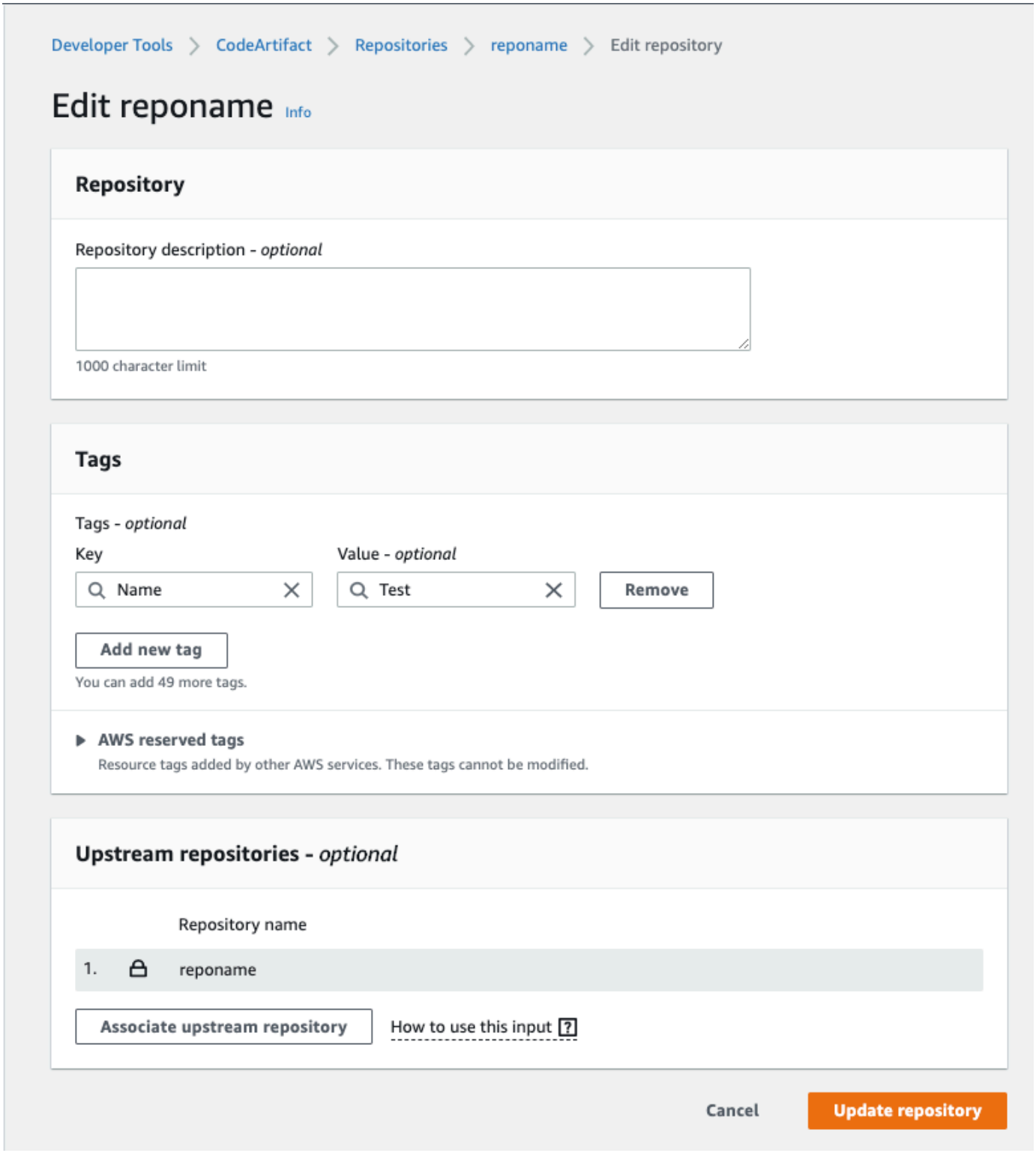
主題

- 將標籤新增至儲存庫（主控台）
- 檢視儲存庫的標籤（主控台）
- 編輯儲存庫的標籤（主控台）
- 從儲存庫移除標籤（主控台）

將標籤新增至儲存庫（主控台）

您可以使用 主控台將標籤新增至現有的儲存庫。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在儲存庫頁面上，選擇要新增標籤的儲存庫。
3. 展開詳細資訊區段。
4. 在儲存庫標籤下，如果儲存庫上沒有標籤，請選擇新增儲存庫標籤。如果儲存庫上有標籤，請選擇檢視和編輯儲存庫標籤。
5. 選擇 Add new tag (新增標籤)。
6. 在金鑰和值欄位中，輸入您要新增的每個標籤的文字。(Value (值) 欄為選用。) 例如，在 Key (索引鍵) 中輸入 **`Name`**。在 Value (值) 中輸入 **`Test`**。



7. (選用) 選擇 Add tag (新增標籤)，新增更多列，然後輸入更多標籤。
8. 選擇更新儲存庫。

檢視儲存庫的標籤 （主控台）

您可以使用 主控台列出現有儲存庫的標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在儲存庫頁面上，選擇您要檢視標籤的儲存庫。
3. 展開詳細資訊區段。
4. 在儲存庫標籤下，選擇檢視和編輯儲存庫標籤。

> **Note**
>
> 如果沒有標籤新增至此儲存庫，主控台會讀取新增儲存庫標籤。

編輯儲存庫的標籤 （主控台）

您可以使用 主控台來編輯已新增至儲存庫的標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在儲存庫頁面上，選擇您要更新標籤的儲存庫。
3. 展開詳細資訊區段。
4. 在儲存庫標籤下，選擇檢視和編輯儲存庫標籤。

> **Note**
>
> 如果沒有標籤新增至此儲存庫，主控台會讀取新增儲存庫標籤。

5. 在 Key (金鑰) 和 Value (加值) 欄，視需要更新每個欄位的值。例如，針對 **`Name`** 索引鍵，在 Value (值) 中將 **`Test`** 變為 **`Prod`**。
6. 選擇更新儲存庫。

從儲存庫移除標籤 （主控台）

您可以使用 主控台從儲存庫刪除標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在儲存庫頁面上，選擇您要移除標籤的儲存庫。
3. 展開詳細資訊區段。
4. 在儲存庫標籤下，選擇檢視和編輯儲存庫標籤。

> **Note**
>
> 如果沒有標籤新增至此儲存庫，主控台會讀取新增儲存庫標籤。

5. 在要刪除的每個標籤的索引鍵和值旁邊，選擇移除。
6. 選擇更新儲存庫。

在 CodeArtifact 中使用上游儲存庫

儲存庫可以讓其他 AWS CodeArtifact 儲存庫做為上游儲存庫。這可讓套件管理員用戶端使用單一儲存庫端點存取包含在多個儲存庫中的套件。

您可以使用 AWS Management Console AWS CLI、 或 SDK，將一或多個上游儲存庫新增至 AWS CodeArtifact 儲存庫。若要將儲存庫與上游儲存庫建立關聯，您必須具有上游儲存庫上`AssociateWithDownstreamRepository`動作的許可。如需詳細資訊，請參閱 使用上游儲存庫建立儲存庫 和 新增或移除上游儲存庫。

如果上游儲存庫與公有儲存庫有外部連線，則位於其下游的儲存庫可以從該公有儲存庫提取套件。例如，假設儲存庫`my_repo`具有名為 的上游儲存庫`upstream`，且`upstream`與公有 npm 儲存庫有外部連線。在這種情況下，連線至 的套件管理員`my_repo`可以從 npm 公有儲存庫提取套件。如需從上游儲存庫或外部連線請求套件的詳細資訊，請參閱 使用上游儲存庫請求套件版本或 從外部連線請求套件。

主題

- 上游儲存庫和外部連線有何不同？
- 新增或移除上游儲存庫
- 將 CodeArtifact 儲存庫連線至公有儲存庫
- 使用上游儲存庫請求套件版本
- 從外部連線請求套件
- 上游儲存庫優先順序
- 上游儲存庫的 API 行為

上游儲存庫和外部連線有何不同？

在 CodeArtifact 中，上游儲存庫和外部連線的行為大致相同，但有一些重要的差異。

1. 您最多可以將 10 個上游儲存庫新增至 CodeArtifact 儲存庫。您只能新增一個外部連線。
2. 有個別的 API 呼叫可新增上游儲存庫或外部連線。
3. 套件保留行為略有不同，因為從上游儲存庫請求的套件會保留在這些儲存庫中。如需詳細資訊，請參閱中繼儲存庫中的套件保留。

新增或移除上游儲存庫

請依照下列各節中的步驟，在 CodeArtifact 儲存庫中新增或移除上游儲存庫。如需上游儲存庫的詳細資訊，請參閱 在 CodeArtifact 中使用上游儲存庫。

本指南包含將其他 CodeArtifact 儲存庫設定為上游儲存庫的相關資訊。如需設定外部連線至公有儲存庫的資訊，例如 npmjs.com、Nuget Gallery、Maven Central 或 PyPI，請參閱新增外部連線。

新增或移除上游儲存庫 （主控台）

執行下列程序中的步驟，使用 CodeArtifact 主控台將儲存庫新增為上游儲存庫。如需使用 新增上游儲存庫的詳細資訊 AWS CLI，請參閱 新增或移除上游儲存庫 (AWS CLI)。

使用 CodeArtifact 主控台新增上游儲存庫

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇包含儲存庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游儲存庫中，選擇關聯上游儲存庫，然後新增要新增為上游儲存庫的儲存庫。您只能在與上游儲存庫相同的網域中新增儲存庫。
6. 選擇更新儲存庫。

使用 CodeArtifact 主控台移除上游儲存庫

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇包含儲存庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游儲存庫中，尋找您要移除的上游儲存庫清單項目，然後選擇取消關聯。

> **Important**
>
> 從 CodeArtifact 儲存庫移除上游儲存庫後，套件管理員將無法存取上游儲存庫或其任何上游儲存庫中的套件。

6. 選擇更新儲存庫。

新增或移除上游儲存庫 (AWS CLI)

您可以使用 AWS Command Line Interface () 新增或移除 CodeArtifact 儲存庫的上游儲存庫AWS CLI。若要這樣做，請使用 `update-repository`命令，並使用 `--upstreams` 參數指定上游儲存庫。

您只能在與上游儲存庫相同的網域中新增儲存庫。

新增上游儲存庫 (AWS CLI)

1. 如果您尚未執行，請依照中的步驟使用 AWS CodeArtifact 設定， AWS CLI 使用 CodeArtifact 設定 。
2. 使用 `aws codeartifact update-repository`命令搭配 `--upstreams`旗標來新增上游儲存庫。

> **Note**
>
> 呼叫 `update-repository`命令會將現有設定的上游儲存庫取代為 `--upstreams`旗標隨附的儲存庫清單。如果您想要新增上游儲存庫並保留現有的儲存庫，您必須在呼叫中包含現有的上游儲存庫。

下列範例命令會將兩個上游儲存庫新增至名為 `my_repo`的儲存庫，該儲存庫位於名為 的網域中`my_domain`。當 CodeArtifact 從儲存庫請求套件時， `--upstreams` 參數中的上游`my_repo`儲存庫順序會決定其搜尋優先順序。如需詳細資訊，請參閱上游儲存庫優先順序。

如需連線至公有、外部儲存庫的相關資訊，例如 npmjs.com 或 Maven Central，請參閱 將 CodeArtifact 儲存庫連線至公有儲存庫。

```
aws codeartifact update-repository \
   --repository my_repo \
```

```
    --domain my_domain \
    --domain-owner 111122223333 \
    --upstreams repositoryName=upstream-1 repositoryName=upstream-2
```

輸出包含上游儲存庫，如下所示。

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

移除上游儲存庫 (AWS CLI)

1. 如果您尚未執行，請依照中的步驟使用 AWS CodeArtifact 設定， AWS CLI 使用 CodeArtifact 設定 。
2. 若要從 CodeArtifact 儲存庫移除上游儲存庫，請使用 update-repository命令搭配 --upstreams旗標。提供給命令的儲存庫清單將是 CodeArtifact 儲存庫的新上游儲存庫集。包含您要保留的現有上游儲存庫，並省略您要移除的上游儲存庫。

   若要從儲存庫移除所有上游儲存庫，請使用 update-repository命令並包含 ，--upstreams而不使用 引數。以下內容會從名為 的儲存庫中移除上游儲存庫my_repo，該儲存庫包含在名為 的網域中my_domain。

```
aws codeartifact update-repository \
   --repository my_repo \
```

```
    --domain my_domain \
    --domain-owner 111122223333 \
    --upstreams
```

輸出顯示 的清單upstreams為空白。

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

將 CodeArtifact 儲存庫連線至公有儲存庫

您可以在 CodeArtifact 儲存庫與外部公有儲存庫之間新增外部連線，例如 https://npmjs.com 或 Maven Central 儲存庫。然後，當您從尚未存在於儲存庫中的 CodeArtifact 儲存庫請求套件時，可以從外部連線擷取套件。這可讓您使用應用程式所使用的開放原始碼相依性。

在 CodeArtifact 中，使用外部連線的預期方法是每個網域有一個儲存庫，並具有與指定公有儲存庫的外部連線。例如，如果您想要連線至 npmjs.com，請在網域中設定一個儲存庫，並外部連線至 npmjs.com，然後設定所有其他儲存庫，其上游為 。如此一來，所有儲存庫都可以使用已從 npmjs.com 擷取的套件，而不是再次擷取和儲存套件。

主題

- 連線至外部儲存庫 （ 主控台 ）
- 連線至外部儲存庫 (CLI)
- 支援的外部連線儲存庫
- 移除外部連線 (CLI)

連線至外部儲存庫（主控台）

當您使用主控台將連線新增至外部儲存庫時，會發生下列情況：

1. 如果外部`-store`儲存庫尚不存在，則會在您的 CodeArtifact 網域中建立儲存庫。這些`-store`儲存庫可做為儲存庫與外部儲存庫之間的中繼儲存庫，並可讓您連線至多個外部儲存庫。
2. 適當的`-store`儲存庫會新增為儲存庫的上游。

下列清單包含 CodeArtifact 中的每個`-store`儲存庫，以及它們連線的個別外部儲存庫。

1. `cargo-store` 已連線至 crates.io。
2. `clojars-store` 已連線至 Clojars 儲存庫。
3. `commonsware-store` 已連線至 CommonsWare Android 儲存庫。
4. `google-android-store` 已連線至 Google Android。
5. `gradle-plugins-store` 已連線至 Gradle 外掛程式。
6. `maven-central-store` 已連線至 Maven Central Repository。
7. `npm-store` 已連線至 npmjs.com。
8. `nuget-store` 已連線至 nuget.org。
9. `pypi-store` 已連線至 Python Packaging Authority。
10. `rubygems-store` 已連線至 RubyGems.org.

連線至外部儲存庫（主控台）

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇包含儲存庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游儲存庫中，選擇關聯上游儲存庫，然後新增以上游身分連接的適當`-store`儲存庫。
6. 選擇更新儲存庫。

將`-store`儲存庫新增為上游儲存庫後，連線至 CodeArtifact 儲存庫的套件管理員可以從個別的外部儲存庫擷取套件。

連線至外部儲存庫 (CLI)

您可以使用 將外部連線直接新增至儲存庫， AWS CLI 以將 CodeArtifact 儲存庫連線至外部儲存庫。這將允許使用者連接到 CodeArtifact 儲存庫或其任何下游儲存庫，從設定的外部儲存庫擷取套件。每個 CodeArtifact 儲存庫只能有一個外部連線。

建議每個網域有一個儲存庫，並具有與指定公有儲存庫的外部連線。若要將其他儲存庫連接到公有儲存庫，請將具有外部連線的儲存庫做為上游連線新增至這些儲存庫。如果您或網域中的其他人已在 主控台中設定外部連線，則您的網域可能已經有一個儲存-store庫，其具有與您要連線之公有儲存庫的外部連線。如需 -store 儲存庫和與主控台連線的詳細資訊，請參閱 連線至外部儲存庫 （主控台）。

將外部連線新增至 CodeArtifact 儲存庫 (CLI)

- 使用 associate-external-connection來新增外部連線。下列範例會將儲存庫連線至 npm 公有登錄檔 http：//npmjs.com。如需支援的外部儲存庫清單，請參閱 支援的外部連線儲存庫。

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

輸出範例：

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

新增外部連線後，請參閱 從外部連線請求套件 以取得使用外部連線從外部儲存庫請求套件的相關資訊。

支援的外部連線儲存庫

CodeArtifact 支援與下列公有儲存庫的外部連線。若要使用 CodeArtifact CLI 指定外部連線，請在執行 `associate-external-connection`命令時，使用 `--external-connection` 參數名稱欄中的值。

儲存庫類型	描述	名稱
Maven	Clojars 儲存庫	`public:maven-clojars`
Maven	CommonsWare Android 儲存庫	`public:maven-commonsware`
Maven	Google Android 儲存庫	`public:maven-googleandroid`
Maven	Gradle 外掛程式儲存庫	`public:maven-gradleplugins`
Maven	Maven Central	`public:maven-central`
npm	npm 公有登錄檔	`public:npmjs`
NuGet	NuGet Gallery	`public:nuget-org`
Python	Python 套件索引	`public:pypi`
Ruby	RubyGems.org	`public:ruby-gems-org`
Rust	Crates.io	`public:crates-io`

移除外部連線 (CLI)

若要移除使用 中的 `associate-external-connection`命令新增的外部連線 AWS CLI，請使用 `disassociate-external-connection`。

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

輸出範例：

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

使用上游儲存庫請求套件版本

當用戶端 （ 例如 npm) 從名為 的 CodeArtifact 儲存庫請求套件版本`my_repo`時，可能會發生下列情況：

- 如果 `my_repo`包含請求的套件版本，則會傳回給用戶端。
- 如果 `my_repo` 不包含請求的套件版本，CodeArtifact 會在 `my_repo`的上游儲存庫中尋找它。如果找到套件版本，則會將其參考複製到 `my_repo`，並將套件版本傳回給用戶端。
- 如果 `my_repo`或其上游儲存庫都不包含套件版本，則會將 HTTP 404 `Not Found`回應傳回給用戶端。

當您使用 `create-repository`或 `update-repository`命令新增上游儲存庫時，傳遞給 `--upstreams` 參數的順序會決定請求套件版本的優先順序。`--upstreams` 依您希望 CodeArtifact 在請求套件版本時使用的順序指定上游儲存庫。如需詳細資訊，請參閱上游儲存庫優先順序。

一個儲存庫允許的直接上游儲存庫數量上限為 10。由於直接上游儲存庫也可以有自己的直接上游儲存庫，因此 CodeArtifact 可以搜尋 10 個以上的儲存庫以取得套件版本。請求套件版本時CodeArtifact 會尋找的儲存庫數目上限為 25。

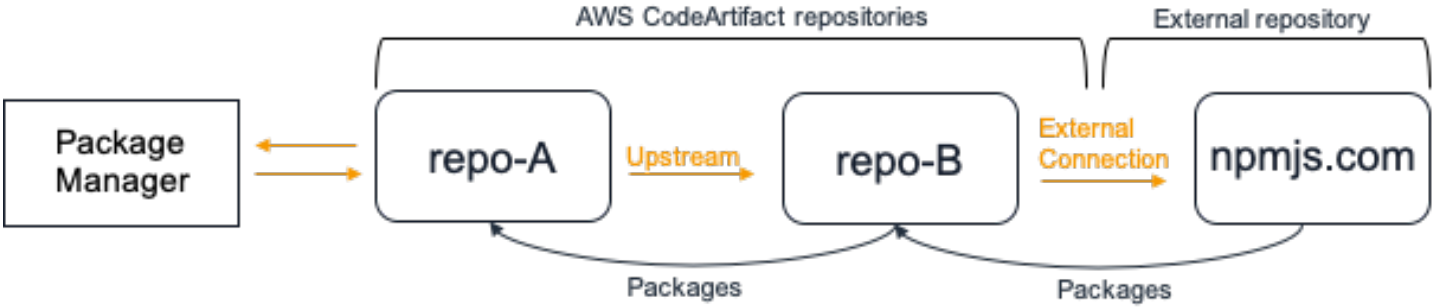
來自上游儲存庫的套件保留

如果在上游儲存庫中找到請求的套件版本，則會保留其參考，並且一律可從下游儲存庫取得。保留的套件版本不受下列任何一項影響：

- 刪除上游儲存庫。
- 中斷上游儲存庫與下游儲存庫的連線。
- 從上游儲存庫刪除套件版本。
- 在上游儲存庫中編輯套件版本 （ 例如，透過新增資產到該儲存庫 ）。

透過上游關係擷取套件

如果 CodeArtifact 儲存庫與具有外部連線的儲存庫有上游關係，則會從外部儲存庫複製不在上游儲存庫中的套件請求。例如，請考慮下列組態：名為 的儲存庫`repo-A`具有名為 的上游儲存庫`repo-B`。`repo-B` 與 https://npmjs.com：//www.microsoft.com 具有外部連線。



如果 npm 設定為使用 `repo-A`儲存庫，則執行 會`npm install`觸發將套件從 https://npmjs.com 複製到 `repo-B`。已安裝的版本也會提取至 `repo-A`。下列範例會安裝 `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
```

執行 之後npm install，只repo-A包含最新版本 (lodash 4.17.20)，因為這是 npm從 擷取的版本repo-A。

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

輸出範例：

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

由於 repo-B 具有與 https://npmjs.com 的外部連線，因此從 https://npmjs.com 匯入的所有套件版本都會存放在 中repo-B。這些套件版本可由與 有上游關係的任何下游儲存庫擷取repo-B。

的內容repo-B提供一種方式，讓您查看隨著時間從 https://npmjs.com 匯入的所有套件和套件版本。例如，若要查看隨著時間匯入的所有lodash套件版本，您可以使用 list-package-versions，如下所示。

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

輸出範例：

```
{
    "package": "lodash",
    "format": "npm",
    "versions": [
        {
            "version": "0.10.0",
            "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
```

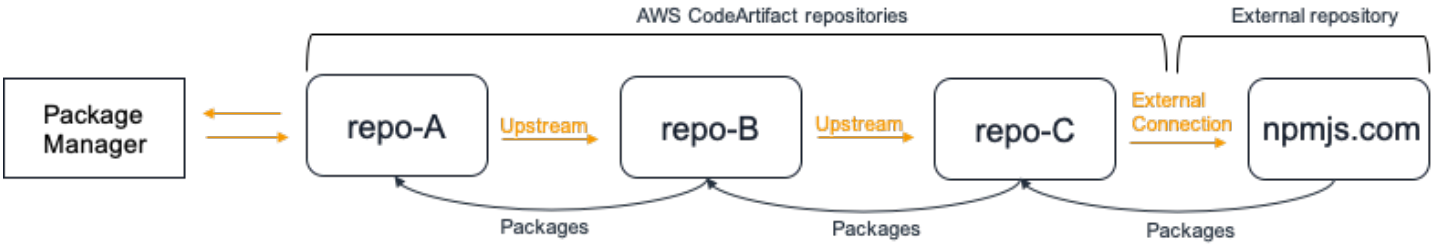
```
        },
        {
            "version": "0.2.2",
            "revision": "REVISION-2-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.2.0",
            "revision": "REVISION-3-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.2.1",
            "revision": "REVISION-4-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.1.0",
            "revision": "REVISION-5-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        }
    ],
    "nextToken": "eyJsaXN0UGFja2FnZVZlcnNpb25zVG9rZW4iOiIwLjIuMiJ9"
}
```

中繼儲存庫中的套件保留

CodeArtifact 允許鏈結上游儲存庫。例如， repo-A可以讓 repo-B做為上游，repo-B也可以讓 repo-C做為上游。此組態可讓 repo-B和 中的套件版本從 repo-C取得repo-A。

當套件管理員連線至儲存庫repo-A並從儲存庫 擷取套件版本時repo-C，套件版本將不會保留在儲存庫 中repo-B。套件版本只會保留在最下游的儲存庫中，在此範例中為 repo-A。它不會保留在任何中繼儲存庫中。對於較長的鏈結也是如此；例如，如果有四個儲存庫 repo-A、repo-C、 repo-B和 ，repo-D以及連線至從 repo-A 擷取套件版本的套件管理員repo-D，則套件版本會保留在中repo-A，但不會保留在 repo-B或 中repo-C。

從外部儲存庫提取套件版本時，套件保留行為類似，但套件版本一律保留在連接外部連線的儲存庫中。例如， `repo-A` 將 `repo-B`作為上游。 `repo-B` 將 `repo-C`作為上游，`repo-C`並將 npmjs.com 設定為外部連線；請參閱下圖。



例如，如果連接到 的套件管理員`repo-A`請求套件版本，例如 lodash 4.17.20 」，而且套件版本不存在於三個儲存庫中的任何一個，則會從 npmjs.com：// 。擷取 lodash 4.17.20 xxxx 時，它將保留在中`repo-A`，因為這是最下游的儲存庫，`repo-C`而且它已連接到 npmjs.com 的外部連線。lodash 的 4.17.20 不會保留在 中`repo-B`，因為這是中繼儲存庫。

從外部連線請求套件

下列各節說明如何在請求套件時，從外部連線和預期的 CodeArtifact 行為請求套件。

主題

- 從外部連線擷取套件
- 外部連線延遲
- 無法使用外部儲存庫時的 CodeArtifact 行為
- 新套件版本的可用性
- 匯入具有多個資產的套件版本

從外部連線擷取套件

若要在將套件新增至 CodeArtifact 儲存庫後從外部連線擷取套件，如 中所述將 CodeArtifact 儲存庫連線至公有儲存庫，請將套件管理員設定為使用您的儲存庫並安裝套件。

> **Note**
>
> 下列指示使用 npm，若要檢視其他套件類型的組態和使用說明，請參閱 搭配 Maven 使用 CodeArtifact 、 搭配 NuGet 使用 CodeArtifact 或 搭配 Python 使用 CodeArtifact 。

從外部連線擷取套件

1. 使用 CodeArtifact 儲存庫設定和驗證套件管理員。對於 npm，請使用下列 `aws codeartifact login` 命令。

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

2. 從公有儲存庫請求套件。對於 npm，請使用下列`npm install`命令，將 *lodash* 取代為您要安裝的套件。

```
npm install lodash
```

3. 將套件複製到 CodeArtifact 儲存庫後，您可以使用 `list-packages`和 `list-package-versions`命令來檢視套件。

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

輸出範例：

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

`list-package-versions` 命令會列出複製到 CodeArtifact 儲存庫的所有套件版本。

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

輸出範例：

```
{
    "defaultDisplayVersion: "1.2.5"
    "format": "npm",
```

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

外部連線延遲

使用外部連線從公有儲存庫擷取套件時，從公有儲存庫擷取套件，以及將套件存放在 CodeArtifact 儲存庫時，會發生延遲。例如，假設您已安裝 npm 套件 "lodash" 的 1.2.5 版，如 中所述從外部連線擷取套件。雖然 lodash `npm install lodash` 命令已成功完成，但套件版本可能尚未出現在 CodeArtifact 儲存庫中。套件版本通常需要約 3 分鐘才會出現在您的儲存庫中，但偶爾可能需要更長的時間。

由於此延遲，您可能已成功擷取套件版本，但可能還無法在 CodeArtifact 主控台或呼叫 ListPackages 和 ListPackageVersions API 操作時，在儲存庫中查看版本。一旦 CodeArtifact 非同步保留套件版本，它就會顯示在主控台中，並透過 API 請求顯示。

無法使用外部儲存庫時的 CodeArtifact 行為

外部儲存庫偶爾會遇到中斷，這表示 CodeArtifact 無法從中擷取套件，或擷取套件比平常慢得多。發生這種情況時，已從外部儲存庫提取npmjs.com並存放在 CodeArtifact 儲存庫中的套件版本將繼續可從 CodeArtifact 下載。不過，即使已設定該儲存庫的外部連線，尚未存放在 CodeArtifact 中的套件仍可能無法使用。例如，您的 CodeArtifact 儲存庫可能包含 npm 套件版本`lodash 4.17.19` ，因為這是您目前在應用程式中使用的內容。當您想要升級到 時`4.17.20`，CodeArtifact 通常會從 npmjs.com：// www.healthnet.com 擷取該新版本，並將其存放在 CodeArtifact 儲存庫中。不過，如果 npmjs.com 發生中斷，則無法使用此新版本。唯一的解決方法是在 npmjs.com 復原後再試一次。

外部儲存庫中斷也可能會影響將新套件版本發佈至 CodeArtifact。在已設定外部連線的儲存庫中，CodeArtifact 不允許發佈已存在於外部儲存庫中的套件版本。如需詳細資訊，請參閱套件概觀。不過，在極少數情況下，外部儲存庫中斷可能表示 CodeArtifact 沒有外部儲存庫中存在哪些套件和套件版本up-to-date。在此情況下，CodeArtifact 可能會允許發佈其通常會拒絕的套件版本。

新套件版本的可用性

若要讓 npmjs.com 等公有儲存庫中的套件版本可透過 CodeArtifact 儲存庫使用，必須先將其新增至區域套件中繼資料快取。此快取由 CodeArtifact 在每個 AWS 區域中維護，並包含描述支援公有儲存庫內容的中繼資料。由於此快取，新套件版本發佈到公有儲存庫到 CodeArtifact 提供的版本之間會有延遲。此延遲會因套件類型而異。

對於 npm、Python 和 Nuget 套件，從新套件版本發佈至 npmjs.com：// 、 pypi.org：// https：// https：//https：//https：//https：//https：//https：//https：//https：///https：/nuget.org：// CodeArtifact https：///https：//https：//https：/https：///http：///http：/http：/http：//http：/http：// http：////http：/http：///http//http；http；；；；；；http；；；CodeArtifact 會自動同步這兩個儲存庫的中繼資料，以確保快取是最新的。

對於 Maven 套件，從新套件版本發佈到公有儲存庫，以及從 CodeArtifact 儲存庫安裝時，可能會有最多 3 小時的延遲。CodeArtifact 最多每 3 小時會檢查一次套件的新版本。在 3 小時快取生命週期到期後，對指定套件名稱的第一個請求會導致該套件的所有新版本都匯入區域快取。

對於常用的 Maven 套件，新版本通常會每 3 小時匯入一次，因為請求率高表示快取通常會在快取生命週期過期後立即更新。對於不常使用的套件，快取在從 CodeArtifact 儲存庫請求套件版本之前，不會有最新版本。在第一個請求中，只有先前匯入的版本可從 CodeArtifact 取得，但此請求會導致快取更新。在後續請求中，新版本的套件將新增至快取，並可供下載。

匯入具有多個資產的套件版本

Maven 和 Python 套件在每個套件版本中可以有多個資產。這使得這些格式的匯入套件比 npm 和 NuGet 套件更為複雜，每個套件版本只有一個資產。如需針對這些套件類型匯入哪些資產，以及如何提供新新增資產的說明，請參閱 從上游和外部連線請求 Python 套件和 從上游和外部連線請求 Maven 套件。

上游儲存庫優先順序

當您從具有一或多個上游儲存庫的儲存庫請求套件版本時，其優先順序會對應至呼叫 `create-repository`或 `update-repository`命令時列出的順序。找到請求的套件版本時，即使搜尋未搜尋所有上游儲存庫，搜尋也會停止。如需詳細資訊，請參閱新增或移除上游儲存庫 (AWS CLI)。

使用 `describe-repository`命令查看優先順序。

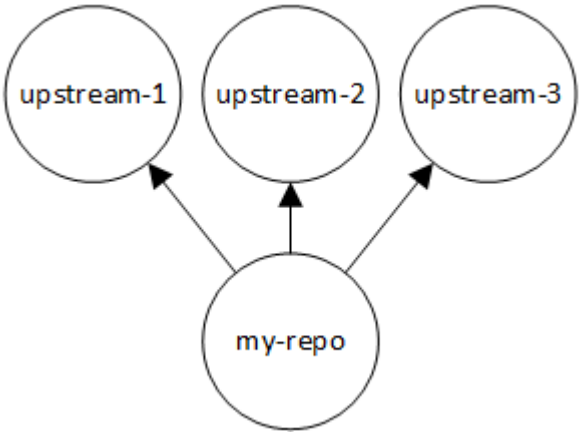
```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-
owner 111122223333
```

結果可能如下。它顯示上游儲存庫優先順序為upstream-1第一、upstream-2第二和upstream-3第三。

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

簡單優先順序範例

在下圖中，儲存my_repo庫有三個上游儲存庫。上游儲存庫的優先順序為upstream-1、upstream-2、upstream-3。



中套件版本的請求會依下列順序my_repo搜尋儲存庫，直到找到為止，或直到 HTTP 404 Not Found回應傳回用戶端為止：

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

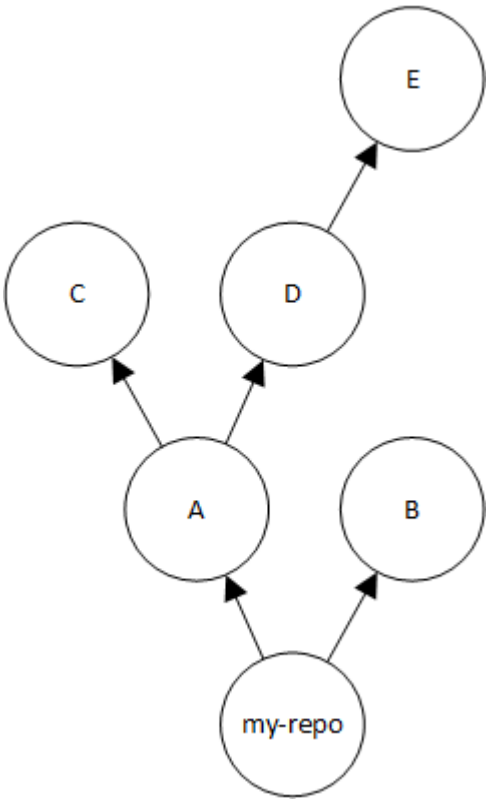
如果找到套件版本，搜尋會停止，即使它未查看所有上游儲存庫。例如，如果在 中找到套件版本upstream-1，搜尋會停止，而 CodeArtifact 不會在 upstream-2或 中尋找upstream-3。

當您使用 AWS CLI 命令list-package-versions列出 中的套件版本時my_repo，它只會在 中尋找my_repo。它不會列出上游儲存庫中的套件版本。

複雜優先順序範例

如果上游儲存庫有自己的上游儲存庫，則在移至下一個上游儲存庫之前，會使用相同的邏輯來尋找套件版本。例如，假設您的儲存my_repo庫有兩個上游儲存庫 A和 B。如果儲存庫A有上游儲存庫， 中套件版本的請求會my_repo先在 中尋找my_repo，再於 中尋找A，然後於 的上游儲存庫中尋找A，以此類推。

在下圖中，儲存my_repo庫包含上游儲存庫。上游儲存庫A有兩個上游儲存庫，並D有一個上游儲存庫。圖表中相同層級的上游儲存庫會以優先順序顯示，從左到右 （儲存庫A的優先順序高於儲存庫 B，而儲存庫C的優先順序高於儲存庫 D)。



在此範例中， 中套件版本的請求會依下列順序在儲存庫中my_repo尋找，直到找到為止，或直到套件管理員傳回 HTTP 404 Not Found回應給用戶端為止：

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

上游儲存庫的 API 行為

當您在連線至上游儲存庫的儲存庫上呼叫特定 CodeArtifact APIs 時，行為可能會有所不同，取決於套件或套件版本是否存放在目標儲存庫或上游儲存庫中。這些 APIs的行為記錄在此處。

如需 CodeArtifact APIs的詳細資訊，請參閱 CodeArtifact API 參考。

如果目標儲存庫中不存在指定的套件版本，則參考套件或套件版本的大多數 APIs 都會傳回ResourceNotFound錯誤。即使上游儲存庫中有套件或套件版本，也是如此。實際上，呼叫這些APIs時會忽略上游儲存庫。這些 APIs為：

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

為了示範此行為，我們有兩個儲存庫： target-repo和 upstream-repo。 target-repo 是空的，並upstream-repo已設定為上游儲存庫。 upstream-repo包含 npm 套件 lodash。

在 上呼叫upstream-repo包含 lodash套件的 DescribePackageVersion API 時，我們會取得下列輸出：

```
{
    "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"
    }
```

在 上呼叫相同的 API 時target-repo，如果該 API 是空的，但已upstream-repo設定為上游，我們會取得下列輸出：

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

CopyPackageVersions API 的行為不同。根據預設，CopyPackageVersionsAPI 只會複製存放在目標儲存庫中的套件版本。如果套件版本存放在上游儲存庫中，但未存放在目標儲存庫中，則不會複製。若要包含僅存放在上游儲存庫中的套件的套件版本，請在 API 請求true中將 的值設定為includeFromUpstream 。

如需 CopyPackageVersions API 的詳細資訊，請參閱 在儲存庫之間複製套件。

在 CodeArtifact 中使用套件

下列主題說明如何使用 CodeArtifact CLI 和 API 對套件執行動作。

主題

- 套件概觀
- 列出套件名稱
- 列出套件版本
- 列出套件版本資產
- 下載套件版本資產
- 在儲存庫之間複製套件
- 刪除套件或套件版本
- 檢視和更新套件版本詳細資訊和相依性
- 更新套件版本狀態
- 編輯套件原始伺服器控制項

套件概觀

套件是軟體和中繼資料的套件，用於解析相依性和安裝軟體。在 CodeArtifact 中，套件包含套件名稱、選用的命名空間，例如 `@types`中的 `@types/node`、一組套件版本，以及套件層級中繼資料，例如 npm 標籤。

內容

- 支援的套件格式
- 套件發佈
  - 發佈許可
  - 覆寫套件資產
  - 私有套件和公有儲存庫
  - 發佈修補的套件版本
  - 發佈的資產大小限制
  - 發佈延遲
- 套件版本狀態

- 套件名稱、套件版本和資產名稱標準化

支援的套件格式

AWS CodeArtifact 支援 Cargo、 generic、Maven、npm、NuGet、PyPI、Ruby、Swift 套件格式。

套件發佈

您可以使用 npm、、twine、、 MavenGradlenuget和 等工具，將任何支援套件格式的新版本發佈至 CodeArtifact 儲存庫dotnet。

發佈許可

您的 AWS Identity and Access Management (IAM) 使用者或角色必須具有發佈至目的地儲存庫的許可。發佈套件需要下列許可：

- 貨物： `codeartifact:PublishPackageVersion`
- 一般： `codeartifact:PublishPackageVersion`
- Maven： `codeartifact:PublishPackageVersion` 和 `codeartifact:PutPackageMetadata`
- npm： `codeartifact:PublishPackageVersion`
- NuGet： `codeartifact:PublishPackageVersion` 和 `codeartifact:ReadFromRepository`
- Python：`codeartifact:PublishPackageVersion`
- Ruby： `codeartifact:PublishPackageVersion`
- Swift： `codeartifact:PublishPackageVersion`

在上述許可清單中，您的 IAM 政策必須指定 `codeartifact:PublishPackageVersion`和 `codeartifact:PutPackageMetadata`許可package的資源。它還必須指定`codeartifact:ReadFromRepository`許可repository的資源。

如需 CodeArtifact 中許可的詳細資訊，請參閱 AWS CodeArtifact 許可參考。

覆寫套件資產

您無法重新發佈已存在且具有不同內容的套件資產。例如，假設您已使用 JAR 資產 發佈 Maven 套件`mypackage-1.0.jar`。只有在舊資產和新資產的檢查總和相同時，您才能再次發佈該資產。若要

使用新內容重新發佈相同的資產，請先使用 delete-package-versions命令刪除套件版本。嘗試以不同的內容重新發佈相同的資產名稱會導致 HTTP 409 衝突錯誤。

對於支援多個資產 （ 一般、PyPI 和 Maven) 的套件格式，您可以將不同名稱的新資產新增至現有的套件版本，前提是您具有必要的許可。對於一般套件，只要套件版本處於 Unfinished 狀態，您就可以新增資產。由於 npm 僅支援每個套件版本的單一資產，若要以任何方式修改發佈的套件版本，您必須先使用 將其刪除delete-package-versions。

如果您嘗試重新發佈已存在的資產 （ 例如 mypackage-1.0.jar)，且已發佈資產和新資產的內容相同，則操作會成功，因為操作是等冪的。

私有套件和公有儲存庫

CodeArtifact 不會將儲存在 CodeArtifact 儲存庫中的套件發佈至公有儲存庫，例如 npmjs.com 或 Maven Central。CodeArtifact 會將套件從公有儲存庫匯入 CodeArtifact 儲存庫，但絕不會朝其他方向移動套件。您發佈至 CodeArtifact 儲存庫的套件會保持私有，且僅適用於您授予存取權 AWS 的帳戶、角色和使用者。

發佈修補的套件版本

有時您可能想要發佈修改過的套件版本，可能是公有儲存庫中可用的版本。例如，您可能在名為 的關鍵應用程式相依性中發現錯誤mydep 1.1，而且您需要比套件廠商可以檢閱並接受變更更快修正。如前所述，如果公有儲存 CodeArtifact 庫可透過上游儲存庫和外部連線從 CodeArtifact 儲存庫存取，則 CodeArtifact 會防止您在 CodeArtifact 儲存庫mydep 1.1中發佈。

若要解決此問題，請將套件版本發佈至無法存取公有儲存庫的不同 CodeArtifact 儲存庫。然後使用 copy-package-versions API 將 的修補版本複製到您要使用它的 mydep 1.1 CodeArtifact 儲存庫。

發佈的資產大小限制

可發佈的套件資產大小上限受限於 中顯示的資產檔案大小上限配額AWS CodeArtifact 中的配額。例如，您無法發佈大於目前資產檔案大小最大配額的 Maven JAR 或 Python 輪。如果您需要在 CodeArtifact 中存放較大的資產，請請求增加配額。

除了資產檔案大小配額上限之外，npm 套件的發佈請求大小上限為 2 GB。此限制與資產檔案大小最大配額無關，且無法隨著配額增加而提高。在 npm 發佈請求 (HTTP PUT) 中，套件中繼資料和 npm 套件 tar 封存的內容會綁定在一起。因此，可發佈的 npm 套件實際大小上限會有所不同，取決於包含中繼資料的大小。

> **Note**
>
> 發佈的 npm 套件的大小上限為小於 2 GB。

發佈延遲

發佈至 CodeArtifact 儲存庫的套件版本通常可在一秒內下載。例如，如果您使用 將 npm 套件版本發佈至 CodeArtifactnpm publish，則該版本應該可在一秒內供 npm install命令使用。不過，發佈可能不一致，有時可能需要更長的時間。如果您必須在發佈後立即使用套件版本，請使用重試以確保下載的可靠性。例如，發佈套件版本後，如果剛發佈的套件版本最初在第一次下載嘗試時不可用，請重複下載最多三次。

> **Note**
>
> 從公有儲存庫匯入套件版本通常需要比發佈更長的時間。如需詳細資訊，請參閱外部連線延遲。

套件版本狀態

CodeArtifact 中的每個套件版本都有一個狀態，描述套件版本的目前狀態和可用性。您可以在 AWS CLI 和 SDK 中變更套件版本狀態。如需詳細資訊，請參閱更新套件版本狀態。

以下是套件版本狀態的可能值：

- 已發佈 – 套件版本已成功發佈，可以使用套件管理員請求。套件版本會包含在傳回給套件管理員的套件版本清單中，例如 的輸出npm view <package-name> versions。套件版本的所有資產皆可從 儲存庫取得。
- 未完成 – 用戶端已上傳套件版本的一或多個資產，但尚未將其移至 Published 狀態來完成。目前只有一般和 Maven 套件版本可以具有 的狀態Unfinished。對於 Maven 套件，當用戶端為套件版本上傳一或多個資產，但不為包含該版本的套件發佈maven-metadata.xml檔案時，就會發生這種情況。當 Maven 套件版本未完成時，不會包含在傳回給用戶端的版本清單中，例如 mvn或 gradle，因此無法做為組建的一部分使用。在呼叫 PublishPackageVersion API 時提供 unfinished旗標，即可刻意將一般套件保持在 Unfinished 狀態。您可以省略 unfinished旗標，或呼叫 UpdatePackageVersionsStatus API，將一般套件變更為 Published 狀態。
- 未列出 – 套件版本的資產可從儲存庫下載，但套件版本不包含在傳回套件管理員的版本清單中。例如，對於 npm 套件， 的輸出npm view <package-name> versions將不會包含套件版本。這

表示 npm 的相依性解析邏輯不會選取套件版本，因為版本不會出現在可用版本清單中。不過，如果 npm `package-lock.json`檔案中已參考未列出的套件版本，仍然可以下載並安裝，例如執行時`npm ci`。

- 已封存 – 無法再下載套件版本的資產。套件版本不會包含在傳回套件管理員的版本清單中。由於資產無法使用，用戶端對套件版本的使用會遭到封鎖。如果您的應用程式建置取決於更新為已封存的版本，則建置會中斷，假設套件版本尚未在本機快取。您無法使用套件管理員或建置工具來重新發佈封存套件版本，因為它仍存在於儲存庫中，但您可以使用 UpdatePackageVersionsStatus API 將套件版本的狀態變更回未列出或已發佈。
- 已處置 – 套件版本不會顯示在清單中，且資產無法從儲存庫下載。Disposed 和 Archived 之間的主要區別在於，狀態為 Disposed 時，CodeArtifact 會永久刪除套件版本的資產。因此，您無法將套件版本從已處置移至已封存、未列出或已發佈。套件版本無法再使用，因為已刪除資產。套件版本標示為已處置後，您將不再支付套件資產的儲存費用。

呼叫 list-package-versions 時，預設會傳回所有狀態的套件版本，沒有`--status`參數。

除了先前列出的狀態之外，也可以使用 DeletePackageVersions API 刪除套件版本。刪除後，套件版本不再位於儲存庫中，您可以使用套件管理員或建置工具自由重新發佈該套件版本。刪除套件版本後，您將不再支付該套件版本資產的儲存費用。

套件名稱、套件版本和資產名稱標準化

CodeArtifact 會先標準化套件名稱、套件版本和資產名稱再儲存，這表示 CodeArtifact 中的名稱或版本可能與發佈套件時提供的名稱或版本不同。如需如何在 CodeArtifact 中針對每個套件類型標準化名稱和版本的詳細資訊，請參閱下列文件：

- Python 套件名稱標準化
- NuGet 套件名稱、版本和資產名稱標準化

CodeArtifact 不會在其他套件格式上執行標準化。

列出套件名稱

使用 CodeArtifact 中的 `list-packages`命令，取得儲存庫中所有套件名稱的清單。此命令只會傳回套件名稱，而不會傳回版本。

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

輸出範例：

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

列出 npm 套件名稱

若要僅列出 npm 套件的名稱，請將 `--format`選項的值設定為 npm。

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

若要列出命名空間中的 npm 套件 (npm 範圍)，請使用 `--namespace`和 `--format`選項。

> **Important**
>
> `--namespace` 選項的值不應包含前置 `@`。若要搜尋命名空間 `@types`，請將 值設定為 `##`。

> **Note**
>
> `--namespace` 選項會依命名空間字首篩選。任何範圍開頭為傳遞給 `--namespace` 選項之值的 npm 套件，都會在`list-packages`回應中傳回。

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

輸出範例：

```
{
    "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"

        }
    ]
}
```

列出 Maven 套件名稱

若要僅列出 Maven 套件的名稱，請將 `--format`選項的值設定為 `maven`。您也必須在 `--namespace`選項中指定 Maven 群組 ID。

> **Note**
>
> `--namespace` 選項會依命名空間字首篩選。任何範圍開頭為傳遞給 `--namespace` 選項之值的 npm 套件，都會在`list-packages`回應中傳回。

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

輸出範例：

```
{
    "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"

        }
    ]
}
```

列出 Python 套件名稱

若要僅列出 Python 套件的名稱，請將 `--format`選項的值設定為 `pypi`。

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

依套件名稱字首篩選

若要傳回以指定字串開頭的套件，您可以使用 `--package-prefix`選項。

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

輸出範例：

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

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

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

        }
    ]
}
```

支援的搜尋選項組合

您可以任意組合使用 `--namespace`、 `--format`和 `--package-prefix`選項，但 `--namespace` 本身無法使用。搜尋範圍開頭為 的所有 npm 套件`@types`，需要指定 `--format`選項。`--namespace` 單獨使用 本身會導致錯誤。

也不支援使用這三個選項中的任何一個，`list-packages`而且 會傳回儲存庫中所有格式的所有套件。

格式輸出

您可以使用所有 AWS CLI 命令可用的參數，讓`list-packages`回應精簡且更易於讀取。使用 `--query` 參數來指定每個傳回套件版本的格式。使用 `--output` 參數將回應格式化為純文字。

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

輸出範例：

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

如需詳細資訊，請參閱 AWS Command Line Interface 使用者指南中的控制 AWS CLI的命令輸出。

預設值和其他選項

根據預設， 傳回的結果數目上限為 list-packages 100。您可以使用 --max-results選項變更此結果限制。

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

的允許值上限為 --max-results 1，000。若要允許列出具有超過 1，000 個套件的儲存庫中的套件， list-packages支援使用回應中的 nextToken 欄位進行分頁。如果儲存庫中的套件數量超過的值--max-results，您可以將 的值傳遞nextToken給另一個叫用 list-packages，以取得下一頁的結果。

```
aws codeartifact list-packages --domain my_domain --domain-owner 111122223333 --
repository my_repo \
  --next-token rO0ABXNyAEdjb...
```

列出套件版本

使用 in AWS CodeArtifact list-package-versions命令來取得儲存庫中套件名稱的所有版本清單。

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

輸出範例：

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

您可以將 `--status` 參數新增至`list-package-versions`呼叫，以根據套件版本狀態篩選結果。如需套件版本狀態的詳細資訊，請參閱 套件版本狀態。

您可以使用 `list-package-versions`--max-results和 `--next-token` 參數，從 分頁回應。針對 `--max-results`，指定從 1 到 1000 的整數，以指定單一頁面中傳回的結果數目。其預設值為 50。若要傳回後續頁面，請`list-package-versions`再次執行 ，並將先前命令輸出中收到`nextToken`的值傳遞給 `--next-token`。不使用 `--next-token`選項時，一律會傳回結果的第一頁。

`list-package-versions` 命令不會列出上游儲存庫中的套件版本。不過，會列出上游儲存庫中套件版本在套件版本請求期間複製到儲存庫的參考。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。

列出 npm 套件版本

若要列出 npm 套件的所有套件版本，請將 `--format`選項的值設定為 npm。

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

若要列出特定命名空間 (npm 範圍) 中的 npm 套件版本，請使用 --namespace選項。--namespace 選項的值不應包含前置 @。若要搜尋命名空間 @types，請將 值設定為 ##。

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

列出 Maven 套件版本

若要列出 Maven 套件的所有套件版本，請將 --format選項的值設定為 maven。您也必須在 --namespace選項中指定 Maven 群組 ID。

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

排序版本

list-package-versions 可以輸出根據發佈時間以遞減順序排序的版本 （ 最新發佈的版本會先列出 ）。使用 值為 的 --sort-by 參數PUBLISHED_TIME，如下所示。

```
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
```

輸出範例：

```
{

  "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...."
}
```

預設顯示版本

的傳回值`defaultDisplayVersion`取決於套件格式：

- 對於一般、Maven 和 PyPI 套件，這是最近發佈的套件版本。
- 對於 npm 套件，它是`latest`標籤參考的版本。如果未設定`latest`標籤，則它是最近發佈的套件版本。

格式輸出

您可以使用所有 AWS CLI 命令可用的參數，讓`list-package-versions`回應精簡且更易讀。使用 `--query` 參數來指定每個傳回套件版本的格式。使用 `--output` 參數將回應格式化為純文字。

```
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]'
```

輸出範例：

```
0.1.1
0.1.2
0.1.0
3.0.0
```

如需詳細資訊，請參閱AWS Command Line Interface 《 使用者指南》中的從 控制命令輸出 AWS CLI。

列出套件版本資產

資產是儲存在 CodeArtifact 中的個別檔案 （ 例如，npm `.tgz` 檔案或 Maven POM 或 JAR 檔案 ），與套件版本相關聯。您可以使用 `list-package-version-assets`命令列出每個套件版本中的資產。

執行 `list-package-version-assets`命令，以傳回您 AWS 帳戶和目前 AWS 區域中每個資產的下列相關資訊：

- 其名稱。
- 其大小，以位元組為單位。
- 一組用於檢查總和驗證的雜湊值。

例如，使用以下命令列出 Python 套件 `flatten-json`版本 的資產`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
```

以下將顯示輸出。

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

列出 npm 套件的資產

npm 套件一律具有名為 的單一資產`package.tgz`。若要列出範圍 npm 套件的資產，請在 `--namespace`選項中包含範圍。

```
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
```

列出 Maven 套件的資產

若要列出 Maven 套件的資產，請在 `--namespace`選項中包含套件命名空間。若要列出 Maven 套件的資產`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
```

下載套件版本資產

資產是儲存在 CodeArtifact 中的個別檔案（例如，npm .tgz 檔案或 Maven POM 或 JAR 檔案），與套件版本相關聯。您可以使用 下載套件資產get-package-version-assets command。這可讓您擷取資產，而無需使用套件管理員用戶端，例如 npm或 pip。若要下載資產，您必須提供可使用 list-package-version-assets命令取得的資產名稱，如需詳細資訊，請參閱 列出套件版本資產。資產將以您指定的檔案名稱下載至本機儲存體。

下列範例會從版本為 *27.1-jre* 的 Maven 套件 *com.google.guava#guava* 下載 guava-*27.1-jre*.jar 資產。

```
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
```

在此範例中，檔案名稱由上述命令中的最後一個引數指定為 *guava-27.1-jre.jar*，因此下載的資產將命名為 *guava-27.1-jre.jar#*

命令的輸出將是：

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

> **Note**
>
> 若要從範圍 npm 套件下載資產，請在 --namespace選項中包含範圍。使用 時，必須省略 @符號--namespace。例如，如果範圍為 @types，請使用 --namespace types。

使用 下載資產`get-package-version-asset`需要套件資源的`codeartifact:GetPackageVersionAsset`許可。如需資源型許可政策的詳細資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的資源型政策。

在儲存庫之間複製套件

您可以在 CodeArtifact 中將套件版本從一個儲存庫複製到另一個儲存庫。這對於套件提升工作流程或團隊或專案之間共用套件版本等案例很有幫助。來源和目的地儲存庫必須位於相同的網域中，才能複製套件版本。

複製套件所需的 IAM 許可

若要在 CodeArtifact 中複製套件版本，呼叫使用者必須具有必要的 IAM 許可，且連接至來源和目的地儲存庫的資源型政策必須具有必要的許可。如需資源型許可政策和 CodeArtifact 儲存庫的詳細資訊，請參閱 儲存庫政策。

呼叫 的使用者`copy-package-versions`必須擁有來源儲存庫的 `ReadFromRepository` 許可，以及目的地儲存庫的 `CopyPackageVersions`許可。

來源儲存庫必須具有 `ReadFromRepository`許可，而目的地儲存庫必須擁有指派給 IAM 帳戶或使用者複製套件的 `CopyPackageVersions`許可。下列政策是使用 `put-repository-permissions-policy`命令新增至來源儲存庫或目的地儲存庫的範例儲存庫政策。將 *111122223333* 取代為呼叫 的帳戶 ID`copy-package-versions`。

> **Note**
>
> 如果存在，呼叫 `put-repository-permissions-policy`將取代目前的儲存庫政策。您可以使用 `get-repository-permissions-policy`命令來查看政策是否存在，如需詳細資訊，請參閱 讀取政策。如果政策確實存在，您可能想要將這些許可新增至其中，而不是取代它。

範例來源儲存庫許可政策

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

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

範例目的地儲存庫許可政策

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

複製套件版本

使用 CodeArtifact 中的 `copy-package-versions`命令，將一或多個套件版本從來源儲存庫複製到相同網域中的目的地儲存庫。下列範例會將名為 的 npm 套件版本 6.0.2 和 4.0.0 `my-package` 從`my_repo`儲存庫複製到`repo-2`儲存庫。

```
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
```

您可以在單一操作中複製相同套件名稱的多個版本。若要複製不同套件名稱的版本，您必須`copy-package-versions`針對每個套件名稱呼叫 。

先前的命令會產生下列輸出，假設兩個版本都可以成功複製。

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

從上游儲存庫複製套件

一般而言， `copy-package-versions`只會在 `--source-repository`選項指定的儲存庫中尋找要複製的版本。不過，您可以使用 `--include-from-upstream`選項，從來源儲存庫及其上游儲存庫複製版本。如果您使用 CodeArtifact SDK，請呼叫 `includeFromUpstream` 參數設為 true 的 `CopyPackageVersions` API。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。

複製範圍 npm 套件

若要複製範圍內的 npm 套件版本，請使用 `--namespace`選項來指定範圍。例如，若要複製套件 `@types/react`，請使用 `--namespace types`。使用 時，必須省略 `@`符號`--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
```

複製 Maven 套件版本

若要在儲存庫之間複製 Maven 套件版本，請使用 `--namespace`選項傳遞 Maven 群組 ID，並使用 選項傳遞 Maven artifactID，以指定要複製的套件`--name`。例如，若要複製單一版本的 `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
```

如果成功複製套件版本，輸出將類似於以下內容。

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

來源儲存庫中不存在的版本

如果您指定的版本不存在於來源儲存庫中，複製將會失敗。如果來源儲存庫中存在某些版本，但有些版本不存在，則所有版本都將無法複製。在下列範例中，來源儲存庫中存在 npm `array-unique` 套件的 0.2.0 版，但 5.6.7 版不是：

```
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
```

此案例中的輸出將類似於以下內容。

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

SKIPPED 錯誤碼用於表示版本未複製到目的地儲存庫，因為無法複製另一個版本。

已存在於目的地儲存庫的版本

當套件版本複製到已存在的儲存庫時，CodeArtifact 會比較兩個儲存庫中的套件資產和套件版本層級中繼資料。

如果來源和目的地儲存庫中的套件版本資產和中繼資料相同，則不會執行複本，但操作會被視為成功。這表示 copy-package-versions 是等冪的。發生這種情況時，來源和目的地儲存庫中已存在的版本將不會列在 的輸出中copy-package-versions。

在下列範例中，來源儲存庫 中array-unique存在兩個版本的 npm 套件repo-1。版本 0.2.1 也存在於目的地儲存庫中，dest-repo而版本 0.2.0 則不存在。

```
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
```

此案例中的輸出將類似於以下內容。

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

0.2.0 版已列在 中，successfulVersions因為它已成功從來源複製到目的地儲存庫。輸出中不會顯示 0.2.1 版，因為它已存在於目的地儲存庫中。

如果來源和目的地儲存庫中的套件版本資產或中繼資料不同，複製操作將會失敗。您可以使用 --allow-overwrite 參數來強制覆寫。

如果某些版本存在於目的地儲存庫中，而有些版本不存在，則所有版本都將無法複製。在下列範例中，來源和目的地儲存庫中都存在 array-unique npm 套件的 0.3.2 版，但套件版本的內容不同。版本 0.2.1 存在於來源儲存庫中，但目的地不存在。

```
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
```

此案例中的輸出將類似於以下內容。

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

版本 0.2.1 會標示為 ，SKIPPED因為它未複製到目的地儲存庫。因為 0.3.2 版的複本已存在於目的地儲存庫中，但在來源和目的地儲存庫中不同，所以未複製。

指定套件版本修訂

套件版本修訂是一種字串，可指定套件版本的特定資產和中繼資料集。您可以指定套件版本修訂，以複製處於特定狀態的套件版本。若要指定套件版本修訂，請使用 --version-revisions 參數將一或多個逗號分隔的套件版本和套件版本修訂對傳遞至 copy-package-versions命令。

> **Note**
>
> 您必須使用 指定 --versions或 --version-revisions 參數copy-package-versions。您不能同時指定兩者。

下列範例只會在套件版本修訂版 的來源儲存庫中存在 0.3.2 版的套件my-package時複製REVISION-1-SAMPLE-6C81EFF7DA55CC套件。

```
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
```

下列範例會複製套件 0my-package.3.2 和 0.3.13 的兩個版本。只有當 的來源儲存庫 0.3.2 版具有my-package修訂版 REVISION-1-SAMPLE-6C81EFF7DA55CC且 0.3.13 版具有修訂版 時，複製才會成功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
```

若要尋找套件版本的修訂，請使用 describe-package-version或 list-package-versions命令。

如需詳細資訊，請參閱 CodeArtifact API 參考中的 套件版本修訂和 CopyPackageVersion。

複製 npm 套件

如需使用 npm 套件copy-package-versions之行為的詳細資訊，請參閱 npm 標籤和 CopyPackageVersions API。

刪除套件或套件版本

您可以使用 delete-package-versions命令一次刪除一或多個套件版本。若要從儲存庫完全移除套件，包括所有相關版本和組態，請使用 delete-package命令。套件可以存在於儲存庫中，而沒有任何套件版本。當使用 delete-package-versions命令刪除所有版本，或使用 put-package-origin-configuration API 操作在沒有任何版本的情況下建立套件時，就會發生這種情況 （ 請參閱 編輯套件原始伺服器控制項)。

主題

- 刪除套件 (AWS CLI)
- 刪除套件 (主控台)
- 刪除套件版本 (AWS CLI)
- 刪除套件版本 (主控台)
- 刪除 npm 套件或套件版本

- 刪除 Maven 套件或套件版本
- 刪除套件或套件版本的最佳實務

刪除套件 (AWS CLI)

您可以使用 delete-package命令刪除套件，包括其所有套件版本和組態。下列範例會刪除my_domain網域中儲存庫my-package中名為 my_repo的 PyPI 套件：

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

輸出範例：

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

您可以針對相同的套件名稱執行 describe-package ，以確認套件已刪除：

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

刪除套件 (主控台)

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇 Repositories (儲存庫)。

3. 選擇您要從中刪除套件的儲存庫。
4. 選擇您要刪除的套件。
5. 選擇刪除套件。

刪除套件版本 (AWS CLI)

您可以使用 delete-package-versions命令一次刪除一或多個套件版本。下列範例會刪除my_domain網域my_repo中 my-package 中名為 5.0.0的 PyPI 4.0.0套件版本 4.0.1、 和 ：

```
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
```

輸出範例：

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

您可以針對list-package-versions相同的套件名稱執行 ，以確認已刪除版本：

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

刪除套件版本 (主控台)

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇 Repositories (儲存庫)。
3. 選擇您要從中刪除套件版本的儲存庫。
4. 選擇您要從中刪除版本的套件。
5. 選取您要刪除的套件版本。
6. 選擇 刪除。

> **Note**
>
> 在 主控台中，您一次只能刪除一個套件版本。若要一次刪除多個 ，請使用 CLI。

刪除 npm 套件或套件版本

若要刪除 npm 套件或個別套件版本，請將 `--format`選項設定為 npm。若要刪除範圍 npm 套件中的套件版本，請使用 `--namespace`選項來指定範圍。例如，若要刪除套件 `@types/react`，請使用 `--namespace types`。使用 時省略 `@`符號`--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
```

若要刪除套件 `@types/react`，包括其所有版本：

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

刪除 Maven 套件或套件版本

若要刪除 Maven 套件或個別套件版本，請將 `--format`選項設定為 ，`maven`並使用 `--namespace`選項傳遞 Maven 群組 ID，並使用 選項傳遞 Maven artifactID，以指定要刪除的套件`--name`。例如，以下說明如何刪除單一版本的 `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
```

下列範例示範如何刪除套件 com.google.guava:guava，包括其所有版本：

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

刪除套件或套件版本的最佳實務

如果您確實需要刪除套件版本，根據最佳實務，建議您建立儲存庫來存放您要刪除之套件版本的備份副本。您可以先呼叫copy-package-versions備份儲存庫來執行此操作：

```
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
```

複製套件版本之後，您就可以在要刪除的套件或套件版本delete-package-versions上呼叫 。

```
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
```

檢視和更新套件版本詳細資訊和相依性

您可以在 CodeArtifact 中檢視套件版本的相關資訊，包括相依性。您也可以更新套件版本的狀態。如需套件版本狀態的詳細資訊，請參閱 套件版本狀態。

檢視套件版本詳細資訊

使用 describe-package-version命令來檢視套件版本的詳細資訊。套件版本詳細資訊會在發佈至 CodeArtifact 時從套件中擷取。不同套件中的詳細資訊會有所不同，取決於其格式，以及作者新增至其中的資訊量。

describe-package-version 命令輸出中的大多數資訊取決於套件格式。例如，會從其package.json檔案describe-package-version擷取 npm 套件的資訊。此修訂由 CodeArtifact 建立。如需詳細資訊，請參閱指定套件版本修訂。

如果兩個具有相同名稱的套件版本都位於不同的命名空間中，則可以位於相同的儲存庫中。使用選用--namespace參數來指定命名空間。如需詳細資訊，請參閱 檢視 npm 套件版本詳細資訊 或 檢視 Maven 套件版本詳細資訊 。

下列範例會傳回儲存my_repo庫中名為 1.9.0之 Python pyhamcrest 套件版本的詳細資訊。

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

輸出可能如下所示。

```
{
  "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 會從套件作者提供的中繼資料中擷取套件版本詳細資訊，例如套件首頁或套件授權資訊。如果此資訊中的任何一個超過 400 KB，即 DynamoDB 項目大小限制，CodeArtifact 將無法處理此類資料，而且您可能無法在主控台或 的回應中看到此資訊describe-package-version。例如，例如 https://pypi.org/project/rapyd-sdk/ 的 python 套件具有非常大的授權欄位，因此 CodeArtifact 不會處理此資訊。

檢視 npm 套件版本詳細資訊

若要檢視 npm 套件版本的詳細資訊，請將 `--format`選項的值設定為 **npm**。或者，在 `--namespace`選項中包含套件版本命名空間 (npm 範圍)。`--namespace` 選項的值不應包含前置 `@`。若要搜尋命名空間 `@types`，請將 值設定為 *##*。

以下內容會傳回 `@types`範圍webpack中名為 `4.41.5`的 npm 套件版本詳細資訊。

```
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
```

輸出可能如下所示。

```
{
  "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"
  }
}
```

檢視 Maven 套件版本詳細資訊

若要檢視 Maven 套件版本的詳細資訊，請將 `--format`選項的值設定為 `maven`，並在 `--namespace`選項中包含套件版本命名空間。

下列範例會傳回命名`org.apache.commons`空間和`my_repo`儲存庫`commons-rng-client-api`中名為 `1.2`之 Maven 套件版本的詳細資訊。

```
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
```

輸出可能如下所示。

```
{
  "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 不會從父 POM 檔案擷取套件版本詳細資訊。指定套件版本的中繼資料只會包含該確切套件版本的 POM 中的資訊，而不是父 POM 或使用 POM `parent`標籤暫時參考的任何其他 POM。這表示 的輸出`describe-package-version`會省略依賴`parent`參考包含此中繼資料的 Maven 套件版本的中繼資料 （ 例如授權資訊 ）。

檢視套件版本相依性

使用 `list-package-version-dependencies`命令來取得套件版本的相依性清單。下列命令列出 `my_domain` 網域中儲存`my_repo`庫`4.41.5`中名為 `my-package`、版本 之 npm 套件的相依性。

```
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
```

輸出可能如下所示。

```
{
  "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"
}
```

如需 dependencyType 欄位的支援值範圍，請參閱 CodeArtifact API 中的 PackageDependency 資料類型。

檢視套件版本 readme 檔案

有些套件格式，例如 npm，包含 README 檔案。使用 get-package-version-readme 取得套件版本的 README 檔案。下列命令會傳回 my_domain 網域中儲存my_repo庫4.41.5中名為 my-package、版本 的 npm 套件README檔案。

> **Note**
>
> CodeArtifact 不支援顯示一般或 Maven 套件中的讀我檔案。

```
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
```

輸出可能如下所示。

```
{
  "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"
}
```

更新套件版本狀態

CodeArtifact 中的每個套件版本都有一個狀態，描述套件版本的目前狀態和可用性。您可以使用 AWS CLI 和 主控台變更套件版本狀態。

> **Note**
>
> 如需套件版本狀態的詳細資訊，包括可用狀態的清單，請參閱 套件版本狀態。

更新套件版本狀態

設定套件版本的狀態可讓 控制套件版本的使用方式，而無需將其完全從儲存庫中刪除。例如，當套件版本的狀態為 時`Unlisted`，它仍然可以正常下載，但不會出現在傳回 等命令的套件版本清單中`npm view`。UpdatePackageVersionsStatus API 允許在單一 API 呼叫中設定相同套件的多個版本的套件版本狀態。如需不同狀態的說明，請參閱 套件概觀。

使用 update-package-versions-status命令將套件版本的狀態變更為 Published、Unlisted或 Archived。若要查看使用 命令所需的 IAM 許可，請參閱 更新套件版本狀態所需的 IAM 許可。下列範例會將 npm 套件 4.1.0 版的狀態chalk設定為 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
```

輸出範例：

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

此範例使用 npm 套件，但命令對其他格式的運作方式相同。您可以使用單一命令將多個版本移至相同的目標狀態，請參閱下列範例。

```
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
```

輸出範例：

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

```
}
```

請注意，一旦發佈，套件版本就無法移回 Unfinished 狀態，因此不允許此狀態做為 --target-status 參數的值。若要將套件版本移至 Disposed 狀態，請改用 dispose-package-versions命令，如下所述。

更新套件版本狀態所需的 IAM 許可

若要呼叫 update-package-versions-status套件，您必須擁有套件資源的 codeartifact:UpdatePackageVersionsStatus許可。這表示您可以授予每個套件呼叫 update-package-versions-status的許可。例如，授予在 npm 套件 *chalk* update-package-versions-status上呼叫 許可的 IAM 政策會包含如下所示的陳述式。

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

更新範圍 npm 套件的狀態

若要使用範圍更新 npm 套件版本的套件版本狀態，請使用 --namespace 參數。例如，若要取消列出的 8.0.0 版@nestjs/core，請使用下列命令。

```
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
```

更新 Maven 套件的狀態

Maven 套件一律具有群組 ID，在 CodeArtifact 中稱為命名空間。呼叫 時，使用 --namespace 參數指定 Maven 群組 IDupdate-package-versions-status。例如，若要封存 Maven 套件 的 2.13.1 版org.apache.logging.log4j:log4j，請使用下列命令。

```
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
```

指定套件版本修訂

套件版本修訂是一種字串，可指定套件版本的特定資產和中繼資料集。您可以指定套件版本修訂，以更新處於特定狀態的套件版本狀態。若要指定套件版本修訂，請使用 --version-revisions 參數傳遞一或多個逗號分隔的套件版本和套件版本修訂對。只有當套件版本的目前修訂版符合指定的值時，才會更新套件版本的狀態。

> **Note**
>
> 使用 –-versions 參數時，也必須定義 --version-revisions 參數。

```
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
```

若要使用單一命令更新多個版本，請將版本和版本修訂對的逗號分隔清單傳遞至--version-revisions選項。下列範例命令會定義兩個不同的套件版本和套件版本修訂對。

```
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
```

輸出範例：

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

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

更新多個套件版本時，傳遞至 的版本`--version-revisions`必須與傳遞至 的版本相同`--versions`。如果未正確指定修訂，該版本將不會更新其狀態。

使用預期的狀態參數

`update-package-versions-status` 命令提供 `--expected-status` 參數，支援指定套件版本的預期目前狀態。如果目前狀態不符合傳遞給 的值`--expected-status`，則該套件版本的狀態將不會更新。

例如，在 *my_repo* 中，npm 套件的 4.0.0 和 4.1.0 版`chalk`目前狀態為 Published。由於狀態不相符`update-package-versions-status`，指定預期狀態 的 呼叫`Unlisted`將無法更新兩個套件版本。

```
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
```

輸出範例：

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

個別套件版本的錯誤

呼叫 時，套件版本的狀態不會更新，原因有很多update-package-versions-status。例如，套件版本修訂可能未正確指定，或預期狀態與目前狀態不符。在這些情況下，版本會包含在 API 回應的failedVersions映射中。如果某個版本失敗，在對 的相同呼叫中指定的其他版本update-package-versions-status可能會略過，而不會更新其狀態。這類版本也會包含在具有 之errorCode的failedVersions地圖中SKIPPED。

在 的目前實作中update-package-versions-status，如果一或多個版本無法變更其狀態，則會略過所有其他版本。也就是說，所有版本都會成功更新，或不會更新任何版本。API 合約不保證此行為；未來某些版本可能會成功，而其他版本在對 的單一呼叫中失敗update-package-versions-status。

下列範例命令包含因套件版本修訂不相符而導致的版本狀態更新失敗。該更新失敗會導致略過另一個版本狀態更新呼叫。

```
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
```

輸出範例：

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

處置套件版本

Disposed 套件狀態具有與 類似的行為Archived，但 CodeArtifact 將永久刪除套件資產，因此不會再向網域擁有者的帳戶收取資產儲存的費用。如需每個套件版本狀態的詳細資訊，請參閱 套件版本狀態。若要將套件版本的狀態變更為 Disposed，請使用 dispose-package-versions命令。此功能與 不同，update-package-versions-status因為處置套件版本是不可逆的。由於套件資產將被刪除，因此版本的狀態無法變更回 Archived、 Unlisted或 Published。對於已處置的套件版本，唯一可以採取的動作是使用 delete-package-versions命令將其刪除。

若要dispose-package-versions成功呼叫 ，呼叫的 IAM 主體必須具有套件資源的 codeartifact:DisposePackageVersions 許可。

dispose-package-versions 命令的行為類似於 update-package-versions-status，包括版本修訂--version-revisions 和預期狀態區段中所述的 和 --expected-status選項的行為。例如，下列命令會嘗試處置套件版本，但由於預期狀態不相符而失敗。

```
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
```

輸出範例：

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

如果使用 --expected-status的 再次執行相同的命令Published，處置將會成功。

```
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
```

輸出範例：

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

編輯套件原始伺服器控制項

在 AWS CodeArtifact 中，套件版本可以直接發佈、從上游儲存庫下拉，或從外部公有儲存庫擷取，以新增至儲存庫。允許透過直接從公有儲存庫發佈和擷取來新增套件的套件版本，讓您容易遭受相依性替代攻擊。如需詳細資訊，請參閱相依性替代攻擊。為了保護自己免受相依性替代攻擊，您可以在儲存庫中的套件上設定套件原始伺服器控制，以限制將該套件的版本新增至儲存庫的方式。

任何想要允許不同套件新版本來自兩個內部來源的團隊，例如直接發佈和外部來源，例如公有儲存庫，都應考慮設定套件原始伺服器控制。根據預設，套件原始伺服器控制會根據套件的第一個版本新增至儲存庫的方式進行設定。如需套件原始伺服器控制設定及其預設值的相關資訊，請參閱 套件原始伺服器控制設定。

若要在使用 `put-package-origin-configuration` API 操作後移除套件記錄，請使用 `delete-package`（請參閱 刪除套件或套件版本)。

常見的套件存取控制案例

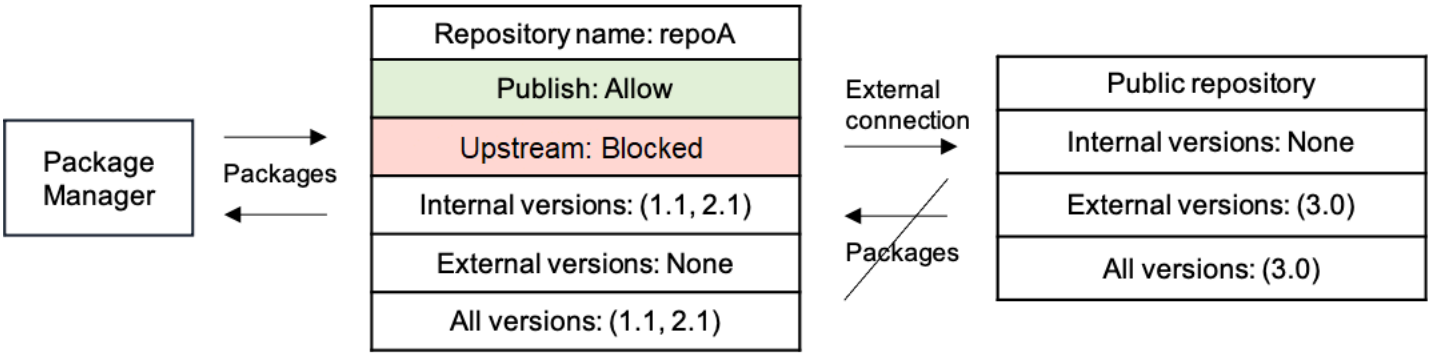
本節包含將套件版本新增至 CodeArtifact 儲存庫的一些常見案例。根據第一個套件版本的新增方式，將為新套件設定套件原始伺服器控制設定。

在下列案例中，內部套件是直接從套件管理員發佈到儲存庫的套件，例如您或您的團隊撰寫和維護的套件。外部套件是存在於公有儲存庫中的套件，可透過外部連線擷取到您的儲存庫。

針對現有的內部套件發佈外部套件版本

在此案例中，請考慮內部套件 packageA。您的團隊會將 packageA 的第一個套件版本發佈至 CodeArtifact 儲存庫。由於這是該套件的第一個套件版本，套件原始伺服器控制設定會自動設定為發佈：允許和上游：封鎖。套件存在於您的儲存庫之後，具有相同名稱的套件會發佈至連線至 CodeArtifact 儲存庫的公有儲存庫。這可能是對內部套件的嘗試相依性替代攻擊，也可能只是巧合。無論如何，套件原始伺服器控制都會設定為封鎖新外部版本的擷取，以保護自己免於潛在的攻擊。

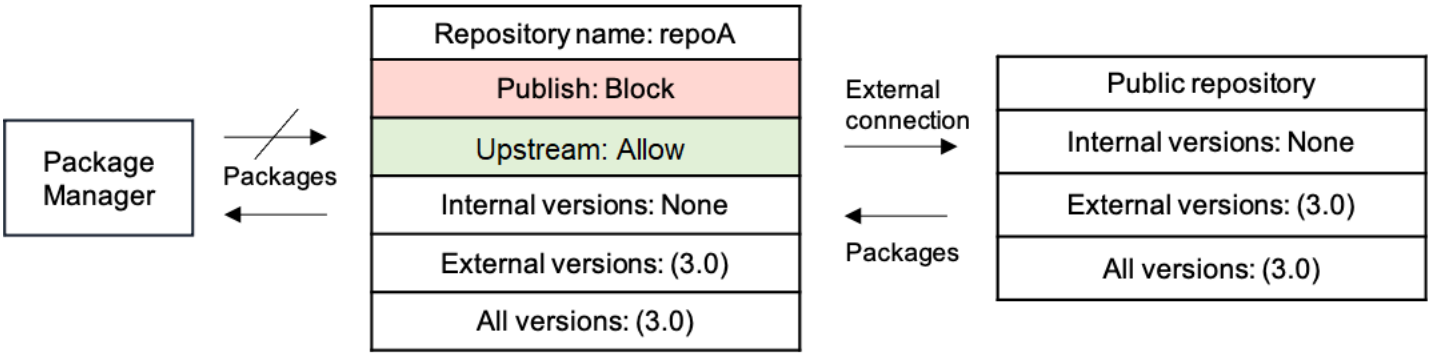
在下圖中，repoA 是具有公有儲存庫外部連線的 CodeArtifact 儲存庫。您的儲存庫包含 packageA 的 1.1 和 2.1 版，但 3.0 版會發佈至公有儲存庫。一般而言，在套件管理員請求套件之後，repoA 會擷取 3.0 版。由於套件擷取設定為 Block，因此 3.0 版不會擷取到您的 CodeArtifact 儲存庫，且無法供與其連線的套件管理員使用。



已發佈現有外部套件的內部套件版本

在此案例中，套件 packageB 在外部存在於您已連線至儲存庫的公有儲存庫中。當套件管理員連接到您的儲存庫請求 packageB 時，套件版本會從公有儲存庫擷取到您的儲存庫。由於這是新增至儲存庫的第一個 packageB 套件版本，因此套件原始伺服器設定會設定為 Publish： BLOCK and Upstream： ALLOW。稍後，您會嘗試將具有相同套件名稱的版本發佈至儲存庫。您可能不知道公有套件，並嘗試以相同名稱發佈不相關的套件，或者嘗試發佈修補版本，或者嘗試直接發佈已存在於外部的確切套件版本。CodeArtifact 會拒絕您嘗試發佈的版本，但可讓您明確覆寫拒絕，並視需要發佈版本。

在下圖中，repoA 是具有公有儲存庫外部連線的 CodeArtifact 儲存庫。您的儲存庫包含從公有儲存庫擷取的 3.0 版。您想要將 1.1 版發佈到您的儲存庫。一般而言，您可以將 1.2 版發佈至 repoA，但由於發佈設定為 Block，因此無法發佈 1.2 版。



發佈現有外部套件的修補套件版本

在此案例中，套件 packageB 在外部存在於您已連線至儲存庫的公有儲存庫中。當套件管理員連接到您的儲存庫請求 packageB 時，套件版本會從公有儲存庫擷取到您的儲存庫。由於這是新增至儲存庫的第一個 packageB 套件版本，因此套件原始伺服器設定會設定為 Publish： BLOCK and Upstream： ALLOW。您的團隊決定需要將此套件的修補套件版本發佈至儲存庫。為了能夠直接發佈套件版本，您的團隊會將套件原始伺服器控制設定變更為發佈：允許和上游：封鎖。此套件的版本現在可以直接發佈到您的儲存庫，並從公有儲存庫擷取。在您的團隊發佈修補的套件版本之後，您的團隊會將套件原始伺服器設定還原為 Publish： BLOCK and Upstream： ALLOW。

套件原始伺服器控制設定

使用套件原始伺服器控制，您可以設定如何將套件版本新增至儲存庫。下列清單包含可用的套件原始伺服器控制設定和值。

> **Note**
>
> 在套件群組上設定原始伺服器控制項時，可用的設定和值會有所不同。如需詳細資訊，請參閱套件群組原始伺服器控制。

發布

此設定會設定是否可使用套件管理員或類似工具，將套件版本直接發佈至儲存庫。

- 允許：可以直接發佈套件版本。
- 封鎖：無法直接發佈套件版本。

上游

此設定會設定套件版本是否可以從外部、公有儲存庫擷取，或在套件管理員要求時從上游儲存庫保留。

- 允許：任何套件版本都可以從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留，或從具有外部連線的公有來源擷取。
- 封鎖：套件版本無法從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留，或從具有外部連線的公有來源擷取。

預設套件原始伺服器控制設定

預設套件原始伺服器控制設定是根據套件相關聯的套件群組原始伺服器控制設定進行設定。如需套件群組和套件群組原始伺服器控制項的詳細資訊，請參閱 在 CodeArtifact 中使用套件群組和 套件群組原始伺服器控制。

如果套件與每個限制類型的限制設定ALLOW為 的套件群組相關聯，則套件的預設套件原始伺服器控制項將根據該套件的第一個版本如何新增至儲存庫。

- 如果套件管理員正確發佈第一個套件版本，則設定將是發佈：允許和上游：鎖定。
- 如果從公有來源擷取第一個套件版本，則設定將是發佈：封鎖和上游：允許。

> **Note**
>
> 在 2022 年 5 月前後，CodeArtifact 儲存庫中存在的套件將具有發佈：允許和上游：允許的預設套件原始伺服器控制。這類套件必須手動設定套件原始伺服器控制。目前預設值自那時起已在新套件上設定，並在 2022 年 7 月 14 日啟動此功能時開始強制執行。如需設定套件原始伺服器控制項的詳細資訊，請參閱 編輯套件原始伺服器控制項。

否則，如果套件與具有至少一個限制設定 `BLOCK`或 的套件群組相關聯`ALLOW_SPECIFIC_REPOSITORIES`，則該套件的預設原始伺服器控制設定將設定為 Publish：ALLOW 和 Upstream： ALLOW。

套件原始伺服器控制如何與套件群組原始伺服器控制互動

由於套件具有原始伺服器控制設定，且其相關聯的套件群組具有原始伺服器控制設定，因此請務必了解這兩個不同的設定如何彼此互動。

這兩個設定之間的互動是 的設定`BLOCK`一律會勝過 的設定`ALLOW`。下表列出一些範例組態及其有效的原始伺服器控制設定。

套件原始伺服器控制設定	套件群組原始伺服器控制設定	有效的原始伺服器控制設定
PUBLISH：允許 UPSTREAM：允許	PUBLISH：允許 UPSTREAM：允許	PUBLISH：允許 UPSTREAM：允許
PUBLISH：BLOCK UPSTREAM：允許	PUBLISH：允許 UPSTREAM：允許	PUBLISH：BLOCK UPSTREAM：允許
PUBLISH：允許 UPSTREAM：允許	PUBLISH：允許 UPSTREAM：封鎖	PUBLISH：允許 UPSTREAM：封鎖

這表示原始伺服器設定為 Publish： ALLOW 和 Upstream： ALLOW 的套件，會有效地延遲至相關聯的套件群組原始伺服器控制設定。

編輯套件原始伺服器控制項

套件原始伺服器控制項會根據套件的第一個套件版本如何新增至儲存庫來自動設定，如需詳細資訊，請參閱 預設套件原始伺服器控制設定。若要新增或編輯 CodeArtifact 儲存庫中套件的套件原始伺服器控制項，請執行下列程序中的步驟。

新增或編輯套件原始伺服器控制項 （主控台）

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇包含您要編輯之套件的儲存庫。
3. 在套件表格中，搜尋並選取您要編輯的套件。
4. 在套件摘要頁面的原始伺服器控制項中，選擇編輯。
5. 在編輯原始伺服器控制項中，選擇您要為此套件設定的套件原始伺服器控制項。套件原始控制設定 Publish 和 Upstream 必須同時設定。

   - 若要允許直接發佈套件版本，請在發佈中選擇允許。若要封鎖發佈套件版本，請選擇封鎖。
   - 若要允許從外部儲存庫擷取套件並從上游儲存庫提取套件，請在上游來源中選擇允許。若要封鎖所有從外部和上游儲存庫擷取和提取套件版本，請選擇封鎖。

新增或編輯套件原始伺服器控制項 (AWS CLI)

1. 如果您尚未設定 ， AWS CLI 請依照中的步驟進行設定使用 AWS CodeArtifact 設定。
2. 使用 `put-package-origin-configuration`命令來新增或編輯套件原始伺服器控制項。取代下列欄位：

   - 將 *my_domain* 取代為 CodeArtifact 網域，其中包含您要更新的套件。
   - 將 *my_repo* 取代為 CodeArtifact 儲存庫，其中包含您要更新的套件。
   - 將 *npm* 取代為您要更新的套件格式。
   - 將 *my_package* 取代為您要更新的套件名稱。
   - 將 *ALLOW* 和 *BLOCK* 取代為您想要的套件原始伺服器控制設定。

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

發佈和上游儲存庫

CodeArtifact 不允許發佈可連線上游儲存庫或公有儲存庫中存在的套件版本。例如，假設您想要將 Maven 套件發佈`com.mycompany.mypackage:1.0`至儲存庫 `myrepo`，且`myrepo`具有與 Maven Central 外部連線的上游儲存庫。請考慮下列案例。

1. 上的套件原始伺服器控制設定`com.mycompany.mypackage`是發佈：允許和上游：允許。如果上游儲存庫或 Maven Central 中存在 `com.mycompany.mypackage:1.0` ，CodeArtifact 會拒絕任何嘗試以 409 `myrepo` 衝突錯誤發佈至其中。您仍然可以發佈不同的版本，例如 `com.mycompany.mypackage:1.1`。
2. 上的套件原始伺服器控制設定`com.mycompany.mypackage`是 Publish： ALLOW 和 Upstream：BLOCK。您可以將任何 版本發佈`com.mycompany.mypackage`至尚未存在的儲存庫，因為套件版本無法連線。
3. 上的套件原始伺服器控制設定`com.mycompany.mypackage`是 Publish： BLOCK and Upstream：ALLOW。您無法將任何套件版本直接發佈到您的儲存庫。

在 CodeArtifact 中使用套件群組

套件群組可用來將組態套用至多個套件，這些套件使用套件格式、套件命名空間和套件名稱，以符合定義的模式。您可以使用套件群組，更方便地設定多個套件的套件原始伺服器控制。套件原始伺服器控制用於封鎖或允許擷取或發佈新的套件版本，以保護使用者免受稱為相依性替代攻擊的惡意動作。

CodeArtifact 中的每個網域會自動包含根套件群組。此根套件群組 `/*`包含所有套件，並允許套件版本預設從所有原始伺服器類型輸入網域中的儲存庫。根套件群組可以修改，但無法刪除。

套件群組組態功能在建立新套件群組或刪除現有套件群組時，會以最終一致的方式運作。這表示在建立或刪除套件群組時，原始伺服器控制項將套用至預期的關聯套件，但由於最終一致行為，會有些許延遲。達到最終一致性的時間取決於網域中的套件群組數量，以及網域中的套件數量。建立或刪除套件群組後，在一段短暫的時間內，原始伺服器控制項不會立即反映在相關聯的套件上。

此外，套件群組原始伺服器控制的更新幾乎立即生效。與建立或刪除套件群組不同，現有套件群組的原始伺服器控制變更會反映在相關聯的套件上，而不會有相同的延遲。

這些主題包含 AWS CodeArtifact 中套件群組的相關資訊。

主題

- 建立套件群組
- 檢視或編輯套件群組
- 刪除套件群組
- 套件群組原始伺服器控制
- 套件群組定義語法和相符行為
- 在 CodeArtifact 中標記套件群組

建立套件群組

您可以使用 CodeArtifact 主控台、 AWS Command Line Interface (AWS CLI) 或 建立套件群組 AWS CloudFormation。如需使用 CloudFormation 管理 CodeArtifact 套件群組的詳細資訊，請參閱 使用 建立 CodeArtifact 資源 AWS CloudFormation。

建立套件群組 （主控台）

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。

2. 在導覽窗格中，選擇網域，然後選擇您要在其中建立套件群組的網域。
3. 選擇套件群組，然後選擇建立套件群組。
4. 在套件群組定義中，輸入套件群組的套件群組定義。套件群組定義會決定哪些套件與群組相關聯。您可以使用文字手動輸入套件群組定義，也可以使用視覺化模式進行選取，系統會自動建立套件群組定義。
5. 若要使用視覺化模式建立套件群組定義：

   a. 選擇視覺化以切換到視覺化模式。

   b. 在套件格式中，選擇要與此群組建立關聯的套件格式。

   c. 在命名空間（範圍）中，選擇要比對的命名空間條件。

      - 等於：完全符合指定的命名空間。如果選擇，請輸入要比對的命名空間。
      - 空白：比對沒有命名空間的套件。
      - 以字詞開頭：比對以指定字詞開頭的命名空間。如果選擇，請輸入要比對的字首。如需單字和單字邊界的詳細資訊，請參閱 字詞、字詞邊界和字首比對。
      - 全部：比對所有命名空間中的套件。

   d. 如果選取等於、空白或開頭字詞，請在套件名稱中選擇要比對的套件名稱條件。

      - 完全等於：完全符合指定的套件名稱。如果選擇，請輸入要比對的套件名稱。
      - 字首開頭：比對以指定字首開頭的套件。
      - 開頭為單字：比對開頭為指定單字的套件。如果選擇，請輸入要比對的字首。如需單字和單字邊界的詳細資訊，請參閱 字詞、字詞邊界和字首比對。
      - 全部：符合所有套件。

   e. 選擇下一步以檢閱定義。
6. 若要使用文字輸入套件群組定義：

   a. 選擇文字以切換至文字模式。

   b. 在套件群組定義中，輸入套件群組定義。如需套件群組定義語法的詳細資訊，請參閱 套件群組定義語法和相符行為。

   c. 選擇下一步以檢閱定義。
7. 在檢閱定義中，根據先前提供的定義檢閱將包含在新套件群組中的套件。檢閱後，選擇下一步。
8. 在套件群組資訊中，選擇性地新增套件群組的描述和聯絡人電子郵件。選擇 Next (下一步)。
9. 在套件原始伺服器控制項中，設定原始伺服器控制項以套用至群組中的套件。如需套件群組原始伺服器控制項的詳細資訊，請參閱 套件群組原始伺服器控制。

10. 選擇建立套件群組。

建立套件群組 (AWS CLI)

使用 `create-package-group`命令在您的網域中建立套件群組。針對 `--package-group`選項，輸入套件群組定義，以決定哪些套件與群組相關聯。如需套件群組定義語法的詳細資訊，請參閱 套件群組定義語法和相符行為。

如果您尚未設定 ， AWS CLI 請依照中的步驟進行設定使用 AWS CodeArtifact 設定。

```
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
```

檢視或編輯套件群組

您可以使用 CodeArtifact 主控台或 AWS Command Line Interface () 來檢視所有套件群組的清單、檢視特定套件群組的詳細資訊，或編輯套件群組的詳細資訊或組態AWS CLI。

檢視或編輯套件群組 （ 主控台 ）

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇包含您要檢視或編輯之套件群組的網域。
3. 選擇套件群組，然後選擇您要檢視或編輯的套件群組。
4. 在詳細資訊中，檢視套件群組的相關資訊，包括其父群組、描述、ARN、聯絡人電子郵件和套件原始伺服器控制項。
5. 在子群組中，檢視將此群組做為父群組的套件群組清單。此清單中的套件群組可以從此套件群組繼承設定。如需詳細資訊，請參閱套件群組階層和模式特異性。
6. 在套件中，根據套件群組定義檢視屬於此套件群組的套件。在強度欄中，您可以看到套件關聯的強度。如需詳細資訊，請參閱套件群組階層和模式特異性。
7. 若要編輯套件群組資訊，請選擇編輯套件群組。

a. 在資訊中，更新套件群組的描述或聯絡資訊。您無法編輯套件群組的定義。

b. 在套件群組原始伺服器控制中，更新套件群組的原始伺服器控制設定，以決定相關聯的套件如何在網域中輸入儲存庫。如需詳細資訊，請參閱套件群組原始伺服器控制。

檢視或編輯套件群組 (AWS CLI)

使用下列命令，透過 檢視或編輯套件群組 AWS CLI。如果您尚未設定 ， AWS CLI 請依照中的步驟進行設定使用 AWS CodeArtifact 設定。

若要檢視網域中的所有套件群組，請使用 `list-package-groups`命令。

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

若要檢視套件群組的詳細資訊，請使用 `describe-package-group`命令。如需套件群組定義的詳細資訊，請參閱 套件群組定義語法和範例。

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

若要檢視套件群組的子套件群組，請使用 `list-sub-package-groups`命令。

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

若要檢視與套件相關聯的套件群組，請使用 `get-associated-package-group`命令。您必須使用 NuGet、Python 和 Swift 套件格式的標準化套件名稱和命名空間。如需套件名稱和命名空間如何標準化的詳細資訊，請參閱 NuGet、Python 和 Swift 名稱標準化文件。

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

```
        --namespace scope
```

若要編輯套件群組，請使用 update-package-group命令。此命令用於更新套件群組的聯絡資訊或描述。如需套件群組原始伺服器控制設定，以及新增或編輯這些設定的詳細資訊，請參閱 套件群組原始伺服器控制。如需套件群組定義的詳細資訊，請參閱 套件群組定義語法和範例

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

刪除套件群組

您可以使用 CodeArtifact 主控台或 AWS Command Line Interface () 刪除套件群組AWS CLI。

刪除套件群組時，請注意下列行為：

- /*無法刪除根套件群組 。
- 不會刪除與該套件群組相關聯的套件和套件版本。
- 刪除套件群組時，直接子套件群組將成為套件群組直接父系套件群組的子系。因此，如果任何子群組繼承來自父系的任何設定，這些設定可能會變更。

刪除套件群組 （ 主控台 ）

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇包含您要檢視或編輯之套件群組的網域。
3. 選擇套件群組。
4. 選擇您要刪除的套件群組，然後選擇刪除。
5. 在欄位中輸入 Delete，然後選擇 Delete。

刪除套件群組 (AWS CLI)

若要刪除套件群組，請使用 delete-package-group命令。

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

套件群組原始伺服器控制

套件原始伺服器控制用於設定套件版本如何進入網域。您可以在套件群組上設定原始伺服器控制，以設定與套件群組相關聯的每個套件版本如何在網域中輸入指定的儲存庫。

套件群組原始伺服器控制設定包含下列項目：

- 限制設定：這些設定定義套件是否可以從發佈、內部上游或外部公有儲存庫在 CodeArtifact 中輸入儲存庫。
- 允許儲存庫清單：每個限制設定都可以設定為允許特定儲存庫。如果將限制設定設定為允許特定儲存庫，則該限制將具有對應的允許儲存庫清單。

> **Note**
>
> 套件群組的原始伺服器控制設定與個別套件的原始伺服器控制設定略有不同。如需套件原始伺服器控制設定的詳細資訊，請參閱 套件原始伺服器控制設定。

限制設定

套件群組原始伺服器控制設定的限制設定會決定與該群組相關聯的套件如何在網域中輸入儲存庫。

發佈

PUBLISH 設定會設定是否可使用套件管理員或類似工具，將套件版本直接發佈至網域中的任何儲存庫。

- 允許：套件版本可以直接發佈到所有儲存庫。
- 封鎖：套件版本無法直接發佈到任何儲存庫。
- ALLOW_SPECIFIC_REPOSITORIES：套件版本只能直接發佈到允許發佈的儲存庫清單中指定的儲存庫。
- INHERIT：PUBLISH設定繼承自第一個父系套件群組，其設定不是 INHERIT。

EXTERNAL_UPSTREAM

EXTERNAL_UPSTREAM 設定會設定套件管理員要求時，是否可以從外部公有儲存庫擷取套件版本。如需支援的外部儲存庫清單，請參閱 支援的外部連線儲存庫。

- 允許：任何套件版本都可以從具有外部連線的公有來源擷取到所有儲存庫。
- 封鎖：套件版本無法從具有外部連線的公有來源擷取到任何儲存庫。
- ALLOW_SPECIFIC_REPOSITORIES：套件版本只能從公有來源擷取到外部上游的允許儲存庫清單中指定的儲存庫。
- INHERIT：EXTERNAL_UPSTREAM設定繼承自第一個父系套件群組，其設定不是 INHERIT。

INTERNAL_UPSTREAM

INTERNAL_UPSTREAM 設定會設定套件管理員請求時，是否可從相同 CodeArtifact 網域的內部上游儲存庫保留套件版本。

- 允許：任何套件版本都可以從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留。
- 封鎖：套件版本無法從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留。
- ALLOW_SPECIFIC_REPOSITORIES：套件版本只能從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留到內部上游的允許儲存庫清單中指定的儲存庫。
- INHERIT：INTERNAL_UPSTREAM設定繼承自第一個父系套件群組，其設定不是 INHERIT。

允許儲存庫清單

當限制設定設定為 時ALLOW_SPECIFIC_REPOSITORIES，套件群組會包含隨附的允許儲存庫清單，其中包含該限制設定允許的儲存庫清單。因此，套件群組包含 0 到 3 個允許儲存庫清單的任何位置，每個設定為 的設定各一個ALLOW_SPECIFIC_REPOSITORIES。

當您將儲存庫新增至套件群組的允許儲存庫清單時，您必須指定要將其新增至哪個允許的儲存庫清單。

可能允許的儲存庫清單如下：

- EXTERNAL_UPSTREAM：允許或封鎖從新增儲存庫中的外部儲存庫擷取套件版本。
- INTERNAL_UPSTREAM：允許或封鎖從已新增儲存庫中的另一個 CodeArtifact 儲存庫提取套件版本。
- PUBLISH：允許或封鎖從套件管理員直接發佈套件版本到新增的儲存庫。

編輯套件群組原始伺服器控制設定

若要新增或編輯套件群組的原始伺服器控制項，請執行下列程序中的步驟。如需套件群組原始伺服器控制設定的相關資訊，請參閱 限制設定和 允許儲存庫清單。

新增或編輯套件群組原始伺服器控制 (CLI)

1. 如果您尚未設定 ， AWS CLI 請依照中的步驟進行設定使用 AWS CodeArtifact 設定。
2. 使用 `update-package-group-origin-configuration`命令來新增或編輯套件原始伺服器控制項。

   - 針對 `--domain`，輸入 CodeArtifact 網域，其中包含您要更新的套件群組。
   - 針對 `--domain-owner`，輸入網域擁有者的帳戶號碼。
   - 針對 `--package-group`，輸入您要更新的套件群組。
   - 針對 `--restrictions`，輸入代表原始伺服器控制限制的鍵值對。
   - 針對 `--add-allowed-repositories`，輸入包含限制類型和儲存庫名稱的 JSON 物件，以新增至該限制的對應允許儲存庫清單。
   - 針對 `--remove-allowed-repositories`，輸入包含限制類型和儲存庫名稱的 JSON 物件，以從該限制的對應允許儲存庫清單中移除。

```
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
```

下列範例會在一個命令中新增多個限制和多個儲存庫。

```
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
```

套件群組原始伺服器控制組態範例

下列範例顯示常見套件管理案例的套件原始控制組態。

允許發佈具有私有名稱的套件，但不擷取

此案例可能是套件管理中的常見案例：

- 允許從套件管理員將具有私有名稱的套件發佈至您網域中的儲存庫，並封鎖從外部公有儲存庫擷取至您網域中的儲存庫。
- 允許從外部、公有儲存庫將所有其他套件擷取至您網域中的儲存庫，並封鎖從套件管理員發佈至您網域中的儲存庫。

若要達成此目的，您應該使用包含私有名稱 (PUBLISH) 和原始伺服器設定的模式 (PUBLISH：ALLOW、EXTERNAL_UPSTREAM： BLOCK 和 INTERNAL_UPSTREAM： ALLOW) 來設定套件群組。這將確保具有私有名稱的套件可以直接發佈，但無法從外部儲存庫擷取。

下列 AWS CLI 命令會使用符合所需行為的原始伺服器限制設定來建立和設定套件群組：

若要建立套件群組：

```
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"
```

若要更新套件群組的原始伺服器組態：

```
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
```

允許透過一個儲存庫從外部儲存庫擷取

在此案例中，您的網域有多個儲存庫。在這些儲存庫中， repoA 具有與 的上游連線repoB，而 具有與公有儲存庫 的外部連線npmjs.com，如下所示：

repoA --> repoB --> npmjs.com

您想要允許從特定套件群組擷取套件，/npm/space/anycompany~從 npmjs.com到 repoA，但只能透過 repoB。您也想要封鎖將與套件群組相關聯的套件擷取到網域中的任何其他儲存庫，並封鎖使用套件管理員直接發佈套件。若要達成此目的，您可以建立和設定套件群組，如下所示：

PUBLISH：BLOCK 和 EXTERNAL_UPSTREAM：ALOW_SPECIFIC_REPOSITORIES 和 INTERNAL_UPSTREAM： ALLOW_SPECIFIC_REPOSITORIES 的原始伺服器限制設定。

repoA 和 repoB已新增至適當的允許儲存庫清單：

- repoA 應新增至INTERNAL_UPSTREAM清單，因為它會從其內部上游 取得套件repoB。
- repoB 應新增至EXTERNAL_UPSTREAM清單，因為它會從外部儲存庫 取得套件npmjs.com。

下列 AWS CLI 命令會使用符合所需行為的原始伺服器限制設定來建立和設定套件群組：

若要建立套件群組：

```
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"
```

若要更新套件群組的原始伺服器組態：

```
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
```

套件群組原始伺服器控制設定如何與套件原始伺服器控制設定互動

由於套件具有原始伺服器控制設定，且其相關聯的套件群組具有原始伺服器控制設定，因此請務必了解這兩個不同設定如何彼此互動。如需設定之間互動的相關資訊，請參閱 套件原始伺服器控制如何與套件群組原始伺服器控制互動。

套件群組定義語法和相符行為

本主題包含定義套件群組、模式比對行為、套件關聯強度和套件群組階層的相關資訊。

內容

- 套件群組定義語法和範例
  - 套件群組定義和標準化
  - 套件群組定義中的命名空間
- 套件群組階層和模式特異性
- 字詞、字詞邊界和字首比對
- 區分大小寫
- 強配對和弱配對
- 其他變化

套件群組定義語法和範例

用於定義套件群組的模式語法會密切遵循套件路徑的格式。套件路徑是從套件的座標元件（格式、命名空間和名稱）建立，方法是將正斜線新增至開頭，並以正斜線分隔每個元件。例如，命名空間中名為 anycompany-ui-components 的 npm 套件的套件路徑為 /npm/space/anycompany-ui-components。

套件群組模式遵循與套件路徑相同的結構，除了未指定為群組定義一部分的元件外，會省略，且模式會以尾碼終止。包含的尾碼決定模式的相符行為，如下所示：

- $ 尾碼將符合完整的套件座標。
- ~ 字尾將符合字首。
- * 尾碼將符合先前定義元件的所有值。

以下是每個允許組合的範例模式：

1. 所有套件格式： /*
2. 特定套件格式： /npm/*
3. 套件格式和命名空間字首： /maven/com.anycompany~
4. 套件格式和命名空間： /npm/space/*
5. 套件格式、命名空間和名稱字首： /npm/space/anycompany-ui~
6. 套件格式、命名空間和名稱： /maven/org.apache.logging.log4j/log4j-core$

如上述範例所示，~尾碼會新增至命名空間或名稱的結尾，以代表字首相符項目，並在用於比對路徑中下一個元件的所有值（所有格式、所有命名空間或所有名稱）時*出現正斜線。

套件群組定義和標準化

CodeArtifact 會標準化 NuGet、Python 和 Swift 套件名稱，並在儲存 Swift 套件命名空間之前將其標準化。CodeArtifact 會在將套件與套件群組定義相符時，使用這些標準化名稱。因此，包含這些格式的命名空間或名稱的套件群組必須使用標準化命名空間和名稱。如需套件名稱和命名空間如何標準化的詳細資訊，請參閱 NuGet、Python 和 Swift 名稱標準化文件。

套件群組定義中的命名空間

對於沒有命名空間的套件或套件格式 (Python 和 NuGet)，套件群組不得包含命名空間。這些套件群組的套件群組定義包含空白命名空間區段。例如，名為 請求的 Python 套件路徑為 /python//requests。

對於具有命名空間 (Maven、一般和 Swift) 的套件或套件格式，如果包含套件名稱，則必須包含命名空間。對於 Swift 套件格式，將使用標準化套件命名空間。如需如何標準化 Swift 套件命名空間的詳細資訊，請參閱 Swift 套件名稱和命名空間標準化。

套件群組階層和模式特異性

「in」或「associated」套件群組的套件是具有符合群組模式但不符合更特定群組模式之套件路徑的套件。例如，假設套件群組 /npm/*和 /npm/space/*，套件路徑 /npm//react 與第一個群組 (/npm/*) 相關聯，而 /npm/space/aui.components 和 /npm/space/amplify-ui-core 與第二個群組 (/npm/space/

*) 相關聯。即使套件可能符合多個群組，每個套件只會與單一群組相關聯、最具體的相符項目，而且只有一個群組的組態適用於套件。

當套件路徑符合多個模式時，可將「更具體」模式視為最長的相符模式。或者，更具體的模式是符合符合較不具體模式之套件適當子集的模式。從我們先前的範例中，每個符合 的套件/npm/space/*也都符合 /npm/*，但反向不是 true，這會讓模式/npm/space/*更具體，因為它是 的適當子集/npm/*。由於一個群組是另一個群組的子集，因此會建立階層，其中 /npm/space/* 是父群組的子群組 /npm/*。

雖然只有最特定的套件群組組態適用於套件，但該群組可設定為繼承其父群組的組態。

字詞、字詞邊界和字首比對

在討論字首比對之前，讓我們定義一些關鍵術語：

- 字母或數字後面加上零個或多個字母、數字或標記字元的單字（例如重音、跛行等）。
- 當達到非單字字元時，單字邊界位於單字結尾。非單字字元是標點符號字元.，例如 -、 和 _。

具體而言，單字的 regex 模式是 [\p{L}\p{N}][\p{L}\p{N}\p{M}]*，可依下列方式細分：

- \p{L} 代表任何字母。
- \p{N} 代表任何數字。
- \p{M} 代表任何標記字元，例如重音、跛行等。

因此， [\p{L}\p{N}] 代表數字或字母，並[\p{L}\p{N}\p{M}]*代表零個或多個字母、數字或標記字元，且單字邊界位於此規則運算式模式的每個相符項目的結尾。

> **Note**
>
> 單字邊界比對是根據「單字」的定義。它不是以字典或 CameCase 中定義的單字為基礎。例如， oneword或 中沒有單字邊界OneWord。

現在已定義單字和單字邊界，我們可以使用它們來描述 CodeArtifact 中的字首比對。若要在單字邊界上指出字首相符項目，會在單字字元之後使用相符字元 (~)。例如， 模式/npm/space/foo~符合套件路徑 /npm/space/foo和 /npm/space/foo-bar，但不符合 /npm/space/food或 /npm/space/foot。

必須使用萬用字元 (*)，而不是在遵循非單字字元~時，例如在模式 中/npm/*。

區分大小寫

套件群組定義會區分大小寫，這表示僅因大小寫而不同的模式可以存在為個別的套件群組。例如，使用者可以/npm//asyncstorage$為 npm Public Registry：AsyncStorage/npm//asyncStorage$、asyncStorage、非同步儲存中存在的三個不同套件，建立具有模式 /npm//AsyncStorage$、 和的個別套件群組。

雖然大小寫很重要，但如果套件的模式變化因大小寫而異，CodeArtifact 仍會將套件與套件群組建立關聯。如果使用者建立/npm//AsyncStorage$套件群組時未建立上述的其他兩個群組，則名稱 AsyncStorage 的所有案例變化，包括 asyncStorage 和非同步儲存，都會與套件群組建立關聯。但是，如下一節所述，強配對和弱配對這些變化的處理方式將與完全符合模式的 AsyncStorage 不同。

強配對和弱配對

上一節中的資訊 區分大小寫指出套件群組區分大小寫，然後繼續說明它們不區分大小寫。這是因為 CodeArtifact 中的套件群組定義具有強烈相符 （或完全相符） 和弱相符 （或變異相符） 的概念。強烈比對是指套件完全符合模式，沒有任何變化。較弱的相符項目是套件符合模式變化時，例如不同的字母大小寫。弱比對行為可防止套件群組模式變化的套件彙總到更一般的套件群組。當套件是最特定相符群組模式的變化 （弱比對） 時，套件會與群組相關聯，但套件會遭到封鎖，而不是套用群組的原始控制組態，以防止從上游提取或發佈套件的任何新版本。此行為可降低因具有幾乎相同名稱之套件的相依性混淆而造成供應鏈攻擊的風險。

為了說明較弱的比對行為，假設套件群組/npm/*允許擷取和區塊發佈。更具體/npm//anycompany-spicy-client$的套件群組 設定為封鎖擷取並允許發佈。名為 anycompany-spicy-client 的套件是套件群組的強烈配對，允許發佈套件版本並封鎖套件版本的擷取。允許發佈套件名稱的唯一大小寫是anycompany-spicy-client，因為它是套件定義模式的強烈相符項目。不同的案例變化，例如 AnyCompany-spicy-client，會遭到封鎖而無法發佈，因為它是較弱的配對。更重要的是，套件群組會封鎖所有案例變化的擷取，而不只是模式中使用的小寫名稱，可降低相依性混淆攻擊的風險。

其他變化

除了大小寫差異之外，較弱的比對也會忽略破折號 -、點 .、底線 _和可混淆字元 （例如來自不同字母的類似外觀字元） 序列的差異。在用於弱比對的正規化期間，CodeArtifact 會執行大小寫摺疊 （類似於轉換為小寫）、將破折號、點和底線字元的序列取代為單一點，以及正規化可混淆字元。

弱比對會將破折號、點和底線視為對等，但不會完全忽略它們。這表示 foo-bar、foo.bar、foo..bar 和 foo_bar 都是弱配對對等項目，但 foobar 不是。雖然數個公有儲存庫實作步驟來防止這些類型的變異，

但公有儲存庫提供的保護不會讓套件群組的這項功能成為不必要的。例如，如 npm 公有登錄檔登錄檔等公有儲存庫，只有在 my-package 已發佈時，才會防止名為 my-package 的套件出現新的變化。如果 my-package 是允許/npm//my-package$發佈和封鎖擷取的內部套件，則您可能不想將 my-package 發佈至 npm Public Registry，以防止允許 my.package 等變體。

雖然 Maven 等某些套件格式會以不同方式處理這些字元 (Maven 將 . 視為命名空間階層分隔符號，但不會視為 -或 _)，但 com.act-on 之類的內容仍可能與 com.act.on 混淆。

> **Note**
>
> 請注意，每當多個變化與套件群組相關聯時，管理員可以為特定變化建立新的套件群組，以為該變化設定不同的行為。

在 CodeArtifact 中標記套件群組

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在 CodeArtifact 中將標籤套用至套件群組。如需 CodeArtifact 資源標記、使用案例、標籤索引鍵和值限制，以及支援的資源類型的相關資訊，請參閱標記 資源。

您可以在建立套件群組時，使用 CLI 指定標籤，或新增、移除或更新現有套件群組的標籤值。

標籤套件群組 (CLI)

您可以使用 CLI 來管理套件群組標籤。

如果您尚未設定 ， AWS CLI 請依照中的步驟進行設定使用 AWS CodeArtifact 設定。

> **Tip**
>
> 若要新增標籤，您必須提供套件群組的 Amazon Resource Name (ARN)。若要取得套件群組的 ARN，請執行 `describe-package-group`命令：
>
> ```
> aws codeartifact describe-package-group \
>    --domain my_domain \
>    --package-group /npm/scope/anycompany~ \
>    --query packageGroup.arn
> ```

主題

- 將標籤新增至套件群組 (CLI)
- 檢視套件群組的標籤 (CLI)
- 編輯套件群組的標籤 (CLI)
- 從套件群組移除標籤 (CLI)

將標籤新增至套件群組 (CLI)

您可以在套件群組建立時，將標籤新增至套件群組，或將標籤新增至現有的套件群組。如需有關在建立標籤時將標籤新增至套件群組的資訊，請參閱 建立套件群組。

若要使用 將標籤新增至現有的套件群組 AWS CLI，請在終端機或命令列執行 tag-resource命令，指定您要新增標籤的套件群組的 Amazon Resource Name (ARN)，以及您要新增標籤的索引鍵和值。如需套件群組 ARNs 的詳細資訊，請參閱 套件群組 ARNs。

您可以將多個標籤新增至套件群組。例如，若要標記套件群組，*/npm/scope/anycompany~* 有兩個標籤、一個標籤索引鍵名為 *key1* 且標籤值為 *value1*，以及一個標籤索引鍵名為 *key2* 且標籤值為 *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
```

如果成功，此命令沒有輸出。

檢視套件群組的標籤 (CLI)

請依照下列步驟使用 AWS CLI 來檢視套件群組的 AWS 標籤。若未新增標籤，傳回的清單空白。

在終端機或命令列，使用套件群組的 Amazon Resource Name (ARN) 執行 list-tags-for-resource命令。如需套件群組 ARNs 的詳細資訊，請參閱 套件群組 ARNs。

例如，若要檢視套件群組的標籤索引鍵和標籤值清單，請命名為 */npm/scope/anycompany~*，其 ARN 值為 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~
```

若成功，此命令會傳回類似如下的資訊：

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

編輯套件群組的標籤 (CLI)

請依照下列步驟使用 AWS CLI 編輯套件群組的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。您也可以從套件群組移除標籤，如下一節所示。

在終端機或命令列，執行 tag-resource命令，指定您要更新標籤的套件群組 ARN，並指定標籤索引鍵和標籤值。如需套件群組 ARNs 的詳細資訊，請參閱 套件群組 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
```

如果成功，此命令沒有輸出。

從套件群組移除標籤 (CLI)

請依照下列步驟，使用 從套件群組 AWS CLI 移除標籤。

> **Note**
>
> 如果您刪除套件群組，所有標籤關聯都會從已刪除的套件群組中移除。刪除套件群組之前，您不需要移除標籤。

在終端機或命令列，執行 untag-resource命令，指定您要移除標籤的套件群組 ARN，以及您要移除之標籤的標籤索引鍵。如需套件群組 ARNs的相關資訊，請參閱 套件群組 ARNs。

例如，若要使用標籤索引鍵 *key1* 和 *key2* 移除套件群組上的多個標籤，*/npm/scope/anycompany~*：

```
aws codeartifact untag-resource \
```

```
    --resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
group/my_domain/npm/scope/anycompany~ \
    --tag-keys key1 key2
```

如果成功，此命令沒有輸出。移除標籤後，您可以使用 `list-tags-for-resource`命令檢視套件群組上的其餘標籤。

在 CodeArtifact 中使用網域

CodeArtifact 網域可讓您更輕鬆地管理跨組織的多個儲存庫。您可以使用網域在不同 AWS 帳戶擁有的許多儲存庫中套用許可。資產只會存放在網域中一次，即使可從多個儲存庫取得也一樣。

雖然您可以擁有多個網域，但我們建議您使用包含所有已發佈成品的單一生產網域，以便您的開發團隊可以尋找和共用套件。您可以使用第二個生產前網域來測試生產網域組態的變更。

這些主題說明如何使用 CodeArtifact 主控台 AWS CLI、 和 AWS CloudFormation 來建立或設定 CodeArtifact 網域。

主題

- 網域概觀
- 建立網域
- 刪除網域
- 網域政策
- 在 CodeArtifact 中標記網域

網域概觀

當您使用 CodeArtifact 時，網域對下列項目很有用：

- 重複資料刪除的儲存：即使資產可在 1 或 1，000 個儲存庫中使用，也只需要在網域中儲存一次資產。這表示您只需支付一次儲存費用。
- 快速複製：當您將套件從上游 CodeArtifact 儲存庫提取到下游或使用 CopyPackageVersions API 時，只需要更新中繼資料記錄。不會複製任何資產。這可讓您快速設定用於預備或測試的新儲存庫。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。
- 輕鬆跨儲存庫和團隊共用：網域中的所有資產和中繼資料都會使用單一 AWS KMS key (KMS 金鑰) 加密。您不需要管理每個儲存庫的金鑰，或授予多個帳戶存取單一金鑰的權限。
- 將政策套用至多個儲存庫：網域管理員可以將政策套用至整個網域。這包括限制哪些帳戶可以存取網域中的儲存庫，以及誰可以設定與公有儲存庫的連線，以用作套件的來源。如需詳細資訊，請參閱網域政策。
- 唯一儲存庫名稱：網域提供儲存庫的命名空間。儲存庫名稱在網域中只需要是唯一的。您應該使用易於理解的有意義的名稱。

網域名稱在 帳戶中必須是唯一的。

您無法在沒有網域的情況下建立儲存庫。當您使用 CreateRepository API 建立儲存庫時，您必須指定網域名稱。您無法將儲存庫從一個網域移至另一個網域。

儲存庫可以由擁有網域的相同 AWS 帳戶或不同的帳戶所擁有。如果擁有的帳戶不同，則必須將網域資源的`CreateRepository`許可授予擁有儲存庫的帳戶。您可以使用 PutDomainPermissionsPolicy 命令將資源政策新增至網域來執行此操作。

雖然組織可以有多個網域，但建議具有包含所有已發佈成品的單一生產網域，以便開發團隊可以在其組織中尋找和共用套件。第二個生產前網域可用於測試生產網域組態的變更。

跨帳戶網域

網域名稱在帳戶中只需要是唯一的，這表示區域中可能有多個網域名稱具有相同的名稱。因此，如果您想要存取未經身分驗證之帳戶擁有的網域，則必須在 CLI 和主控台中提供網域擁有者 ID 以及網域名稱。請參閱下列 CLI 範例。

存取您經過身分驗證的帳戶所擁有的網域：

存取您驗證之帳戶中的網域時，您只需指定網域名稱。下列範例列出您帳戶所擁有 *my_domain* ### *my_repo* 儲存庫中的套件。

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

存取您未驗證之帳戶擁有的網域：

存取您未驗證之帳戶所擁有的網域時，您需要指定網域擁有者以及網域名稱。下列範例會列出其他##網域中其他###儲存庫中的套件，該網域是由您未驗證的帳戶所擁有。請注意新增 `--domain-owner` 參數。

```
aws codeartifact list-packages --domain other-domain --domain-owner 111122223333 --
repository other-repo
```

CodeArtifact 支援的 AWS KMS 金鑰類型

CodeArtifact 僅支援對稱 KMS 金鑰。您無法使用非對稱 KMS 金鑰來加密 CodeArtifact 網域。如需詳細資訊，請參閱識別對稱和非對稱 KMS 金鑰。若要了解如何建立新的客戶受管金鑰，請參閱《 AWS Key Management Service 開發人員指南》中的建立對稱加密 KMS 金鑰。

CodeArtifact 支援 AWS KMS 外部金鑰存放區 (XKS)。您要負責使用 XKS 金鑰執行金鑰操作的可用性、耐用性和延遲，這可能會影響 CodeArtifact 的可用性、耐用性和延遲。搭配 CodeArtifact 使用 XKS 金鑰的一些效果範例：

- 由於所請求套件及其所有相依性的每個資產都受到解密延遲的影響，因此建置延遲可能會隨著 XKS 操作延遲的增加而大幅增加。
- 由於所有資產都在 CodeArtifact 中加密，因此遺失 XKS 金鑰資料會導致遺失使用 XKS 金鑰與網域相關聯的所有資產。

如需 XKS 金鑰的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的外部金鑰存放區。

建立網域

您可以使用 CodeArtifact 主控台、 AWS Command Line Interface (AWS CLI) 或 建立網域 AWS CloudFormation。當您建立網域時，它不包含任何儲存庫。如需詳細資訊，請參閱建立 儲存庫。如需使用 CloudFormation 管理 CodeArtifact 網域的詳細資訊，請參閱 使用 建立 CodeArtifact 資源 AWS CloudFormation。

主題

- 建立網域 （ 主控台 ）
- 建立網域 (AWS CLI)
- 範例 AWS KMS 金鑰政策

建立網域 （ 主控台 ）

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇建立網域。
3. 在名稱中，輸入網域的名稱。
4. 展開 Additional configuration (其他組態)。
5. 使用 AWS KMS key (KMS 金鑰 ） 加密網域中的所有資產。您可以使用 AWS 受管 KMS 金鑰或您管理的 KMS 金鑰。如需 CodeArtifact 中支援的 KMS 金鑰類型的詳細資訊，請參閱 CodeArtifact 支援的 AWS KMS 金鑰類型。

- 如果您想要使用預設 ，請選擇 AWS 受管金鑰 AWS 受管金鑰。
- 如果您想要使用您管理的 KMS 金鑰，請選擇客戶受管金鑰。若要使用您管理的 KMS 金鑰，請在客戶受管金鑰 ARN 中搜尋並選擇 KMS 金鑰。

如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的 AWS 受管金鑰和客戶受管金鑰。

6. 選擇建立網域。

建立網域 (AWS CLI)

若要使用 建立網域 AWS CLI，請使用 `create-domain`命令。您必須使用 AWS KMS key (KMS 金鑰) 來加密網域中的所有資產。您可以使用 AWS 受管 KMS 金鑰或您管理的 KMS 金鑰。如果您使用 AWS 受管 KMS 金鑰，請勿使用 `--encryption-key` 參數。

如需 CodeArtifact 中支援的 KMS 金鑰類型的詳細資訊，請參閱 CodeArtifact 支援的 AWS KMS 金鑰類型。如需 KMS 金鑰的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的 AWS 受管金鑰和客戶受管金鑰。

```
aws codeartifact create-domain --domain my_domain
```

JSON 格式的資料會顯示在輸出中，其中包含新網域的詳細資訊。

```
{
    "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"
    }
}
```

如果您使用您管理的 KMS 金鑰，請在 `--encryption-key` 參數中包含其 Amazon Resource Name (ARN)。

```
aws codeartifact create-domain --domain my_domain --encryption-key arn:aws:kms:us-
west-2:111122223333:key/your-kms-key
```

JSON 格式的資料會顯示在輸出中，其中包含新網域的詳細資訊。

```
{
    "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"
    }
}
```

使用標籤建立網域

若要建立具有標籤的網域，請將 `--tags` 參數新增至您的`create-domain`命令。

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

範例 AWS KMS 金鑰政策

當您在 CodeArtifact 中建立網域時，您可以使用 KMS 金鑰來加密網域中的所有資產。您可以選擇受 AWS 管 KMS 金鑰或您管理的客戶受管金鑰。如需 KMS 金鑰的詳細資訊，請參閱 AWS Key Management Service 開發人員指南。

若要使用客戶受管金鑰，您的 KMS 金鑰必須具有授予 CodeArtifact 存取權的金鑰政策。金鑰政策是 AWS KMS 金鑰的資源政策，也是控制 KMS 金鑰存取的主要方式。每個 KMS 金鑰都必須只有一個金鑰政策。金鑰政策中的陳述式決定誰有使用 KMS 金鑰的許可以及可以使用它的方式。

下列範例金鑰政策陳述式允許 AWS CodeArtifact 代表授權使用者建立授予和檢視金鑰詳細資訊。此政策陳述式使用 `kms:ViaService`和 `kms:CallerAccount`條件索引鍵，將許可限制為 CodeArtifact 代表指定的帳戶 ID。它也會將所有 AWS KMS 許可授予 IAM 根使用者，以便在建立金鑰之後進行管理。

```
{
```

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

刪除網域

您可以使用 CodeArtifact 主控台或 AWS Command Line Interface () 刪除網域AWS CLI。

主題

- 刪除網域的限制
- 刪除網域 （ 主控台 ）

- 刪除網域 (AWS CLI)

刪除網域的限制

一般而言，您無法刪除包含儲存庫的網域。刪除網域之前，您必須先刪除其儲存庫。如需詳細資訊，請參閱刪除儲存庫。

不過，如果 CodeArtifact 不再能夠存取網域的 KMS 金鑰，您可以刪除網域，即使它仍然包含儲存庫。如果您刪除網域的 KMS 金鑰或撤銷 CodeArtifact 用來存取金鑰的 KMS 授權，就會發生這種情況。在此狀態下，您無法存取網域中的儲存庫或存放在其中的套件。當 CodeArtifact 無法存取網域的 KMS 金鑰時，也無法列出和刪除儲存庫。因此，當網域的 KMS 金鑰無法存取時，網域刪除不會檢查網域是否包含儲存庫。

> **Note**
>
> 當刪除仍包含儲存庫的網域時，CodeArtifact 將在 15 分鐘內非同步刪除儲存庫。刪除網域後，儲存庫仍會顯示在 CodeArtifact 主控台和`list-repositories`命令的輸出中，直到自動清除儲存庫為止。

刪除網域 （主控台）

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇網域，然後選擇您要刪除的網域。
3. 選擇 刪除。

刪除網域 (AWS CLI)

使用 delete-domain命令刪除網域。

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

JSON 格式的資料會顯示在輸出中，其中包含已刪除網域的詳細資訊。

```
{
    "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"
    }
}
```

網域政策

CodeArtifact 支援使用資源型許可來控制存取。以資源為基礎的許可可讓您指定誰可以存取資源，以及他們可以對其執行哪些動作。根據預設，只有擁有網域的 AWS 帳戶才能建立和存取網域中的儲存庫。您可以將政策文件套用至網域，以允許其他 IAM 主體存取它。

如需詳細資訊，請參閱政策和許可以及以身分為基礎的政策和以資源為基礎的政策。

主題

- 啟用網域的跨帳戶存取
- 網域政策範例
- 使用 的網域政策範例 AWS Organizations
- 設定網域政策
- 讀取網域政策
- 刪除網域政策

啟用網域的跨帳戶存取

資源政策是 JSON 格式的文字檔案。檔案必須指定委託人（演員）、一或多個動作和效果 (Allow 或 Deny)。若要在另一個帳戶擁有的網域中建立儲存庫，必須授予主體網域資源的`CreateRepository`許可。

例如，下列資源政策會授予帳戶在網域中建立儲存庫的`123456789012`許可。

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

```
            "Action": [
                "codeartifact:CreateRepository"
            ],
            "Effect": "Allow",
            "Principal": {
                 "AWS": "arn:aws:iam::123456789012:root"
            },
            "Resource": "*"
        }
    ]
}
```

若要允許使用標籤建立儲存庫，您必須包含 codeartifact:TagResource許可。這也會授予帳戶將標籤新增至網域和其中所有儲存庫的存取權。

系統會針對網域和網域內的所有資源，評估網域政策的所有操作。這表示網域政策可用於將許可套用至網域中的儲存庫和套件。當 Resource元素設為 時*，陳述式會套用至網域中的所有資源。例如，如果上述政策也包含在允許的 IAM 動作codeartifact:DescribeRepository清單中，則政策會允許DescribeRepository對網域中的每個儲存庫進行呼叫。網域政策可用來在 Resource元素中使用特定資源 ARNs，將許可套用至網域中的特定資源。

> **Note**
>
> 網域和儲存庫政策都可用來設定許可。當兩個政策都存在時，如果任一政策允許，則會評估這兩個政策並允許 動作。如需詳細資訊，請參閱儲存庫與網域政策之間的互動。

若要存取另一個帳戶所擁有網域中的套件，必須授予主體網域資源的GetAuthorizationToken許可。這可讓網域擁有者控制哪些帳戶可以讀取網域中儲存庫的內容。

例如，下列資源政策會授予帳戶123456789012許可，以擷取網域中任何儲存庫的身分驗證字符。

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

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

> **Note**
>
> 想要從儲存庫端點擷取套件的主體，除了網域的 ReadFromRepository許可之外，還必須授予儲存庫資源的 GetAuthorizationToken許可。同樣地，除了 之外，還必須授予想要將套件發佈至儲存庫端點的主體 PublishPackageVersion許可GetAuthorizationToken。如需 ReadFromRepository和 PublishPackageVersion許可的詳細資訊，請參閱 儲存庫政策。

網域政策範例

當多個帳戶使用網域時，應授予帳戶一組基本許可，以允許完整使用網域。下列資源政策列出一組允許完整使用網域的許可。

```
{
    "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**
>
> 如果網域及其所有儲存庫都由單一帳戶擁有，而且只需要從該帳戶使用，則您不需要建立網域政策。

使用 的網域政策範例 AWS Organizations

您可以使用 `aws:PrincipalOrgID`條件金鑰，從組織中的所有帳戶授予 CodeArtifact 網域的存取權，如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "DomainPolicyForOrganization",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
             "codeartifact:GetDomainPermissionsPolicy",
             "codeartifact:ListRepositoriesInDomain",
             "codeartifact:GetAuthorizationToken",
             "codeartifact:DescribeDomain",
             "codeartifact:CreateRepository"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "aws:PrincipalOrgID":["o-xxxxxxxxxxx"]}
        }
    }
}
```

如需使用 `aws:PrincipalOrgID` 條件金鑰的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 全域條件內容金鑰。

設定網域政策

您可以使用 `put-domain-permissions-policy`命令將政策連接至網域。

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

當您呼叫 時put-domains-permissions-policy，評估許可時會忽略網域上的資源政策。這可確保網域擁有者無法將自己鎖定在網域之外，這會使他們無法更新資源政策。

> **Note**
>
> 您無法將許可授予另一個 AWS 帳戶，以使用資源政策更新網域上的資源政策，因為在呼叫 put-domain-permissions-policy 時會忽略資源政策。

輸出範例：

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

命令的輸出包含網域資源的 Amazon Resource Name (ARN)、政策文件的完整內容，以及修訂識別符。您可以使用 --policy-revision選項put-domain-permissions-policy將修訂識別符傳遞至 。這可確保文件的已知修訂被覆寫，而不是由另一個寫入器設定的較新版本。

讀取網域政策

若要讀取政策文件的現有版本，請使用 get-domain-permissions-policy命令。若要格式化輸出以提高可讀性，請使用 --output和 --query policy.document搭配 Python json.tool模組，如下所示。

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

輸出範例：

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

刪除網域政策

使用 `delete-domain-permissions-policy`命令從網域刪除政策。

```
aws codeartifact delete-domain-permissions-policy --domain my_domain --domain-
owner 111122223333
```

輸出的格式與 `get-domain-permissions-policy`和 `delete-domain-permissions-policy`命令的格式相同。

在 CodeArtifact 中標記網域

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在 CodeArtifact 中將標籤套用至您的網域。如需 CodeArtifact 資源標記、使用案例、標籤索引鍵和值限制，以及支援的資源類型的詳細資訊，請參閱標記 資源。

您可以在建立網域時使用 CLI 來指定標籤。您可以使用 主控台或 CLI 來新增或移除標籤，並更新網域中標籤的值。每個網域最多可以新增 50 個標籤。

主題

- 標籤網域 (CLI)
- 標籤網域 （ 主控台 ）

標籤網域 (CLI)

您可以使用 CLI 來管理網域標籤。

主題

- 將標籤新增至網域 (CLI)
- 檢視網域的標籤 (CLI)
- 編輯網域的標籤 (CLI)
- 從網域移除標籤 (CLI)

將標籤新增至網域 (CLI)

您可以使用 主控台或 AWS CLI 來標記網域。

若要在建立網域時新增標籤，請參閱 建立 儲存庫。

在這些步驟中，我們假設您已經安裝新版 AWS CLI 或更新到最新版本。如需詳細資訊，請參閱安裝 AWS Command Line Interface。

在終端機或命令列，執行 tag-resource命令，指定您要新增標籤之網域的 Amazon Resource Name (ARN)，以及您要新增之標籤的索引鍵和值。

> **Note**
>
> 若要取得網域的 ARN，請執行 `describe-domain`命令：
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

您可以將多個標籤新增至網域。例如，若要使用兩個標籤標記名為 *my_domain* 的網域、標籤值為 value*1 ###### key*1，以及標籤值為 value2 的標籤鍵名為 *key*2：

```
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
```

如果成功，此命令沒有輸出。

檢視網域的標籤 (CLI)

請依照下列步驟使用 AWS CLI 檢視網域的 AWS 標籤。若未新增標籤，傳回的清單空白。

在終端機或命令列，使用網域的 Amazon Resource Name (ARN) 執行 list-tags-for-resource命令。

> **Note**
>
> 若要取得網域的 ARN，請執行 describe-domain命令：
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

例如，若要檢視名為 *my_domain* 且具有 arn:aws:codeartifact:*us-west-2*:*123456789012*:domain/*my_domain* ARN 值之網域的標籤索引鍵和標籤值清單：

```
aws codeartifact list-tags-for-resource --resource-arn arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain
```

若成功，此命令會傳回類似如下的資訊：

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

編輯網域的標籤 (CLI)

請依照下列步驟使用 AWS CLI 編輯網域的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。您也可以從網域移除標籤，如下一節所示。

在終端機或命令列，執行 tag-resource命令，指定您要更新標籤的網域 ARN，並指定標籤索引鍵和標籤值：

> **Note**
>
> 若要取得網域的 ARN，請執行 describe-domain命令：
>
> ```
> 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
```

如果成功，此命令沒有輸出。

從網域移除標籤 (CLI)

請依照下列步驟，使用 從網域 AWS CLI 移除標籤。

> **Note**
>
> 如果您刪除網域，所有標籤關聯都會從已刪除的網域中移除。刪除網域之前，您不需要移除標籤。

在終端機或命令列，執行 untag-resource命令，指定您要移除標籤之網域的 ARN，以及您要移除之標籤的標籤索引鍵。

> **Note**
>
> 若要取得網域的 ARN，請執行 `describe-domain`命令：
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

例如，若要使用標籤索引*# key1* 和 *key2* 移除名為 *mydomain* 之網域上的多個標籤：

```
aws codeartifact untag-resource --resource-arn arn:aws:codeartifact:us-
west-2:123456789012:domain/my_domain --tag-keys key1 key2
```

如果成功，此命令沒有輸出。移除標籤後，您可以使用 `list-tags-for-resource`命令檢視網域上剩餘的標籤。

標籤網域 （ 主控台 ）

您可以使用主控台或 CLI 以標記資源。

主題

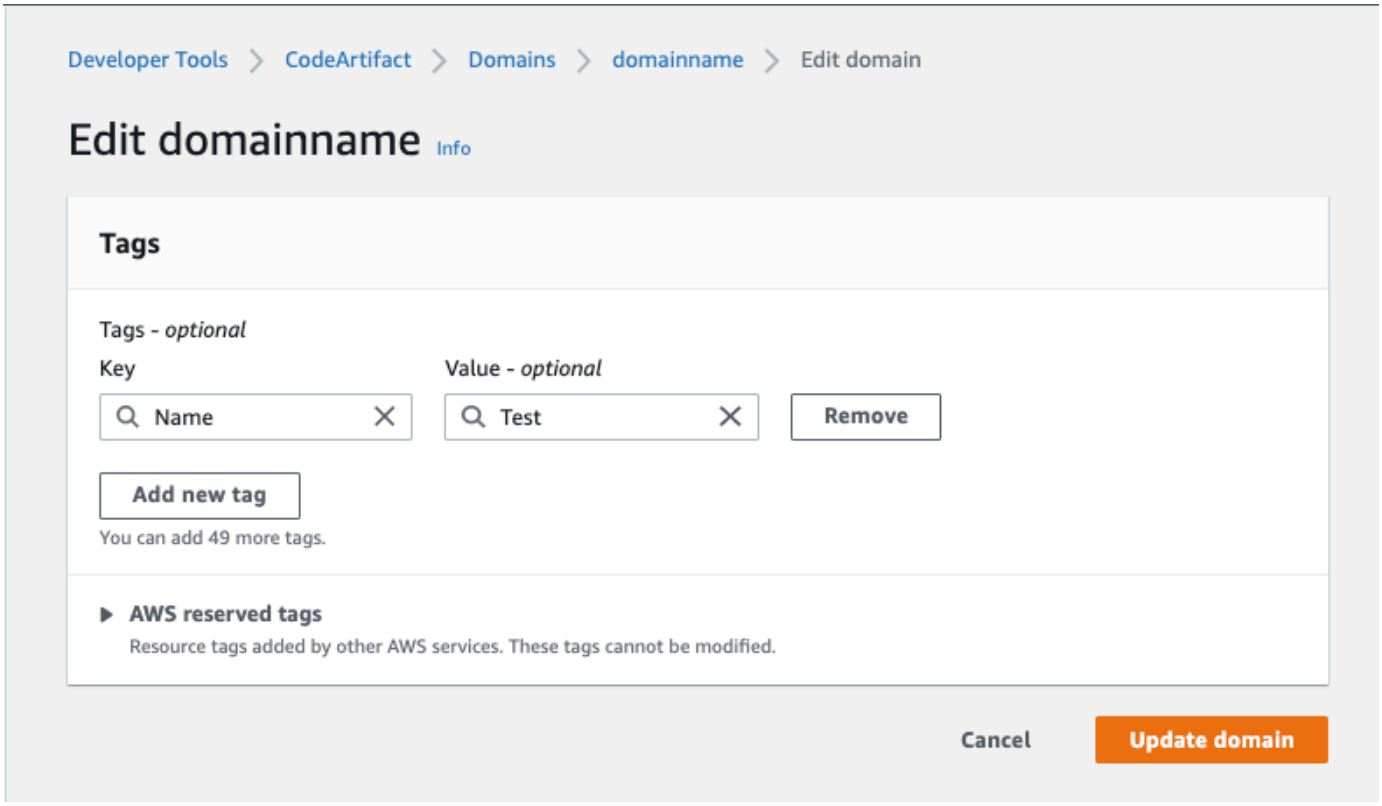
- 將標籤新增至網域 （ 主控台 ）

- 檢視網域的標籤（主控台）
- 編輯網域的標籤（主控台）
- 從網域移除標籤（主控台）

將標籤新增至網域（主控台）

您可以使用 主控台將標籤新增至現有網域。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在網域頁面上，選擇要新增標籤的網域。
3. 展開詳細資訊區段。
4. 在網域標籤下，如果網域上沒有標籤，請選擇新增網域標籤，或者如果有，請選擇檢視和編輯網域標籤。
5. 選擇 Add new tag (新增標籤)。
6. 在金鑰和值欄位中，輸入您要新增的每個標籤的文字。(Value (值) 欄為選用。) 例如，在 Key (索引鍵) 中輸入 **`Name`**。在 Value (值) 中輸入 **`Test`**。



7. (選用) 選擇 Add tag (新增標籤)，新增更多列，然後輸入更多標籤。

8. 選擇更新網域。

檢視網域的標籤 （主控台）

您可以使用 主控台列出現有網域的標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在網域頁面上，選擇您要檢視標籤的網域。
3. 展開詳細資訊區段。
4. 在網域標籤下，選擇檢視和編輯網域標籤。

> **Note**
>
> 如果沒有標籤新增至此網域，主控台會讀取新增網域標籤。

編輯網域的標籤 （主控台）

您可以使用 主控台來編輯已新增至網域的標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在網域頁面上，選擇您要更新標籤的網域。
3. 展開詳細資訊區段。
4. 在網域標籤下，選擇檢視和編輯網域標籤。

> **Note**
>
> 如果沒有標籤新增至此網域，主控台會讀取新增網域標籤。

5. 在 Key (金鑰) 和 Value (加值) 欄，視需要更新每個欄位的值。例如，針對 **`Name`** 索引鍵，在 Value (值) 中將 **`Test`** 變為 **`Prod`**。
6. 選擇更新網域。

從網域移除標籤（主控台）

您可以使用 主控台從網域刪除標籤。

1. 開啟位於 https：//https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在網域頁面上，選擇您要移除標籤的網域。
3. 展開詳細資訊區段。
4. 在網域標籤下，選擇檢視和編輯網域標籤。

> **Note**
>
> 如果沒有標籤新增至此網域，主控台會讀取新增網域標籤。

5. 在要刪除的每個標籤的索引鍵和值旁邊，選擇移除。
6. 選擇更新網域。

搭配貨運使用 CodeArtifact

這些主題說明如何搭配 CodeArtifact 使用 Rust 套件管理工具 Cargo。

> **Note**
>
> CodeArtifact 僅支援 Cargo 1.74.0 及更高版本。Cargo 1.74.0 是支援 CodeArtifact 儲存庫身分驗證的最早版本。

主題

- 透過 CodeArtifact 設定和使用貨運
- 貨運命令支援

透過 CodeArtifact 設定和使用貨運

您可以使用 Cargo 從 CodeArtifact 儲存庫發佈和下載木箱，或從 Rust 社群的木箱登錄檔 crates.io 擷取木箱。本主題說明如何設定 Cargo 來驗證和使用 CodeArtifact 儲存庫。

使用 CodeArtifact 設定貨運

若要使用 Cargo 從 AWS CodeArtifact 安裝和發佈箱子，您必須先使用 CodeArtifact 儲存庫資訊進行設定。請依照下列其中一個程序的步驟，使用 CodeArtifact 儲存庫端點資訊和登入資料來設定貨運。

使用主控台指示設定貨運

您可以使用 主控台中的組態指示，將 Cargo 連線至 CodeArtifact 儲存庫。主控台指示提供針對 CodeArtifact 儲存庫自訂的貨運組態。您可以使用此自訂組態來設定貨運，而無需尋找和填寫 CodeArtifact 資訊。

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇要連線至貨運的儲存庫。
3. 選擇檢視連線指示。
4. 選擇您的作業系統。
5. 選擇貨運。

6. 遵循產生的說明，將貨運連線至 CodeArtifact 儲存庫。

手動設定貨運

如果您無法或不想使用主控台的組態指示，您可以使用下列指示手動將貨運連線至 CodeArtifact 儲存庫。

macOS and Linux

若要使用 CodeArtifact 設定 Cargo，您需要在 Cargo 組態中將 CodeArtifact 儲存庫定義為登錄檔，並提供登入資料。

- 將 *my_registry* 取代為您的登錄檔名稱。
- 以您的 CodeArtifact 網域名稱取代 *my_domain*。
- 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
- 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

複製組態以發佈和下載 Cargo 套件到您的儲存庫，並將其儲存在`~/.cargo/config.toml`檔案中以供系統層級組態或專案層級組態`.cargo/config.toml`使用：

```
[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

若要使用 CodeArtifact 設定 Cargo，您需要在 Cargo 組態中將 CodeArtifact 儲存庫定義為登錄檔，並提供登入資料。

- 以您的登錄檔名稱取代 *my_registry*。

- 以您的 CodeArtifact 網域名稱取代 *my_domain*。
- 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 --domain-owner。如需詳細資訊，請參閱跨帳戶網域。
- 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

將組態複製到僅從您的儲存庫下載貨運套件，並將其儲存在%USERPROFILE%\.cargo\config.toml檔案中，用於系統層級組態或.cargo\config.toml專案層級組態：

```
[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. 若要使用 CodeArtifact 設定 Cargo，您需要在 Cargo 組態中將 CodeArtifact 儲存庫定義為登錄檔，並提供登入資料。

   - 以您的登錄檔名稱取代 *my_registry*。
   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 --domain-owner。如需詳細資訊，請參閱跨帳戶網域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

   複製組態以發佈和下載 Cargo 套件到您的儲存庫，並將其儲存在 %USERPROFILE%\.cargo\config.toml檔案中以供系統層級組態或專案層級組態.cargo\config.toml使用。

   建議您使用登入資料提供者 cargo:token，其會使用存放在您~/.cargo/credentials.toml檔案中的登入資料。如果您使用 cargo publish ，您可能會在 期間遇

到錯誤，cargo:token-from-stdout因為 Cargo 用戶端在 期間未正確裁剪授權字符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. 若要使用 Windows 將 Cargo 套件發佈到您的儲存庫，您必須使用 CodeArtifact get-authorization-token命令和 Cargo login命令來擷取授權字符和您的登入資料。

   - 將 *my_registry* 取代為您的登錄檔名稱，如 中所定義[registries.*my_registry*]。
   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 --domain-owner。如需詳細資訊，請參閱跨帳戶網域。

```
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**
>
> 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

上述範例中的 [registries.*my_registry*]區段使用 *my_registry*和 提供 index 和 credential-provider資訊來定義登錄檔。

- index 指定登錄檔的索引 URL，這是結尾為 的 CodeArtifact 儲存庫端點/。非 Git 儲存庫的登錄檔需要 sparse+字首。

> **Note**
>
> 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

- `credential-provider` 指定指定登錄檔的登入資料提供者。如果`credential-provider`未設定 ，`registry.global-credential-providers`則會使用 中的提供者。透過將`credential-provider`設定為 `cargo:token-from-stdout`，Cargo 用戶端會在從 CodeArtifact 儲存庫發佈或下載時自動擷取新的授權字符，因此您不需要每 12 小時手動重新整理授權字符。

`[registry]` 區段定義使用的預設登錄檔。

- `default` 指定在 中定義的登錄檔名稱`[registries.my_registry]`，預設在從 CodeArtifact 儲存庫發佈或下載時使用。

`[source.crates-io]` 區段定義未指定預設登錄檔時所使用的預設登錄檔。

- `replace-with = "my_registry"` 會使用 中定義的 CodeArtifact 儲存庫來取代公有登錄檔 crates.io`[registries.my_registry]`。如果您需要從外部連線請求套件，例如 crates.io，則建議使用此組態。

  若要取得 CodeArtifact 的所有優點，例如防止相依性混淆攻擊的套件原始伺服器控制，建議您使用來源取代。使用來源取代時，CodeArtifact 會將所有請求代理至外部連線，並從外部連線複製套件至您的儲存庫。如果沒有取代來源，Cargo 用戶端會根據專案中`Cargo.toml`檔案中的組態直接擷取套件。如果相依性未標示 `registry=my_registry`，則 Cargo 用戶端會直接從 crates.io 擷取相依性，而不與您的 CodeArtifact 儲存庫通訊。

> **Note**
>
> 如果您開始使用來源替換，然後更新您的組態檔案以不使用來源替換，您可能會遇到錯誤。相反的情況也可能導致錯誤。因此，建議您避免變更專案的組態。

安裝貨運箱

使用下列程序從 CodeArtifact 儲存庫或 crates.io 安裝貨運箱。

從 CodeArtifact 安裝貨運箱

您可以使用貨運 (cargo) CLI，從 CodeArtifact 儲存庫快速安裝特定版本的貨運箱。

使用 從 CodeArtifact 儲存庫安裝貨運箱 **cargo**

1. 如果您尚未執行，請依照 中的步驟透過 CodeArtifact 設定和使用貨運設定 cargo CLI，以使用具有適當登入資料的 CodeArtifact 儲存庫。
2. 使用下列命令從 CodeArtifact 安裝貨運箱：

   ```
   cargo add my_cargo_package@1.0.0
   ```

   如需詳細資訊，請參閱「貨運手冊」中的貨運新增。

將貨運箱發佈至 CodeArtifact

使用下列程序，使用 CLI 將貨運箱發佈至 CodeArtifact cargo 儲存庫。

1. 如果您尚未執行，請依照 中的步驟透過 CodeArtifact 設定和使用貨運設定 cargo CLI，以使用具有適當登入資料的 CodeArtifact 儲存庫。
2. 使用下列命令將貨運箱發佈至 CodeArtifact 儲存庫：

   ```
   cargo publish
   ```

   如需詳細資訊，請參閱「貨運手冊」中的貨運發佈。

貨運命令支援

除了不支援的特定命令之外，以下各節摘要 CodeArtifact 儲存庫支援的貨運命令。

內容

- 需要存取登錄檔的支援命令
- 不支援的命令

需要存取登錄檔的支援命令

本節列出 Cargo 命令，其中 Cargo 用戶端需要存取其所設定之登錄檔。已驗證這些命令在針對 CodeArtifact 儲存庫調用時可正常運作。

Command	描述
組建	建置本機套件及其相依性。
檢查	檢查本機套件及其相依性是否有錯誤。
擷取	擷取套件的相依性。
發佈	將套件發佈至登錄檔。

不支援的命令

CodeArtifact 儲存庫不支援這些 Cargo 命令。

Command	描述
擁有者	管理登錄檔上的木箱擁有者。
search	搜尋登錄檔中的套件。

搭配 Maven 使用 CodeArtifact

Maven 儲存庫格式由許多不同的語言使用，包括 Java、Kotlin、Scala 和 Clojure。它受到許多不同的建置工具支援，包括 Maven、Gradle、Scala SBT、Apache Ivy 和 Leiningen。

我們已針對下列版本測試並確認與 CodeArtifact 的相容性：

- 最新 Maven 版本：3.6.3。
- 最新 Gradle 版本：6.4.1。 5.5.1 也已經過測試。
- 最新 Clojure 版本：1.11.1 也已經過測試。

主題

- 搭配 Gradle 使用 CodeArtifact
- 將 CodeArtifact 與 mvn 搭配使用
- 搭配 deps.edn 使用 CodeArtifact
- 使用 curl 發佈
- 使用 Maven 檢查總和
- 使用 Maven 快照
- 從上游和外部連線請求 Maven 套件
- Maven 疑難排解

搭配 Gradle 使用 CodeArtifact

在環境變數中擁有 CodeArtifact 驗證權杖後，如使用環境變數傳遞驗證權杖中所述，請遵循這些指示從 CodeArtifact 儲存庫取用 Maven 套件，並將新套件發佈至該儲存庫。

主題

- 擷取相依性
- 擷取外掛程式
- 發佈成品
- 在 IntelliJ IDEA 中執行 Gradle 建置

擷取相依性

若要從 Gradle 組建中的 CodeArtifact 擷取相依性，請使用下列程序。

在 Gradle 建置中從 CodeArtifact 擷取相依性

1. 如果您尚未建立 CodeArtifact 驗證權杖，請依照 中的程序，將權杖存放在 環境變數中使用 環境變數傳遞身分驗證字符。
2. 將maven區段新增至專案build.gradle檔案中的 repositories區段。

```
maven {
         url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
         credentials {
             username "aws"
             password System.env.CODEARTIFACT_AUTH_TOKEN
         }
}
```

上述範例中url的 是 CodeArtifact 儲存庫的端點。Gradle 使用 端點連線到您的儲存庫。在此範例中， my_domain是您的網域名稱、 111122223333 是網域擁有者的 ID， my_repo是您的儲存庫名稱。您可以使用 get-repository-endpoint AWS CLI 命令來擷取儲存庫的端點。

例如，在名為 *my_domain ######### my_repo* 的儲存庫時，命令如下：

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format maven
```

get-repository-endpoint 命令將傳回儲存庫端點：

```
url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
```

上述範例中的 credentials 物件包含您在步驟 1 中建立的 CodeArtifact 驗證字符，Gradle 會使用此字符來驗證 CodeArtifact。

> **Note**
>
> 若要使用雙堆疊端點，請使用 codeartifact.*region*.on.aws端點。

3. （選用）- 若要使用 CodeArtifact 儲存庫做為專案相依性的唯一來源，`repositories`請從 移除中的任何其他區段`build.gradle`。如果您有多個儲存庫，Gradle 會依列出的順序搜尋每個儲存庫的相依性。
4. 設定儲存庫之後，您可以使用標準 Gradle 語法將專案相依性新增至 `dependencies`區段。

```
dependencies {
    implementation 'com.google.guava:guava:27.1-jre'
    implementation 'commons-cli:commons-cli:1.4'
    testImplementation 'org.testng:testng:6.14.3'
}
```

擷取外掛程式

根據預設，Gradle 會從公有 Gradle 外掛程式入口網站解析外掛程式。若要從 CodeArtifact 儲存庫提取外掛程式，請使用下列程序。

從 CodeArtifact 儲存庫提取外掛程式

1. 如果您尚未建立 CodeArtifact 驗證權杖，請依照 中的程序，將權杖存放在 環境變數中使用 環境變數傳遞身分驗證字符。
2. 將 `pluginManagement`區塊新增至您的 `settings.gradle` 檔案。`pluginManagement` 區塊必須出現在 中的任何其他陳述式之前`settings.gradle`，請參閱下列程式碼片段：

```
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
            }
        }
    }
}
```

這將確保 Gradle 從指定的儲存庫解析外掛程式。儲存庫必須具有與 Gradle 外掛程式入口網站 （ 例如 gradle-plugins-store) 有外部連線的上游儲存庫，以便建置可以使用常用的 Gradle 外掛程式。如需詳細資訊，請參閱 Gradle 文件。

發佈成品

本節說明如何將以 Gradle 建置的 Java 程式庫發佈至 CodeArtifact 儲存庫。

首先，將maven-publish外掛程式新增至專案build.gradle檔案的 plugins區段。

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

接著，將publishing區段新增至專案build.gradle檔案。

```
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
            }
        }
    }
}
```

maven-publish 外掛程式會根據 publishing 區段中version指定的 groupId、 artifactId和產生 POM 檔案。

這些變更為 build.gradle 完成後，請執行下列命令來建置專案，並將其上傳至儲存庫。

```
./gradlew publish
```

使用 `list-package-versions` 來檢查套件是否已成功發佈。

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven\
  --namespace com.company.framework --package my-package-name
```

輸出範例：

```
{
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "example",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

如需詳細資訊，請參閱 Gradle 網站上的這些主題：

- 建置 Java 程式庫
- 將專案發佈為模組

在 IntelliJ IDEA 中執行 Gradle 建置

您可以在 IntelliJ IDEA 中執行 Gradle 建置，從 CodeArtifact 提取相依性。若要使用 CodeArtifact 驗證，您必須向 Gradle 提供 CodeArtifact 授權字符。提供身分驗證字符的方法有三種。

- 方法 1：在 中存放身分驗證字符`gradle.properties`。如果您能夠覆寫或新增至`gradle.properties`檔案的內容，請使用此方法。
- 方法 2：將身分驗證權杖存放在單獨的檔案中。如果您不想修改`gradle.properties`檔案，請使用此方法。
- 方法 3：在 中`aws`以內嵌指令碼執行 ，為每次執行產生新的身分驗證權杖`build.gradle`。如果您希望 Gradle 指令碼在每次執行時擷取新的字符，請使用此方法。字符不會存放在檔案系統上。

Token stored in gradle.properties

方法 1：將身分驗證字符存放在 **gradle.properties**

> **Note**
>
> 此範例顯示位於 中的 gradle.properties 檔案GRADLE_USER_HOME。

1. 使用下列程式碼片段更新您的build.gradle檔案：

```
repositories {
    maven {
            url
 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username "aws"
                password "$codeartifactToken"
            }
    }
}
```

2. 若要從 CodeArtifact 擷取外掛程式，請將 pluginManagement區塊新增至 settings.gradle 檔案。pluginManagement 區塊必須出現在 中的任何其他陳述式之前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. 擷取 CodeArtifact 驗證字符：

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. 將身分驗證字符寫入 gradle.properties 檔案：

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > ~/.gradle/gradle.properties
```

Token stored in separate file

方法 2：將身分驗證字符存放在單獨的檔案中

1. 使用下列程式碼片段更新您的build.gradle檔案：

```
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. 若要從 CodeArtifact 擷取外掛程式，請將 pluginManagement區塊新增至 settings.gradle 檔案。pluginManagement 區塊必須出現在 中的任何其他陳述式之前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. 擷取 CodeArtifact 驗證字符：

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. 將身分驗證字符寫入檔案中指定的build.gradle檔案：

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > file
```

Token generated for each run in build.gradle

方法 3：在 中**aws**以內嵌指令碼執行 ，為每次執行產生新的身分驗證權杖 **build.gradle**

1. 使用下列程式碼片段更新您的build.gradle檔案：

```
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. 若要從 CodeArtifact 擷取外掛程式，請將 pluginManagement區塊新增至 settings.gradle 檔案。pluginManagement 區塊必須出現在 中的任何其他陳述式之前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
            }
        }
    }
}
```

將 CodeArtifact 與 mvn 搭配使用

您可以使用 mvn命令來執行 Maven 組建。本節說明如何設定 mvn 以使用 CodeArtifact 儲存庫。

主題

- 擷取相依性
- 發佈成品
- 發佈第三方成品
- 限制將 Maven 相依性下載到 CodeArtifact 儲存庫
- Apache Maven 專案資訊

擷取相依性

若要mvn設定 從 CodeArtifact 儲存庫擷取相依性，您必須編輯 Maven 組態檔案settings.xml、 和選擇性的專案 POM。

1. 如果您尚未建立並存放 CodeArtifact 驗證字符到環境變數，如中所述使用 環境變數傳遞身分驗證字符，以設定 CodeArtifact 儲存庫的身分驗證。
2. 在 settings.xml (通常位於 ~/.m2/settings.xml) 中，新增參考CODEARTIFACT_AUTH_TOKEN環境變數的<servers>區段，讓 Maven 在 HTTP 請求中傳遞字符。

   ```
   <settings>
   ...
       <servers>
           <server>
               <id>codeartifact</id>
               <username>aws</username>
               <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
           </server>
       </servers>
   ...
   </settings>
   ```

3. 在 <repository> 元素中新增 CodeArtifact 儲存庫的 URL 端點。您可以在 settings.xml或專案的 POM 檔案中執行此操作。

   您可以使用 get-repository-endpoint AWS CLI 命令來擷取儲存庫的端點。

   例如，在名為 *my_domain ######### my_repo* 的儲存庫時，命令如下：

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --repository my_repo --
   format maven
   ```

   get-repository-endpoint 命令將傳回儲存庫端點：

   ```
   url 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   maven/my_repo/'
   ```

> **Note**
>
> 若要使用雙堆疊端點，請使用 codeartifact.*region*.on.aws端點。

將儲存庫端點新增至 settings.xml，如下所示。

```
<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>
```

或者，您可以將 <repositories>區段新增至專案 POM 檔案，以僅針對該專案使用 CodeArtifact。

```
<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**
>
> 您可以在 <id>元素中使用任何值，但 <server>和 <repository>元素中的值必須相同。這可讓指定的登入資料包含在對 CodeArtifact 的請求中。

進行這些組態變更後，您可以建置專案。

```
mvn compile
```

Maven 會記錄其下載到主控台的所有相依性的完整 URL。

```
[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)
```

發佈成品

若要將 Maven 成品發佈mvn至 CodeArtifact 儲存庫，您也必須編輯 ~/.m2/settings.xml和專案 POM。

1. 如果您尚未建立並存放 CodeArtifact 驗證字符到環境變數，如中所述使用 環境變數傳遞身分驗證字符，以設定 CodeArtifact 儲存庫的身分驗證。

2. 將<servers>區段新增至 ，settings.xml並參考 CODEARTIFACT_AUTH_TOKEN環境變數，讓 Maven 在 HTTP 請求中傳遞字符。

```
<settings>
...
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

3. 將<distributionManagement>區段新增至專案的 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>
```

進行這些組態變更後，您可以建置專案並將其發佈至指定的儲存庫。

```
mvn deploy
```

使用 list-package-versions 來檢查套件是否已成功發佈。

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven \
  --namespace com.company.framework --package my-package-name
```

輸出範例：

```
{
    "defaultDisplayVersion": null,
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "my-package-name",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

發佈第三方成品

您可以使用 將第三方 Maven 成品發佈至 CodeArtifact 儲存庫`mvn deploy:deploy-file`。這對於想要發佈成品且只有 JAR 檔案且無法存取套件原始碼或 POM 檔案的使用者很有幫助。

`mvn deploy:deploy-file` 命令會根據命令列中傳遞的資訊產生 POM 檔案。

發佈第三方 Maven 成品

1. 如果您尚未建立並存放 CodeArtifact 驗證字符到環境變數，如中所述使用 環境變數傳遞身分驗證字符，以設定 CodeArtifact 儲存庫的身分驗證。
2. 建立具有下列內容`~/.m2/settings.xml`的檔案：

```
<settings>
    <servers>
        <server>
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
</settings>
```

3. 執行 `mvn deploy:deploy-file` 命令：

```
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**
>
> 上述範例會發佈 commons-cli 1.4。修改 groupId、 artifactID、版本和檔案引數以發佈不同的 JAR。

這些指示是根據指南中的範例，從 Apache Maven 文件將第三方 JARs 部署到遠端儲存庫。

限制將 Maven 相依性下載到 CodeArtifact 儲存庫

如果套件無法從設定的儲存庫擷取，則依預設，mvn命令會從 Maven Central 擷取套件。將mirrors元素新增至 settings.xml ，讓 mvn一律使用您的 CodeArtifact 儲存庫。

```
<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>
```

如果您新增mirrors元素，則您的 settings.xml或 中也必須有pluginRepository元素pom.xml。下列範例會從 CodeArtifact 儲存庫擷取應用程式相依性和 Maven 外掛程式。

```
<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>
```

下列範例會從 CodeArtifact 儲存庫擷取應用程式相依性，並從 Maven Central 擷取 Maven 外掛程式。

```
<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>
```

Apache Maven 專案資訊

如需 Maven 的詳細資訊，請參閱 Apache Maven 專案網站上的這些主題：

- 設定多個儲存庫
- 設定參考
- 分佈管理
- 設定檔

搭配 deps.edn 使用 CodeArtifact

您可以使用 `deps.edn`搭配 `clj` 來管理 Clojure 專案的相依性。本節說明如何設定 `deps.edn` 以使用 CodeArtifact 儲存庫。

主題

- 擷取相依性
- 發佈成品

擷取相依性

若要`Clojure`設定 從 CodeArtifact 儲存庫擷取相依性，您必須編輯 Maven 組態檔案 `settings.xml`。

1. 在 中`settings.xml`，新增參考`CODEARTIFACT_AUTH_TOKEN`環境變數的`<servers>`區段，讓 Clojure 在 HTTP 請求中傳遞字符。

> **Note**
>
> Clojure 預期 settings.xml 檔案位於 `~/.m2/settings.xml`。若為其他位置，請在此位置建立 檔案。

```
<settings>
...
    <servers>
        <server>
```

```
            <id>codeartifact</id>
            <username>aws</username>
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

2. 如果您還沒有 POM xml，請使用 為您的專案產生 POM xml`clj -Spom`。
3. 在您的`deps.edn`組態檔案中，新增與 Maven 的伺服器 ID 相符的儲存庫`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` 保證會先檢查 Maven 程式庫的 `central`和 `clojars` 儲存庫。之後，`deps.edn`將檢查 中列出的其他儲存庫。
> - 為了防止直接從 Clojar 和 Maven Central 下載，`central`且 `clojars` 需要設定為 `nil`。

請確定您在 環境變數中有 CodeArtifact Auth 權杖 （ 請參閱 使用 環境變數傳遞身分驗證字符)。在這些變更之後建置套件時，`deps.edn`將從 CodeArtifact 擷取 中的相依性。

> **Note**
>
> 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

發佈成品

1. 更新您的 Maven 設定`deps.edn`並包含 CodeArtifact 做為 maven 辨識的伺服器 （ 請參閱 擷取相依性)。您可以使用 deps-deploy 等工具，將成品上傳至 CodeArtifact。

2. 在您的 中build.clj，新增deploy任務，將必要的成品上傳到先前的設定codeartifact儲存庫。

```
(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. 執行 命令來發佈成品： `clj -T:build deploy`

如需修改預設儲存庫的詳細資訊，請參閱 Clojure Deps 中的修改預設儲存庫和 CLI 參考原理。

使用 curl 發佈

本節說明如何使用 HTTP `curl` 用戶端將 Maven 成品發佈至 CodeArtifact 儲存庫。如果您沒有或想要在環境中安裝 Maven 用戶端，使用 發佈成品curl會很有用。

使用 發佈 Maven 成品 **`curl`**

1. 遵循中的步驟擷取 CodeArtifact 授權字符，使用 環境變數傳遞身分驗證字符並返回這些步驟。
2. 使用下列curl命令將 JAR 發佈至 CodeArtifact 儲存庫：

   在此程序中的每個curl命令中，取代下列預留位置：

   - 將 *my_domain* 取代為您的 CodeArtifact 網域名稱。
   - 以 CodeArtifact 網域擁有者的 ID 取代 *111122223333*。
   - 將 *us-west-2* 取代為您 CodeArtifact 網域所在的區域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

```
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**
>
> 您必須在 `--data-binary` 參數的值前面加上 `@` 字元。將值放入引號時，`@`必須包含在引號內。

3. 使用下列`curl`命令將 POM 發佈至 CodeArtifact 儲存庫：

```
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. 此時，Maven 成品將位於您的 CodeArtifact 儲存庫中，狀態為 `Unfinished`。若要能夠使用套件，它必須處於 `Published` 狀態。您可以透過將`maven-metadata.xml`檔案`Unfinished`從`Published`上傳至您的套件，或呼叫 UpdatePackageVersionsStatus API 來變更狀態，將套件從 移至 。

   a. 選項 1：使用下列`curl`命令將`maven-metadata.xml`檔案新增至您的套件：

```
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
```

   以下是`maven-metadata.xml`檔案內容的範例：

```
<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. 選項 2：Published使用 UpdatePackageVersionsStatus API 將套件狀態更新為 。

```
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
```

如果您只有成品的 JAR 檔案，則可以使用 將消耗性套件版本發佈至 CodeArtifact 儲存庫mvn。如果您無法存取成品的原始碼或 POM，這很有用。如需詳細資訊，請參閱 發佈第三方成品。

使用 Maven 檢查總和

當 Maven 成品發佈至 AWS CodeArtifact 儲存庫時，與套件中每個資產或檔案相關聯的檢查總和會用來驗證上傳。資產的範例包括 jar、pom 和 war 檔案。對於每個資產，Maven 成品包含多個檢查總和檔案，這些檔案使用資產名稱搭配額外的副檔名，例如 md5或 sha1。例如，名為 的檔案檢查總和檔案my-maven-package.jar可能是 my-maven-package.jar.md5和 my-maven-package.jar.sha1。

> **Note**
>
> Maven 使用 一詞artifact。在本指南中，Maven 套件與 Maven 成品相同。如需詳細資訊，請參閱 AWS CodeArtifact 套件。

檢查總和儲存

CodeArtifact 不會將 Maven 檢查總和儲存為資產。這表示檢查總和不會在 ListPackageVersionAssets API 的輸出中顯示為個別資產。反之，CodeArtifact 計算的檢查總和可用於所有支援的檢查總和類型中的每個資產。例如，在 Maven 套件版本上呼叫 ListPackageVersionAssets 的部分回應commons-lang:commons-lang 2.1為：

```
{
```

```
    "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
    }
}
```

即使檢查總和未儲存為資產，Maven 用戶端仍然可以在預期的位置發佈和下載檢查總和。例如，如果 `commons-lang:commons-lang 2.1` 位於名為 的儲存庫中`maven-repo`，則 JAR 檔案的 SHA-256 檢查總和的 URL 路徑為：

```
/maven/maven-repo/commons-lang/commons-lang/2.1/commons-lang-2.1.jar.sha256
```

如果您要使用一般 HTTP 用戶端，例如 `curl`，將現有的 Maven 套件（例如，先前存放在 Amazon S3 中的套件）上傳至 CodeArtifact，則不需要上傳檢查總和。CodeArtifact 會自動產生它們。如果您想

要驗證資產是否已正確上傳，您可以使用 ListPackageVersionAssets API 操作，將回應中的檢查總和與每個資產的原始檢查總和值進行比較。

發佈期間的檢查總和不相符

除了資產和檢查總和之外，Maven 成品也包含 `maven-metadata.xml` 檔案。Maven 套件的正常發佈順序是先上傳所有資產和檢查總和，然後上傳 `maven-metadata.xml`。例如，假設用戶端設定為發佈 SHA-256 檢查總和檔案，`commons-lang 2.1`前述 Maven 套件版本的發佈順序為：

```
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
```

上傳資產的檢查總和檔案時，例如 JAR 檔案，如果上傳的檢查總和值與 CodeArtifact 計算的檢查總和值不相符，則檢查總和上傳請求會失敗，並顯示 400 （無效的請求） 回應。如果對應的資產不存在，請求將會失敗，並顯示 404 （找不到） 回應。若要避免此錯誤，您必須先上傳資產，然後上傳檢查總和。

`maven-metadata.xml` 上傳時，CodeArtifact 通常會將 Maven 套件版本的狀態從 變更為 `Unfinished` `Published`。如果偵測到任何資產的檢查總和不相符，CodeArtifact `maven-metadata.xml` 將傳回 400 （無效的請求） 以回應發佈請求。此錯誤可能會導致用戶端停止上傳該套件版本的檔案。如果發生這種情況，且`maven-metadata.xml`檔案未上傳，則無法下載已上傳套件版本的任何資產。這是因為套件版本的狀態未設定為 `Published`，且保持 `Unfinished`。

CodeArtifact 允許將進一步資產新增至 Maven 套件版本，即使`maven-metadata.xml`已上傳且套件版本狀態已設為 `Published`。在此狀態中，上傳不相符檢查總和檔案的請求也會失敗，並顯示 400 （無效的請求） 回應。不過，由於套件版本狀態已設為 `Published`，因此您可以從套件下載任何資產，包括檢查總和檔案上傳失敗的資產。下載檢查總和資產時，如果檢查總和檔案上傳失敗，用戶端收到的檢查總和值會是 CodeArtifact 根據上傳的資產資料計算的檢查總和值。

CodeArtifact 檢查總和比較區分大小寫，CodeArtifact 計算的檢查總和格式為小寫。因此，如果`909FA780F76DA393E992A3D2D495F468`上傳檢查總和，它將失敗，檢查總和不相符，因為 CodeArtifact 不會將其視為等於 `909fa780f76da393e992a3d2d495f468`。

從檢查總和不相符復原

如果檢查總和上傳因為檢查總和不相符而失敗，請嘗試下列其中一項來復原：

- 執行再次發佈 Maven 成品的命令。如果網路問題損毀了檢查總和檔案，這可能會起作用。如果這樣可以解決網路問題，檢查總和會相符且下載成功。
- 刪除套件版本，然後重新發佈。如需詳細資訊，請參閱 AWS CodeArtifact API 參考中的 DeletePackageVersions。

使用 Maven 快照

Maven 快照是 Maven 套件的特殊版本，參考最新的生產分支程式碼。它是最終發行版本之前的開發版本。您可以透過附加到套件版本的尾碼SNAPSHOT來識別 Maven 套件的快照版本。例如， 版本的快照1.1為 1.1-SNAPSHOT。如需詳細資訊，請參閱 Apache Maven 專案網站上的什麼是 SNAPSHOT 版本？。

AWS CodeArtifact 支援發佈和使用 Maven 快照。使用以時間為基礎的版本編號的唯一快照是唯一支援的快照。CodeArtifact 不支援 Maven 2 用戶端產生的非唯一快照。您可以將支援的 Maven 快照發佈至任何 CodeArtifact 儲存庫。

主題

- CodeArtifact 中的快照發佈
- 使用快照版本
- 刪除快照版本
- 使用 curl 發佈快照
- 快照和外部連線
- 快照和上游儲存庫

CodeArtifact 中的快照發佈

AWS CodeArtifact 支援用戶端在發佈快照時使用的請求模式mvn，例如 。因此，您可以遵循建置工具或套件管理員的文件，而無需詳細了解 Maven 快照的發佈方式。如果您正在執行更複雜的動作，本節會詳細說明 CodeArtifact 如何處理快照。

當 Maven 快照發佈至 CodeArtifact 儲存庫時，其先前版本會保留在稱為組建的新版本中。每次發佈 Maven 快照時，都會建立新的建置版本。快照的所有先前版本都會維護在其建置版本中。發佈 Maven 快照時，其套件版本狀態會設為 ，Published而包含先前版本的組建狀態會設為 Unlisted。此行為僅適用於 Maven 套件版本，其中套件版本具有 -SNAPSHOT做為尾碼。

例如，名為 的 maven 套件快照版本com.mycompany.myapp:pkg-1會上傳至名為 的 CodeArtifact 儲存庫my-maven-repo。快照版本為 1.0-SNAPSHOT。目前尚未com.mycompany.myapp:pkg-1發佈 的版本。首先，初始建置的資產會發佈於下列路徑：

```
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
```

請注意，時間戳記20210728.194552-1是由發佈快照建置的用戶端產生。

上傳 .pom 和 .jar 檔案之後，儲存庫中唯一存在com.mycompany.myapp:pkg-1的 版本是 1.0-20210728.194552-1。即使上述路徑中指定的版本是 ，也會發生這種情況1.0-SNAPSHOT。目前套件版本狀態為 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"
}
```

接著，用戶端會上傳套件版本的 maven-metadata.xml 檔案：

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

成功上傳 maven-metadata.xml 檔案時，CodeArtifact 會建立1.0-SNAPSHOT套件版本並將1.0-20210728.194552-1版本設定為 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"
}
```

此時，快照版本1.0-SNAPSHOT可以在組建中使用。雖然儲存庫 com.mycompany.myapp:pkg-1中有兩個版本的 my-maven-repo，但它們都包含相同的資產。

```
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"
]
```

在 --package-version 參數變更為 的情況下執行先前顯示的相同list-package-version-assets命令，1.0-20210728.194552-1會產生相同的輸出。

當 1.0-SNAPSHOT 的其他組建新增至儲存庫時，會為每個新組建建立新的Unlisted套件版本。每次1.0-SNAPSHOT都會更新 版本的資產，讓版本一律參考該版本的最新組建。1.0-SNAPSHOT 使用最新資產更新 是透過上傳新建置maven-metadata.xml的檔案來啟動。

使用快照版本

如果您請求快照，Published則會傳回狀態為 的版本。這一律是 Maven 快照的最新版本。您也可以使用建置版本編號 （ 例如，1.0-20210728.194552-1) 來請求特定建置快照，而不是 URL 路徑中的快照版本 （ 例如，1.0-SNAPSHOT)。若要查看 Maven 快照的建置版本，請使用 CodeArtifact API 指南中的 ListPackageVersions API，並將狀態參數設定為 Unlisted。 CodeArtifact

刪除快照版本

若要刪除 Maven 快照的所有建置版本，請使用 DeletePackageVersions API，指定您要刪除的版本。

使用 curl 發佈快照

如果您現有的快照版本存放在 Amazon Simple Storage Service (Amazon S3) 或其他成品儲存庫產品中，建議您將其重新發佈至 AWS CodeArtifact。由於 CodeArtifact 如何支援 Maven 快照（請參閱 CodeArtifact 中的快照發佈），使用一般 HTTP 用戶端如 發佈快照比發佈 Maven 發行版本`curl`更複雜，如 中所述使用 curl 發佈。請注意，如果您使用 `mvn`或 等 Maven 用戶端建置和部署快照版本，則此區段無關。 `gradle`您需要遵循該用戶端的文件。

發佈快照版本涉及發佈一或多個快照版本的組建。在 CodeArtifact 中，如果有 n 個快照版本的組建，將會有 n + 1 個 CodeArtifact 版本：n 個全部組建狀態為 的版本`Unlisted`，以及一個狀態為 的快照版本（最新發佈的組建）`Published`。快照版本（也就是具有包含「-SNAPSHOT」之版本字串的版本，包含一組與最新發佈建置相同的資產。使用 建立此結構的最簡單方式`curl`如下：

1. 使用 發佈所有建置的所有資產`curl`。
2. 使用 發佈最後一個組建的 `maven-metadata.xml` 檔案（也就是具有最新日期時間戳記的組建）`curl`。這將在版本字串中建立具有「`-SNAPSHOT`」的版本，並使用正確的資產集。
3. 使用 UpdatePackageVersionsStatus API 將所有非最新建置版本的狀態設定為 `Unlisted`。

使用下列`curl`命令來發佈`1.0-SNAPSHOT`套件快照版本的快照資產（例如 .jar 和 .pom 檔案）`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
```

使用這些範例時：

- 以您的 CodeArtifact 網域名稱取代 *my_domain*。
- 以 CodeArtifact 網域擁有者的 AWS 帳戶 ID 取代 *111122223333*。
- 將 *us-west-2* 取代 AWS 區域 為您的 CodeArtifact 網域所在的 。
- 將 *my_maven_repo* 取代為您的 CodeArtifact 儲存庫名稱。

> **Important**
>
> 您必須在 `--data-binary` 參數的值前面加上 `@` 字元。將值放入引號時，`@`必須包含在引號內。

每個組建可能要上傳兩個以上的資產。例如，除了主要 JAR 和 之外，可能還有 Javadoc 和來源 JAR 檔案`pom.xml`。不需要發佈套件版本資產的檢查總和檔案，因為 CodeArtifact 會自動為每個上傳的資產產生檢查總和。若要驗證資產是否已正確上傳，請使用 `list-package-version-assets`命令擷取產生的檢查總和，並將其與原始檢查總和進行比較。如需 CodeArtifact 如何處理 Maven 檢查總和的詳細資訊，請參閱 使用 Maven 檢查總和。

使用下列 curl 命令來發佈最新建置版本的 `maven-metadata.xml` 檔案：

```
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
```

`maven-metadata.xml` 檔案必須至少參考 `<snapshotVersions>`元素中最新建置版本的其中一個資產。此外，`<timestamp>`值必須存在，且必須符合資產檔案名稱中的時間戳記。例如，對於先前發佈的`20210729.171330-2`組建， 的內容`maven-metadata.xml`會是：

```
<?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>
```

在 發佈maven-metadata.xml之後，最後一個步驟是將所有其他建置版本 （ 也就是除了最新建置以外的所有建置版本 ） 設定為套件版本狀態為 Unlisted。例如，如果1.0-SNAPSHOT版本有兩個組建，而第一個組建是 20210728.194552-1，則將該組建設定為 的命令Unlisted為：

```
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
```

快照和外部連線

Maven 快照無法透過外部連線從 Maven 公有儲存庫擷取。 AWS CodeArtifact 僅支援匯入 Maven 發行版本。

快照和上游儲存庫

一般而言，Maven 快照在與上游儲存庫搭配使用時的運作方式與 Maven 發行版本相同，但如果您計劃將相同套件版本的快照發佈到兩個與上游關係的儲存庫，則會有限制。例如，假設 an AWS CodeArtifact 網域中有兩個儲存庫，R而 U，其中 U是 的上游R。如果您在 中發佈新組建R，當 Maven 用戶端請求該快照版本的最新組建時，CodeArtifact 會從 傳回最新版本U。這可能是非預期的，因為最新版本現在位於 中R，而不是 U。有兩種方法可以避免這種情況：

1. 如果 1.0-SNAPSHOT中1.0-SNAPSHOT存在 ，請勿發佈快照版本的組建R，例如 。 U
2. 使用 CodeArtifact 套件原始伺服器控制，在 中停用該套件上的上游R。後者可讓您在 中發佈的1.0-SNAPSHOT組建R，但也會R防止 從U尚未保留的套件取得任何其他版本。

從上游和外部連線請求 Maven 套件

匯入標準資產名稱

從公有儲存庫匯入 Maven 套件版本時，例如 Maven Central，AWS CodeArtifact 會嘗試匯入該套件版本中的所有資產。如 中所述使用上游儲存庫請求套件版本，匯入會在下列情況發生：

- 用戶端從 CodeArtifact 儲存庫請求 Maven 資產。
- 套件版本尚未存在於儲存庫或其上游。
- 公有 Maven 儲存庫有可連線的外部連線。

即使用戶端可能只請求一個資產，CodeArtifact 仍會嘗試匯入該套件版本可找到的所有資產。CodeArtifact 如何探索 Maven 套件版本可用的資產取決於特定的公有儲存庫。有些公有 Maven 儲存庫支援請求資產清單，但有些則不支援。對於不提供列出資產方式的儲存庫，CodeArtifact 會產生一組可能存在的資產名稱。例如，當junit 4.13.2請求 Maven 套件版本的任何資產時，CodeArtifact 會嘗試匯入下列資產：

- junit-4.13.2.pom
- junit-4.13.2.jar
- junit-4.13.2-javadoc.jar
- junit-4.13.2-sources.jar

匯入非標準資產名稱

當 Maven 用戶端請求的資產不符合上述其中一種模式時，CodeArtifact 會檢查該資產是否存在於公有儲存庫中。如果資產存在，如果有資產，則會將其匯入並新增至現有的套件版本記錄。例如，Maven 套件版本com.android.tools.build:aapt2 7.3.1-8691043包含下列資產：

- 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

當用戶端請求 POM 檔案時，如果 CodeArtifact 無法列出套件版本的資產，POM 將是唯一匯入的資產。這是因為其他資產都不符合標準資產名稱模式。不過，當用戶端請求其中一個 JAR 資產時，該資產會匯入並新增至存放在 CodeArtifact 中的現有套件版本。最下游儲存庫（用戶端提出請求的儲存庫）中的套件版本以及連接外部連線的儲存庫都會更新為包含新資產，如 中所述來自上游儲存庫的套件保留。

一般而言，一旦套件版本保留在 CodeArtifact 儲存庫中，它不會受到上游儲存庫中的變更影響。如需詳細資訊，請參閱來自上游儲存庫的套件保留。不過，先前描述的非標準名稱 Maven 資產的行為是此規則的例外狀況。雖然下游套件版本不會在沒有用戶端請求額外資產的情況下變更，但在這種情況下，保留的套件版本會在最初保留之後進行修改，因此不可變。此行為是必要的，因為具有非標準名稱的 Maven 資產將無法透過 CodeArtifact 存取。在將套件版本保留在 CodeArtifact 儲存庫之後，如果它們新增至公有儲存庫上的 Maven 套件版本，則行為也會啟用。

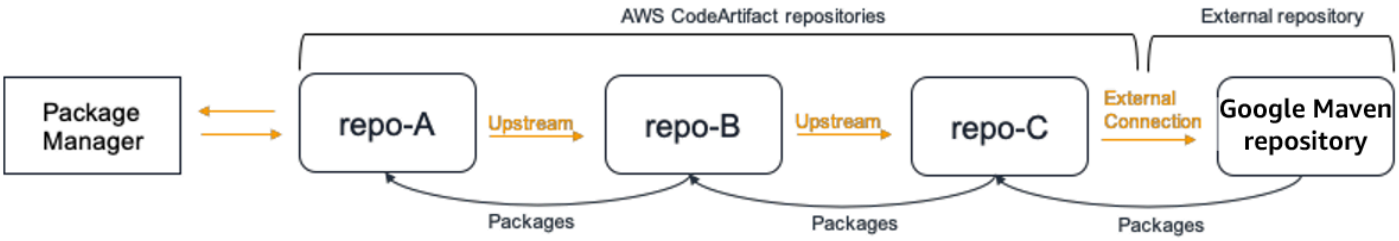
檢查資產原始伺服器

將新資產新增至先前保留的 Maven 套件版本時，CodeArtifact 會確認保留套件版本的原始伺服器與新資產的原始伺服器相同。這可防止建立「混合」套件版本，其中不同的資產來自不同的公有儲存庫。如果沒有此檢查，如果 Maven 套件版本發佈到多個公有儲存庫，且這些儲存庫是 CodeArtifact 儲存庫上游圖形的一部分，則可能會發生資產混合。

在上游儲存庫中匯入新的資產和套件版本狀態

上游儲存庫中套件版本的套件版本狀態可防止 CodeArtifact 在下游儲存庫中保留這些版本。

例如，假設網域有三個儲存庫：repo-A、 repo-B和 repo-C，其中 repo-B 是 的 upsteamrepo-A， repo-C 位於 上游repo-B。



Maven 套件7.3.1的套件版本com.android.tools.build:aapt2存在於 中，repo-B且狀態為 Published。它不存在於 中repo-A。如果用戶端向 請求此套件版本的資產repo-A，回應將為 200

(OK)，Maven 套件版本7.3.1將保留在 中repo-A。不過，如果 7.3.1中的套件版本狀態repo-B為 Archived或 Disposed，則回應將為 404 （找不到），因為這兩個狀態的套件版本的資產無法下載。

請注意，在 repo-A、 repo-B和 com.android.tools.build:aapt2中將套件原始伺服器控制設定為 upstream=BLOCK的 repo-C，將防止從 擷取該套件所有版本的新資產repo-A，無論套件版本狀態為何。

Maven 疑難排解

以下資訊可協助您對 Maven 和 CodeArtifact 的常見問題進行疑難排解。

停用平行放置以修正錯誤 429：請求太多

從 3.9.0 版開始，Maven 會平行上傳套件成品（一次最多 5 個檔案）。這可能會導致 CodeArtifact 偶爾以錯誤回應代碼 429（太多請求）回應。如果您遇到此錯誤，您可以停用平行放置來修正它。

若要停用平行放置，請在 settings.xml 檔案中的設定檔false中將 aether.connector.basic.parallelPut 屬性設定為 ，如下列範例所示：

```
<settings>
    <profiles>
        <profile>
            <id>default</id>
            <properties>
                <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
            </properties>
        </profile>
    </profiles>
<settings>
```

如需詳細資訊，請參閱 Maven 文件中的成品解析程式組態選項。

搭配 npm 使用 CodeArtifact

這些主題說明如何搭配 CodeArtifact 使用 Node.js 套件管理員 npm。

> **Note**
>
> CodeArtifact 支援 `node v4.9.1` 和更新版本 和更新版本 和 `npm v5.0.0`和更新版本。

主題

- 搭配 CodeArtifact 設定和使用 npm
- 設定和使用 Yarn 搭配 CodeArtifact
- npm 命令支援
- npm 標籤處理
- 支援 npm 相容套件管理員

搭配 CodeArtifact 設定和使用 npm

在 CodeArtifact 中建立儲存庫之後，您可以使用 npm 用戶端來安裝和發佈套件。使用儲存庫端點和授權字符設定 npm 的建議方法是使用 `aws codeartifact login`命令。您也可以手動設定 npm。

內容

- 使用登入命令設定 npm
- 不使用登入命令設定 npm
- 執行 npm 命令
- 驗證 npm 身分驗證和授權
- 變更回預設 npm 登錄檔
- 使用 npm 8.x 或更高版本對慢速安裝進行故障診斷

使用登入命令設定 npm

使用 `aws codeartifact login`命令來擷取登入資料，以便與 npm 搭配使用。

> **Note**
>
> 如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

> **Important**
>
> 如果您使用的是 npm 10.x 或更新版本，則必須使用 2 AWS CLI .9.5 或更新版本才能成功執行`aws codeartifact login`命令。

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

此命令會對 ~/.npmrc 檔案進行下列變更：

- 使用您的 AWS 登入資料從 CodeArtifact 擷取授權字符後，新增授權字符。
- 將 npm 登錄檔設定為 `--repository`選項指定的儲存庫。
- 對於 npm 6 和更低：新增 ，`"always-auth=true"`以便為每個 npm 命令傳送授權字符。

呼叫 後的預設授權期間`login`為 12 小時，`login`必須呼叫 以定期重新整理字符。如需使用 `login`命令建立的授權字符的詳細資訊，請參閱 使用 `login`命令建立的字符。

不使用登入命令設定 npm

您可以使用 CodeArtifact 儲存庫設定 npm，無需 `aws codeartifact login`命令，方法是手動更新 npm 組態。

設定 npm 而不使用登入命令

1. 在命令列中，擷取 CodeArtifact 授權字符並將其存放在環境變數中。npm 將使用此字符來驗證您的 CodeArtifact 儲存庫。

> **Note**
>
> 下列命令適用於 macOS 或 Linux 機器。如需在 Windows 機器上設定環境變數的資訊，請參閱 使用 環境變數傳遞身分驗證字符。

```
CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

2. 執行下列命令，取得 CodeArtifact 儲存庫的端點。您的儲存庫端點用來將 npm 指向您的儲存庫，以安裝或發佈套件。

   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format npm
```

   下列 URL 是範例儲存庫端點。

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/my_repo/
```

> **Important**
>
> 登錄 URL 必須以正斜線 (/) 結尾。否則，您無法連線至儲存庫。

3. 使用 `npm config set`命令將登錄檔設定為 CodeArtifact 儲存庫。將 URL 取代為上一個步驟的儲存庫端點 URL。

```
npm config set
 registry=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/
```

> **Note**
>
> 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

4. 使用 `npm config set`命令將授權字符新增至 npm 組態。

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:_authToken=$CODEARTIFACT_AUTH_TOKEN
```

對於 npm 6 或更低：若要讓 npm 一律將身分驗證字符傳遞至 CodeArtifact，即使是GET請求，請使用 設定`always-auth`組態變數`npm config set`。

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:always-auth=true
```

範例 npm 組態檔案 (**`.npmrc`**)

以下是遵循上述指示設定 CodeArtifact 登錄端點、新增身分驗證字符和設定 之後的範例`.npmrc`檔案`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
```

執行 npm 命令

設定 npm 用戶端之後，您可以執行 npm 命令。假設套件存在於您的儲存庫或其上游儲存庫之一，您可以使用 安裝套件`npm install`。例如，使用下列命令來安裝 `lodash`套件。

```
npm install lodash
```

使用下列命令將新的 npm 套件發佈至 CodeArtifact 儲存庫。

```
npm publish
```

如需如何建立 npm 套件的資訊，請參閱 npm 文件網站上的建立 Node.js 模組。如需 CodeArtifact 支援的 npm 命令清單，請參閱 npm 命令支援。

驗證 npm 身分驗證和授權

叫用 `npm ping`命令是一種驗證下列項目的方法：

- 您已正確設定您的登入資料，因此您可以向 CodeArtifact 儲存庫進行身分驗證。
- 授權組態會授予您 `ReadFromRepository`許可。

成功調用 的輸出`npm ping`如下所示。

```
$ 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
```

`-d` 選項會導致 npm 列印其他偵錯資訊，包括儲存庫 URL。此資訊可讓您輕鬆地確認 npm 已設定為使用您預期的儲存庫。

變更回預設 npm 登錄檔

使用 CodeArtifact 設定 npm 會將 npm 登錄檔設定為指定的 CodeArtifact 儲存庫。當您完成連線至 CodeArtifact 時，您可以執行下列命令，將 npm 登錄檔設回其預設登錄檔。

```
npm config set registry https://registry.npmjs.com/
```

使用 npm 8.x 或更高版本對慢速安裝進行故障診斷

npm 版本 8.x 及更高版本中存在已知問題，其中，如果向套件儲存庫提出請求，且儲存庫將用戶端重新導向至 Amazon S3，而不是直接串流資產，則 npm 用戶端每個相依性可以掛斷幾分鐘。

由於 CodeArtifact 儲存庫的設計一律會將請求重新導向至 Amazon S3，因此有時候會發生此問題，這會導致建置時間過長，因為 npm 安裝時間過長。此行為的執行個體會以進度列顯示自己幾分鐘。

若要避免此問題，請使用 `--no-progress`或 `progress=false`旗標搭配 npm cli 命令，如下列範例所示。

```
npm install lodash --no-progress
```

設定和使用 Yarn 搭配 CodeArtifact

建立儲存庫之後，您可以使用 Yarn 用戶端來管理 npm 套件。

> **Note**
>
> `Yarn 1.X` 會從 npm 組態檔案 (.npmrc) 讀取和使用資訊，而 `Yarn 2.X` 不會。的組態`Yarn 2.X`必須在 .yarnrc.yml 檔案中定義。

內容

- 使用 aws codeartifact login命令設定 Yarn 1.X
- 使用 yarn config set命令設定 Yarn 2.X

使用 `aws codeartifact login`命令設定 Yarn 1.X

對於 `Yarn 1.X`，您可以使用 `aws codeartifact login`命令設定 Yarn with CodeArtifact。`login`命令將使用 CodeArtifact 儲存庫端點資訊和登入資料來設定 ~/.npmrc 檔案。透過 `Yarn 1.X`，`yarn`命令會使用來自 ~/.npmrc 檔案的組態資訊。

`Yarn 1.X` 使用登入命令設定

1. 如果您尚未這麼做，請設定 AWS 登入資料以搭配 使用 AWS CLI，如 中所述CodeArtifact 入門。
2. 若要成功執行`aws codeartifact login`命令，必須安裝 npm。如需安裝說明，請參閱 npm 文件中的下載和安裝 Node.js 和 npm。
3. 使用 `aws codeartifact login`命令來擷取 CodeArtifact 登入資料，並設定您的 ~/.npmrc 檔案。

    - 以您的 CodeArtifact 網域名稱取代 *`my_domain`*。

- 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
- 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

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

`login` 命令會對 ~/.npmrc 檔案進行下列變更：

- 使用您的 AWS 登入資料從 CodeArtifact 擷取授權字符後，新增授權字符。
- 將 npm 登錄檔設定為 `--repository`選項指定的儲存庫。
- 對於 npm 6 和更低：新增 ，`"always-auth=true"`以便為每個 npm 命令傳送授權字符。

呼叫 後的預設授權期間`login`為 12 小時，`login`必須呼叫 才能定期重新整理字符。如需使用 `login`命令建立的授權字符的詳細資訊，請參閱 使用 `login`命令建立的字符。

4. 對於 npm 7.X 和 8.X，您必須將 `always-auth=true`新增至 ~/.npmrc 檔案，才能使用 Yarn。
   - 在文字編輯器中開啟您的 ~/.npmrc 檔案，並在新行`always-auth=true`新增 。

您可以使用 `yarn config list`命令來檢查 Yarn 是否使用正確的組態。執行 命令後，請檢查 `info npm config`區段中的值。內容看起來應該類似下列程式碼片段。

```
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
}
```

使用 `yarn config set`命令設定 Yarn 2.X

下列程序詳細說明如何使用 命令從命令列`Yarn 2.X`更新`.yarnrc.yml`組態來設定 `yarn config set`。

從命令列更新**yarnrc.yml**組態

1. 如果您尚未這麼做，請設定 AWS 登入資料以搭配 使用 AWS CLI，如 中所述CodeArtifact 入門。
2. 使用 `aws codeartifact get-repository-endpoint`命令來取得 CodeArtifact 儲存庫的端點。

   - 將 *my_domain* 取代為您的 CodeArtifact 網域名稱。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format npm
   ```

3. 使用儲存庫端點更新 .yarnrc.yml 檔案中`npmRegistryServer`的值。

   ```
   yarn config set npmRegistryServer
    "https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"
   ```

4. 擷取 CodeArtifact 授權字符並將其存放在 環境變數中。

   > **Note**
   >
   > 下列命令適用於 macOS 或 Linux 機器。如需在 Windows 機器上設定環境變數的資訊，請參閱 使用 環境變數傳遞身分驗證字符。

   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

   ```
   export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

5. 使用 `yarn config set`命令將 CodeArtifact 身分驗證字符新增至 .yarnrc.yml 檔案。將下列命令中的 URL 取代為步驟 2 中的儲存庫端點 URL。

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
 "${CODEARTIFACT_AUTH_TOKEN}"
```

6. 使用 `yarn config set`命令將 的值設定為 `npmAlwaysAuth` `true`。將下列命令中的 URL 取代為步驟 2 中的儲存庫端點 URL。

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAlwaysAuth'
 "true"
```

設定後，您的 .yarnrc.yml 組態檔案應具有類似下列程式碼片段的內容。

```
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/"
```

您也可以使用 `yarn config`命令來檢查 `npmRegistries`和 的值`npmRegistryServer`。

npm 命令支援

以下各節摘要說明 CodeArtifact 儲存庫所支援的 npm 命令，以及不支援的特定命令。

內容

- 與儲存庫互動的支援命令
- 支援的用戶端命令
- 不支援的命令

與儲存庫互動的支援命令

本節列出 npm 命令，其中 npm 用戶端向已設定的登錄檔提出一或多個請求（例如，使用 `npm config set registry`）。已驗證這些命令在針對 CodeArtifact 儲存庫調用時可正常運作。

Command	描述
錯誤	嘗試猜測套件錯誤追蹤器 URL 的位置，然後嘗試開啟它。
ci	安裝具有乾淨板塊的專案。
棄用	棄用套件的版本。
dist-tag	修改套件分佈標籤。
文件	嘗試猜測套件文件 URL 的位置，然後嘗試使用 `--browser` 組態參數將其開啟。
醫生	執行一組檢查，以確保您的 npm 安裝具有管理 JavaScript 套件所需的項目。
安裝	安裝套件。
install-ci-test	安裝具有乾淨板塊的專案並執行測試。別名：`npm cit`。此命令會立即執行 `npm ci`，後面接著 `npm test`。
安裝測試	安裝套件並執行測試。執行 `npm install`，緊接著 `npm test`。
過時	檢查設定的登錄檔，以查看是否有任何已安裝的套件目前已過期。
ping	Ping 已設定或指定的 npm 登錄檔，並驗證身分驗證。
發佈	將套件版本發佈至登錄檔。

Command	描述
update	猜測套件儲存庫 URL 的位置，然後嘗試使用 `--browser` 組態參數將其開啟。
檢視	顯示套件中繼資料。可用於列印中繼資料屬性。

支援的用戶端命令

這些命令不需要與儲存庫進行任何直接互動，因此 CodeArtifact 不需要執行任何動作來支援儲存庫。

Command	描述
組建	建置套件。
快取	操作套件快取。
完成	在所有 npm 命令中啟用索引標籤完成。
組態	更新使用者和全域`npmrc`檔案的內容。
資料刪除	搜尋本機套件樹狀結構，並嘗試透過將相依性進一步移動至樹狀結構來簡化結構，讓多個相依套件更有效地共用這些屬性。
編輯	編輯已安裝的套件。選取目前工作目錄中的相依性，並在預設編輯器中開啟套件資料夾。
探索	瀏覽已安裝的套件。在指定的已安裝套件目錄中產生子殼。如果指定命令，則會在子殼中執行，然後立即終止。
help	取得 npm 的說明。
help-search	搜尋 npm 說明文件。
初始化	建立 `package.json` 檔案。
連結	Symlinks 套件資料夾。

Command	描述
ls	列出已安裝的套件。
套件	從套件建立 tarball。
prefix	顯示字首。除非-g另有指定，否則這是包含package.json 檔案的最接近的父目錄。
黑棗	移除未列在父系套件相依性清單上的套件。
重建	在相符的資料夾上執行 npm build命令。
重新啟動	執行套件的停止、重新啟動和啟動指令碼，以及相關聯的前置和後置指令碼。
根	列印要標準輸出的有效node_modules 資料夾。
run-script	執行任意套件指令碼。
收縮包裝	鎖定要發佈的相依性版本。
解除安裝	解除安裝套件。

不支援的命令

CodeArtifact 儲存庫不支援這些 npm 命令。

Command	描述	備註
存取	設定已發佈套件的存取層級。	CodeArtifact 使用與公有 npmjs 儲存庫不同的許可模型。
adduser	新增登錄使用者帳戶	CodeArtifact 使用與公有 npmjs 儲存庫不同的使用者模型。

Command	描述	備註
稽核	執行安全稽核。	CodeArtifact 目前未提供安全漏洞資料。
勾點	管理 npm 勾點，包括新增、移除、列出和更新。	CodeArtifact 目前不支援任何類型的變更通知機制。
登入	驗證使用者。這是 `npm adduser` 的別名。	CodeArtifact 使用與公有 npmjs 儲存庫不同的身分驗證模型。如需詳細資訊，請參閱使用 npm 進行身分驗證。
登出	登出登錄檔。	CodeArtifact 使用與公有 npmjs 儲存庫不同的身分驗證模型。無法從 CodeArtifact 儲存庫登出，但身分驗證字符會在其可設定的過期時間後過期。預設字符持續時間為 12 小時。
擁有者	管理套件擁有者。	CodeArtifact 使用與公有 npmjs 儲存庫不同的許可模型。
profile	變更登錄檔設定檔上的設定。	CodeArtifact 使用與公有 npmjs 儲存庫不同的使用者模型。
search	搜尋登錄檔中符合搜尋詞彙的套件。	CodeArtifact 支援使用 list-packages 命令的有限搜尋功能。
星星	標記您最愛的套件。	CodeArtifact 目前不支援任何類型的最愛機制。
星星	檢視標示為我的最愛之套件。	CodeArtifact 目前不支援任何類型的最愛機制。

Command	描述	備註
團隊	管理組織團隊和團隊成員資格。	CodeArtifact 使用與公有 npmjs 儲存庫不同的使用者和群組成員資格模型。如需詳細資訊，請參閱《IAM 使用者指南》中的身分（使用者、群組和角色）。
token	管理您的身分驗證字符。	CodeArtifact 使用不同的模型來取得身分驗證字符。如需詳細資訊，請參閱使用 npm 進行身分驗證。
取消發佈	從登錄檔移除套件。	CodeArtifact 不支援使用 npm 用戶端從儲存庫移除套件版本。您可以使用 delete-package-version 命令。
whoami	顯示 npm 使用者名稱。	CodeArtifact 使用與公有 npmjs 儲存庫不同的使用者模型。

npm 標籤處理

npm 登錄檔支援標籤，這些是套件版本的字串別名。您可以使用標籤來提供別名，而不是版本號碼。例如，您可能有一個具有多個開發串流的專案，並為每個串流使用不同的標籤（例如 、`stablebeta`、`dev`、`canary`)。如需詳細資訊，請參閱 npm 網站上的 dist-tag。

根據預設，npm 會使用 `latest`標籤來識別套件的目前版本。 `npm install` *`pkg`*（不含 `@`*`version`*或 `@`*`tag`* 指標） 會安裝最新的標籤。一般而言，專案只會使用最新的標籤來提供穩定的發行版本。其他標籤用於不穩定或發行前版本。

使用 npm 用戶端編輯標籤

CodeArtifact add儲存庫中的三個`npm dist-tag`命令 (`rm`、 和 `ls`) 函數與預設 npm 登錄檔中的命令相同。

npm 標籤和 CopyPackageVersions API

當您使用 CopyPackageVersions API 複製 npm 套件版本時，所有別名為該版本的標籤都會複製到目的地儲存庫。當複製的版本具有也存在於目的地的標籤時，複製操作會將目的地儲存庫中的標籤值設定為符合來源儲存庫中的值。

例如，假設儲存庫 S 和儲存庫 D 都包含具有最新標籤集的單一版本web-helper套件，如下表所示。

儲存庫	套件名稱	套件標籤
S	web-helper	最新 (1.0.1 版的別名)
D	web-helper	最新 (1.0.0 版的別名)

CopyPackageVersions 被叫用將 web-helper 1.0.1 從 S 複製到 D。操作完成後，儲存庫 D web-helper中的 latest標籤別名為 1.0.1，而不是 1.0.0。

如果您需要在複製後變更標籤，請使用 npm dist-tag命令直接在目的地儲存庫中修改標籤。如需 CopyPackageVersions API 的詳細資訊，請參閱在儲存庫之間複製套件。

npm 標籤和上游儲存庫

當 npm 請求套件的標籤，且該套件的版本也出現在上游儲存庫中時，CodeArtifact 會先合併標籤，再將其傳回用戶端。例如，名為 R 的儲存庫具有名為 U 的上游儲存庫。下表顯示兩個儲存庫中存在web-helper的名為 的套件標籤。

儲存庫	套件名稱	套件標籤
R	web-helper	最新 (1.0.0 版的別名)
U	web-helper	alpha (1.0.1 版的別名)

在此情況下，當 npm 用戶端從儲存庫 R 擷取web-helper套件的標籤時，它會同時接收最新的標籤和 Alpha 標籤。標籤指向的版本不會變更。

當上游和下游儲存庫中的相同套件上都存在相同的標籤時，CodeArtifact 會使用上游儲存庫中存在的標籤。例如，假設 webhelper 上的標籤已修改為如下所示。

儲存庫	套件名稱	套件標籤
R	web-helper	最新 (1.0.0 版的別名)
U	web-helper	最新 (1.0.1 版的別名)

在此情況下，當 npm 用戶端從儲存庫 R 擷取套件 Web-helper 的標籤時，最新的標籤會別名為 1.0.1 版，因為這是上游儲存庫中的標籤。這可讓您透過執行 ，在尚未出現在下游儲存庫的上游儲存庫中輕鬆使用新的套件版本npm update。

在下游儲存庫中發佈新版本的套件時，在上游儲存庫中使用 標籤可能會造成問題。例如，假設套件 Web-helper 上的最新標籤在 R 和 U 中都相同。

儲存庫	套件名稱	套件標籤
R	web-helper	最新 (1.0.1 版的別名)
U	web-helper	最新 (1.0.1 版的別名)

當 1.0.2 版發佈至 R 時，npm 會將最新的標籤更新至 1.0.2。

儲存庫	套件名稱	套件標籤
R	web-helper	最新 (1.0.2 版的別名)
U	web-helper	最新 (1.0.1 版的別名)

不過，npm 用戶端永遠不會看到此標籤值，因為 U 中最新的 值是 1.0.1。在發佈 1.0.2 之後，立即npm install針對儲存庫 R 執行 會安裝 1.0.1，而不是剛發佈的版本。若要安裝最近發佈的版本，您必須指定確切的套件版本，如下所示。

```
npm install web-helper@1.0.2
```

支援 npm 相容套件管理員

這些其他套件管理員與 CodeArtifact 相容，並且使用 npm 套件格式和 npm 線路通訊協定：

- pnpm 套件管理員。確認可與 CodeArtifact 搭配使用的最新版本為 3.3.4，已於 2019 年 5 月 18 日發行。
- Yarn 套件管理員。確認可與 CodeArtifact 搭配使用的最新版本為 1.21.1，已於 2019 年 12 月 11 日發行。

> **Note**
>
> 我們建議搭配 CodeArtifact 使用 Yarn 2.x。Yarn 1.x 沒有 HTTP 重試，這表示更容易發生間歇性服務錯誤，導致 500 層級的狀態碼或錯誤。無法為 Yarn 1.x 設定不同的重試策略，但已將其新增至 Yarn 2.x。您可以使用 Yarn 1.x，但您可能需要在建置指令碼中新增更高層級的重試。例如，在迴圈中執行您的 yarn 命令，以便在下載套件失敗時重試。

搭配 NuGet 使用 CodeArtifact

這些主題說明如何使用 CodeArtifact 來使用和發佈NuGet套件。

> **Note**
>
> AWS CodeArtifact 僅支援 NuGet.exe 4.8 版和更新版本。

主題

- 搭配 Visual Studio 使用 CodeArtifact
- 搭配 nuget 或 dotnet CLI 使用 CodeArtifact
- NuGet 套件名稱、版本和資產名稱標準化
- NuGet 相容性

搭配 Visual Studio 使用 CodeArtifact

您可以使用 CodeArtifact 登入資料提供者，在 Visual Studio 中直接使用 CodeArtifact 套件。登入資料提供者可簡化在 Visual Studio 中 CodeArtifact 儲存庫的設定和身分驗證，並可在 中使用AWS Toolkit for Visual Studio。

> **Note**
>
> AWS Toolkit for Visual Studio 不適用於 Visual Studio for Mac。

若要搭配 CLI 工具設定和使用 NuGet，請參閱 搭配 nuget 或 dotnet CLI 使用 CodeArtifact 。

主題

- 使用 CodeArtifact 登入資料提供者設定 Visual Studio
- 使用 Visual Studio Package Manager 主控台

使用 CodeArtifact 登入資料提供者設定 Visual Studio

CodeArtifact 登入資料提供者可簡化 CodeArtifact 和 Visual Studio 之間的設定和持續身分驗證。CodeArtifact 身分驗證字符的有效期最長為 12 小時。為了避免在 Visual Studio 中工作時手動重新整理權杖，憑證提供者會在目前的權杖過期之前定期擷取新的權杖。

> **Important**
>
> 若要使用登入資料提供者，請確定任何現有的 AWS CodeArtifact 登入資料都已從您的`nuget.config`檔案清除，這些登入資料可能已手動新增，或執行 `aws codeartifact login`來設定 NuGet。

在 Visual Studio 中使用 CodeArtifact 搭配 AWS Toolkit for Visual Studio

1. AWS Toolkit for Visual Studio 使用下列步驟安裝 。工具組與使用下列步驟的 Visual Studio 2017 和 2019 相容。 AWS CodeArtifact 不支援 Visual Studio 2015 和更早版本。
   1. Toolkit for Visual Studio for Visual Studio 2017 和 Visual Studio 2019 分佈在 Visual Studio Marketplace 中。您也可以使用 Tools ≫ Extensions and Updates (Visual Studio 2017) 或 Extensions ≫ Manage Extensions (Visual Studio 2019)，在 Visual Studio 中安裝和更新工具組。
   2. 安裝工具組後，從檢視功能表中選擇 AWS Explorer 來開啟工具組。
2. 遵循AWS Toolkit for Visual Studio 《 使用者指南》中的提供登入 AWS 資料中的步驟，以您的登入資料設定 Toolkit for Visual Studio。 AWS
3. （選用）設定您要搭配 CodeArtifact 使用的 AWS 設定檔。如果未設定，CodeArtifact 將使用預設設定檔。若要設定設定檔，請前往工具 > NuGet 套件管理員 > 選取 CodeArtifact AWS 設定檔。
4. 在 Visual Studio 中將 CodeArtifact 儲存庫新增為套件來源。
   1. 在 AWS Explorer 視窗中導覽至您的儲存庫，按一下滑鼠右鍵並選取 `Copy NuGet Source Endpoint`。
   2. 使用工具 > 選項命令並捲動至 NuGet Package Manager。
   3. 選取套件來源節點。
   4. 選取 ＋，編輯名稱，並將步驟 3a 中複製的儲存庫 URL 端點貼到來源方塊中，然後選取更新。
   5. 選取新新增套件來源的核取方塊以啟用它。

> **Note**
>
> 建議您將 NuGet.org 的外部連線新增至 CodeArtifact 儲存庫，並在 Visual Studio 中停用 nuget.org 套件來源。使用外部連線時，從 NuGet.org 擷取的所有套件都會存放在 CodeArtifact 儲存庫中。如果 NuGet.org 無法使用，您的應用程式相依性仍然可用於 CI 組建和本機開發。如需外部連線的詳細資訊，請參閱 將 CodeArtifact 儲存庫連線至公有儲存庫。

5. 重新啟動 Visual Studio 以使變更生效。

設定之後，如果您已新增外部連線，則 Visual Studio 可以使用 CodeArtifact 儲存庫、其任何上游儲存庫或 NuGet.org 的套件。如需有關在 Visual Studio 中瀏覽和安裝 NuGet 套件的詳細資訊，請參閱 NuGet 文件中的使用 NuGet 套件管理員在 Visual Studio 中安裝和管理套件。

使用 Visual Studio Package Manager 主控台

Visual Studio Package Manager 主控台不會使用 CodeArtifact 登入資料提供者的 Visual Studio 版本。若要使用它，您必須設定命令列登入資料提供者。如需詳細資訊，請參閱「搭配 nuget 或 dotnet CLI 使用 CodeArtifact 」。

搭配 nuget 或 dotnet CLI 使用 CodeArtifact

您可以使用 `nuget`和 等 CLI 工具`dotnet`，從 CodeArtifact 發佈和使用套件。本文件提供有關設定 CLI 工具以及使用它們來發佈或取用套件的資訊。

主題

- 設定 nuget 或 dotnet CLI
- 從 CodeArtifact 取用 NuGet 套件
- 將 NuGet 套件發佈至 CodeArtifact
- CodeArtifact NuGet 登入資料提供者參考
- CodeArtifact NuGet 登入資料提供者版本

設定 nuget 或 dotnet CLI

您可以使用 CodeArtifact NuGet 登入資料提供者、使用 或手動設定 nuget AWS CLI或 dotnet CLI。強烈建議使用登入資料提供者設定 NuGet，以簡化設定並持續進行身分驗證。

方法 1：使用 CodeArtifact NuGet 登入資料提供者設定

CodeArtifact NuGet 登入資料提供者使用 NuGet CLI 工具簡化 CodeArtifact 的身分驗證和組態。CodeArtifact 身分驗證字符的有效期最長為 12 小時。為了避免在使用 nuget 或 dotnet CLI 時手動重新整理權杖，憑證提供者會在目前的權杖過期之前定期擷取新的權杖。

> **Important**
>
> 若要使用登入資料提供者，請確定任何現有的 AWS CodeArtifact 登入資料都已從您的`nuget.config`檔案清除，這些登入資料可能已手動新增，或執行 `aws codeartifact login`來設定 NuGet。

安裝和設定 CodeArtifact NuGet 登入資料提供者

dotnet

1. 使用下列`dotnet`命令，從 NuGet.org 下載最新版本的 AWS.CodeArtifact.NuGet.CredentialProvider 工具。

   ```
   dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
   ```

2. 使用 `codeartifact-creds install`命令將登入資料提供者複製到 NuGet 外掛程式資料夾。

   ```
   dotnet codeartifact-creds install
   ```

3. （選用）：設定您要與登入資料提供者搭配使用的 AWS 設定檔。如果未設定，登入資料提供者將使用預設設定檔。如需 AWS CLI 設定檔的詳細資訊，請參閱具名設定檔。

   ```
   dotnet codeartifact-creds configure set profile profile_name
   ```

nuget

執行下列步驟，使用 NuGet CLI 從 Amazon S3 儲存貯體安裝 CodeArtifact NuGet 登入資料提供者並進行設定。登入資料提供者將使用預設 AWS CLI 設定檔，如需設定檔的詳細資訊，請參閱具名設定檔。

1. 從 Amazon S3 儲存貯體下載最新版本的 CodeArtifact NuGet 登入資料提供者 (codeartifact-nuget-credentialprovider.zip)。

   若要檢視和下載舊版，請參閱 CodeArtifact NuGet 登入資料提供者版本。
2. 解壓縮檔案。
3. 在 Windows `%user_profile%/.nuget/plugins/netfx/`或 `~/.nuget/plugins/netfx` Linux 或 MacOS 上，將 AWS.CodeArtifact.NuGetCredentialProvider 資料夾從 netfx 資料夾複製到 。
4. 在 Windows `%user_profile%/.nuget/plugins/netcore/`或 `~/.nuget/plugins/netcore` Linux 或 MacOS 上，將 AWS.CodeArtifact.NuGetCredentialProvider 資料夾從 netcore 資料夾複製到 。

建立儲存庫並設定登入資料提供者之後，您可以使用 `nuget`或 `dotnet` CLI 工具來安裝和發佈套件。如需詳細資訊，請參閱 從 CodeArtifact 取用 NuGet 套件 和 將 NuGet 套件發佈至 CodeArtifact。

方法 2：使用登入命令設定 nuget 或 dotnet

中的 `codeartifact login`命令會將儲存庫端點和授權字符 AWS CLI 新增至您的 NuGet 組態檔案，讓 nuget 或 dotnet 連線到 CodeArtifact 儲存庫。這將修改位於 的使用者層級 NuGet 組態，`%appdata%\NuGet\NuGet.Config`適用於 Windows 和 `~/.config/NuGet/NuGet.Config` `~/.nuget/NuGet/NuGet.Config` Mac/Linux。如需 NuGet 組態的詳細資訊，請參閱常見 NuGet 組態。

使用 **`login`**命令設定 nuget 或 dotnet

1. 設定您的 AWS 登入資料以搭配 使用 AWS CLI，如 中所述CodeArtifact 入門。
2. 確定 NuGet CLI 工具 (`nuget` 或 `dotnet`) 已正確安裝和設定。如需說明，請參閱 nuget 或 dotnet 文件。
3. 使用 CodeArtifact `login`命令來擷取登入資料，以便與 NuGet 搭配使用。

> **Note**
>
> 如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

dotnet

> **Important**
>
> Linux 和 MacOS 使用者：由於非 Windows 平台不支援加密，因此您擷取的登入資料會以純文字形式儲存在組態檔案中。

```
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
```

登入命令將：

- 使用您的 AWS 登入資料從 CodeArtifact 擷取授權字符。
- 使用 NuGet 套件來源的新項目來更新您的使用者層級 NuGet 組態。指向 CodeArtifact 儲存庫端點的來源將稱為 *domain_name/repo_name*。

呼叫 後的預設授權期間`login`為 12 小時，`login`必須呼叫 以定期重新整理字符。如需使用 `login`命令建立的授權字符的詳細資訊，請參閱 使用 `login`命令建立的字符。

建立儲存庫並設定身分驗證之後`dotnet`，您可以使用 `nuget`、 或 `msbuild` CLI 用戶端來安裝和發佈套件。如需詳細資訊，請參閱 從 CodeArtifact 取用 NuGet 套件 和 將 NuGet 套件發佈至 CodeArtifact。

方法 3：在沒有登入命令的情況下設定 nuget 或 dotnet

對於手動組態，您必須將儲存庫端點和授權字符新增至 NuGet 組態檔案，以啟用 nuget 或 dotnet 來連線至 CodeArtifact 儲存庫。

手動設定 nuget 或 dotnet 以連接至 CodeArtifact 儲存庫。

1. 使用 `get-repository-endpoint` AWS CLI 命令判斷 CodeArtifact 儲存庫端點。

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --domain-
   owner 111122223333 --repository my_repo --format nuget
   ```

   輸出範例：

   ```
   {
      "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-
   west-2.amazonaws.com/nuget/my_repo/"
   }
   ```

2. 使用 `get-authorization-token` AWS CLI 命令，從套件管理員取得連接至儲存庫的授權字符。

   ```
   aws codeartifact get-authorization-token --domain my_domain
   ```

   輸出範例：

   ```
   {
      "authorizationToken": "eyJ2I...viOw",
      "expiration": 1601616533.0
   }
   ```

3. 透過附加`/v3/index.json`至步驟 3 `get-repository-endpoint`中 傳回的 URL 來建立完整的儲存庫端點 URL。
4. 設定 nuget 或 dotnet 以使用步驟 1 的儲存庫端點和步驟 2 的授權字符。

> **Note**
>
> 來源 URL 必須以 結尾`/v3/index.json`，讓 nuget 或 dotnet 成功連線至 CodeArtifact 儲存庫。

dotnet

Linux 和 MacOS 使用者：由於非 Windows 平台不支援加密，您必須將 `--store-password-in-clear-text`旗標新增至下列命令。請注意，這會將您的密碼以純文字形式存放在您的組態檔案中。

```
dotnet nuget add source https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/nuget/my_repo/v3/index.json --name packageSourceName --password eyJ2I...vi0w --username aws
```

> **Note**
>
> 若要更新現有的來源，請使用 `dotnet nuget update source`命令。

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
```

輸出範例：

```
Package source with Name: domain_name/repo_name added successfully.
```

> **Note**
>
> 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

從 CodeArtifact 取用 NuGet 套件

使用 CodeArtifact 設定 NuGet 後，您可以使用存放在 CodeArtifact 儲存庫或其其中一個上游儲存庫中的 NuGet 套件。

若要使用 CodeArtifact 儲存庫或其中一個上游儲存庫中的套件版本，請執行下列命令，將 *packageName* 取代為您想要使用的套件名稱nugetdotnet，並使用 NuGet 組態檔案中 CodeArtifact

儲存庫的來源名稱取代 *packageSourceName*。如果您使用 `login`命令來設定 NuGet 組態，來源名稱為 *domain_name/repo_name*。

> **Note**
>
> 請求套件時，NuGet 用戶端會快取該套件存在的版本。由於此行為，在所需版本可用之前，先前請求的套件安裝可能會失敗。若要避免此失敗並成功安裝存在的套件，您可以在安裝 之前使用 `nuget locals all --clear`或 清除 NuGet 快取`dotnet nuget locals all --clear`，或藉由提供 的 `-NoCache`選項`nuget`或 的 `--no-cache`選項，避免在 `install`和 `restore`命令期間使用快取`dotnet`。

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

安裝特定版本的套件

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

如需詳細資訊，請參閱 Microsoft 文件中的使用 nuget.exe CLI 管理套件或使用 dotnet CLI 安裝和管理套件。

從 NuGet.org 取用 NuGet 套件

您可以透過 CodeArtifact 儲存庫使用 NuGet.org 的 NuGet 套件，方法是使用 NuGet.org 的外部連線來設定儲存庫。 NuGet.org 從 NuGet.org 取用的套件會擷取並儲存在 CodeArtifact 儲存庫中。如需新增外部連線的詳細資訊，請參閱 將 CodeArtifact 儲存庫連線至公有儲存庫。

將 NuGet 套件發佈至 CodeArtifact

使用 CodeArtifact 設定 NuGet 後，您可以使用 nuget或 dotnet將套件版本發佈至 CodeArtifact 儲存庫。

若要將套件版本推送至 CodeArtifact 儲存庫，請在 NuGet 組態檔案中，使用.nupkg檔案的完整路徑和 CodeArtifact 儲存庫的來源名稱執行下列命令。如果您使用 login命令來設定 NuGet 組態，則來源名稱為 domain_name/repo_name。

> **Note**
>
> 如果您沒有要發佈的 NuGet 套件，您可以建立該套件。如需詳細資訊，請參閱 Microsoft 文件中的套件建立工作流程。

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 登入資料提供者參考

CodeArtifact NuGet 登入資料提供者可讓您輕鬆地使用 CodeArtifact 儲存庫設定和驗證 NuGet。

CodeArtifact NuGet 登入資料提供者命令

本節包含 CodeArtifact NuGet 登入資料提供者的命令清單。這些命令必須以 做為字首，dotnet codeartifact-creds如下列範例所示。

```
dotnet codeartifact-creds command
```

- `configure set profile profile`：設定登入資料提供者以使用提供的 AWS 設定檔。
- `configure unset profile`：如果已設定，則移除已設定的設定檔。

- `install`：將登入資料提供者複製到 `plugins` 資料夾。
- `install --profile profile`：將登入資料提供者複製到 `plugins` 資料夾，並將其設定為使用提供的 AWS 設定檔。
- `uninstall`：解除安裝登入資料提供者。這不會移除組態檔案的變更。
- `uninstall --delete-configuration`：解除安裝登入資料提供者，並移除組態檔案的所有變更。

CodeArtifact NuGet 登入資料提供者日誌

若要啟用 CodeArtifact NuGet 登入資料提供者的記錄，您必須在環境中設定日誌檔案。登入資料提供者日誌包含有用的偵錯資訊，例如：

- 用於進行連線的 AWS 設定檔
- 任何身分驗證錯誤
- 如果提供的端點不是 CodeArtifact URL

設定 CodeArtifact NuGet 登入資料提供者日誌檔案

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

設定日誌檔案後，任何`codeartifact-creds`命令都會將其日誌輸出附加至該檔案的內容。

CodeArtifact NuGet 登入資料提供者版本

下表包含 CodeArtifact NuGet 登入資料提供者的版本歷史記錄資訊和下載連結。

版本	變更	發佈日期	下載連結 (S3)
1.0.2 （最新）	升級的相依性	06/26/2024	下載 1.0.2 版
1.0.1	新增對 net5、net6 和 SSO 設定檔的支援	03/05/2022	下載 v1.0.1
1.0.0	初始 CodeArtifact NuGet 登入資料提供者版本	11/20/2020	下載 v1.0.0

NuGet 套件名稱、版本和資產名稱標準化

CodeArtifact 會先標準化套件和資產名稱，再存放套件版本，這表示 CodeArtifact 中的名稱或版本可能與發佈套件或資產時提供的名稱或版本不同。

套件名稱標準化：CodeArtifact 會將所有字母轉換為小寫，以標準化 NuGet 套件名稱。

套件版本標準化：CodeArtifact 使用與 NuGet 相同的模式標準化 NuGet 套件版本。以下資訊來自 NuGet 文件的標準化版本編號。

- 前導零會從版本編號中移除：
  - `1.00` 被視為 `1.0`
  - `1.01.1` 被視為 `1.1.1`
  - `1.00.0.1` 被視為 `1.0.0.1`
- 版本編號的第四部分會省略零：
  - `1.0.0.0` 被視為 `1.0.0`
  - `1.0.01.0` 被視為 `1.0.1`
- SemVer 2.0.0 建置中繼資料已移除：
  - `1.0.7+r3456` 被視為 `1.0.7`

套件資產名稱標準化：CodeArtifact 會從標準化套件名稱和套件版本建構 NuGet 套件資產名稱。

非標準化套件名稱和版本名稱可與 API 和 CLI 請求搭配使用，因為 CodeArtifact 會對這些請求的套件名稱和版本輸入執行標準化。例如， `--package Newtonsoft.JSON`和 的輸入`--version 12.0.03.0`會標準化，並傳回具有標準化套件名稱 `newtonsoft.json`和 版本的套件`12.0.3`。

您必須在 API 和 CLI 請求中使用標準化套件資產名稱，因為 CodeArtifact 不會在`--asset`輸入上執行標準化。

您必須在 ARNs 中使用標準化名稱和版本。

若要尋找套件的標準化名稱，請使用 `aws codeartifact list-packages`命令。如需詳細資訊，請參閱列出套件名稱。

若要尋找套件的非標準化名稱，請使用 `aws codeartifact describe-package-version`命令。套件的非標準化名稱會在 `displayName` 欄位中傳回。如需詳細資訊，請參閱檢視和更新套件版本詳細資訊和相依性。

NuGet 相容性

本指南包含 CodeArtifact 與不同 NuGet 工具和版本相容性的相關資訊。

主題

- 一般 NuGet 相容性
- NuGet 命令列支援

一般 NuGet 相容性

AWS CodeArtifact 支援 NuGet 4.8 和更新版本。

AWS CodeArtifact 僅支援 NuGet HTTP 通訊協定的 V3。這表示不支援某些依賴通訊協定 V2 的 CLI 命令。如需詳細資訊，請參閱 nuget.exe 命令支援 一節。

AWS CodeArtifact 不支援 PowerShellGet 2.x。

NuGet 命令列支援

AWS CodeArtifact 支援 NuGet (`nuget.exe`) 和 .NET Core (`dotnet`) CLI 工具。

nuget.exe 命令支援

由於 CodeArtifact 僅支援 NuGet HTTP 通訊協定的 V3，因此針對 CodeArtifact 資源使用時，下列命令將無法運作：

- `list`： `nuget list`命令會顯示來自指定來源的套件清單。若要取得 CodeArtifact 儲存庫中的套件清單，您可以從 CLI AWS 使用 列出套件名稱命令。

搭配 Python 使用 CodeArtifact

這些主題說明如何搭配 CodeArtifact 使用 pip、Python 套件管理員twine和 Python 套件發佈公用程式。

主題

- 搭配 CodeArtifact 設定和使用 pip
- 使用 CodeArtifact 設定和使用雙身
- Python 套件名稱標準化
- Python 相容性
- 從上游和外部連線請求 Python 套件

搭配 CodeArtifact 設定和使用 pip

pip 是 Python 套件的套件安裝程式。若要使用 pip 從 CodeArtifact 儲存庫安裝 Python 套件，您必須先使用 CodeArtifact 儲存庫資訊和登入資料來設定 pip 用戶端。

pip 只能用於安裝 Python 套件。若要發佈 Python 套件，您可以使用雙工。如需詳細資訊，請參閱使用 CodeArtifact 設定和使用雙身。

使用 `login`命令設定 pip

首先，設定您的 AWS 登入資料以搭配 使用 AWS CLI，如 中所述CodeArtifact 入門。然後，使用 CodeArtifact `login`命令來擷取登入資料並使用`pip`它們進行設定。

> **Note**
>
> 如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

若要設定 pip，請執行下列命令。

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

login 使用您的 AWS 登入資料從 CodeArtifact 擷取授權字符。login 命令會編輯 ~/.config/pip/pip.conf以index-url將 設定為 --repository選項指定的儲存庫，以設定 pip 搭配 CodeArtifact 使用。

呼叫 後的預設授權期間login為 12 小時，login必須呼叫 以定期重新整理字符。如需使用 login命令建立的授權字符的詳細資訊，請參閱 使用 login命令建立的字符。

設定不含登入命令的 pip

如果您無法使用 login命令來設定 pip，您可以使用 pip config。

1. 使用 AWS CLI 擷取新的授權字符。

   > **Note**
   >
   > 如果您要存取您擁有之網域中的儲存庫，則不需要包含 --domain-owner。如需詳細資訊，請參閱跨帳戶網域。

   ```
   CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

2. 使用 pip config 設定 CodeArtifact 登錄 URL 和登入資料。下列命令只會更新目前的環境組態檔案。若要更新整個系統的組態檔案，請將 取代site為 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**
   >
   > 若要使用雙堆疊端點，請使用 codeartifact.*region*.on.aws端點。

> **Important**
>
> 登錄 URL 必須以正斜線 (/) 結尾。否則，您無法連線至儲存庫。

範例 pip 組態檔案

以下是設定 CodeArtifact 登錄 URL 和登入資料後`pip.conf`的檔案範例。

```
[global]
index-url = https://aws:eyJ2ZX...@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/pypi/my_repo/simple/
```

執行 pip

若要執行`pip`命令，您必須`pip`使用 CodeArtifact 設定 。如需詳細資訊，請參閱下列文件。

1. 請遵循 使用 AWS CodeArtifact 設定區段中的步驟來設定 AWS 您的帳戶、工具和許可。
2. `twine` 按照中的步驟進行設定使用 CodeArtifact 設定和使用雙身。

假設您的儲存庫或其其中一個上游儲存庫中有套件，您可以使用 安裝套件`pip install`。例如，使用下列命令來安裝 `requests`套件。

```
pip install requests
```

使用 `-i`選項暫時還原為從 https://pypi.org 安裝套件，而不是從 CodeArtifact 儲存庫安裝套件。

```
pip install -i https://pypi.org/simple requests
```

使用 CodeArtifact 設定和使用雙身

Twine 是 Python 套件的套件發佈公用程式。若要使用孿生將 Python 套件發佈到您的 CodeArtifact 儲存庫，您必須先使用 CodeArtifact 儲存庫資訊和登入資料來設定孿生。

Twine 只能用於發佈 Python 套件。若要安裝 Python 套件，您可以使用 pip。如需詳細資訊，請參閱搭配 CodeArtifact 設定和使用 pip。

使用 **`login`**命令設定雙身

首先，設定您的 AWS 登入資料以搭配 使用 AWS CLI，如 中所述CodeArtifact 入門。然後，使用 CodeArtifact `login`命令來擷取登入資料，並與其設定雙身。

> **Note**
>
> 如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

若要設定雙身，請執行下列命令。

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

`login` 使用您的 AWS 登入資料從 CodeArtifact 擷取授權字符。`login` 命令會編輯 `~/.pypirc` 以使用 登入資料新增 `--repository`選項指定的儲存庫，藉此設定雙身以搭配 CodeArtifact 使用。

呼叫 後的預設授權期間`login`為 12 小時，`login`必須呼叫 以定期重新整理字符。如需使用 `login`命令建立的授權字符的詳細資訊，請參閱 使用 `login`命令建立的字符。

不使用 `login`命令設定雙身

如果您無法使用 `login`命令來設定雙身，您可以使用 `~/.pypirc` 檔案或環境變數。若要使用 檔案，請將下列項目新增至該`~/.pypirc`檔案。密碼必須是 `get-authorization-token` API 取得的身分驗證字符。

```
[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**
>
> 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

若要使用環境變數，請執行下列動作。

> **Note**
>
> 如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

```
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`
```

執行雙身

使用雙工發佈 Python 套件資產之前，您必須先設定 CodeArtifact 許可和資源。

1. 請遵循 使用 AWS CodeArtifact 設定區段中的步驟來設定 AWS 您的帳戶、工具和許可。
2. 遵循 使用 `login`命令設定雙身或 中的步驟來設定雙身不使用 `login`命令設定雙身。

設定雙身後，您可以執行`twine`命令。使用下列命令來發佈 Python 套件資產。

```
twine upload --repository codeartifact mypackage-1.0.tgz
```

如需有關如何建置和封裝 Python 應用程式的資訊，請參閱 Python 封裝授權機構網站上的產生分佈封存。

Python 套件名稱標準化

CodeArtifact 會先標準化套件名稱再存放，這表示 CodeArtifact 中的套件名稱可能與發佈套件時提供的名稱不同。

對於 Python 套件，執行標準化時，套件名稱會小寫`-`，且字元 `.`、 和 的所有執行個體`_`都會以單一`-`字元取代。因此，套件名稱 `pigeon_cli` 和 `pigeon.cli`會標準化並儲存為 `pigeon-cli`。非標準化名稱可由 pip 和 Twine 使用，但標準化名稱必須在 CodeArtifact CLI 或 API 請求 （ 例如 `list-package-versions`) 和 ARNs 中使用。如需 Python 套件名稱標準化的詳細資訊，請參閱 Python 文件中的 PEP 503。

Python 相容性

CodeArtifact 不支援 PyPI 的 `XML-RPC`或 `JSON` API。 APIs

CodeArtifact 支援 PyPI 的 `Legacy` API，但 `simple` API 除外。 APIs 雖然 CodeArtifact 不支援 `/simple/` API 端點，但它確實支援`/simple/<project>/`端點。

如需詳細資訊，請參閱 Python Packaging Authority 的 GitHub 儲存庫上的以下內容。

- XML-RPC API
- JSON API
- 舊版 API

pip 命令支援

以下各節摘要說明 CodeArtifact 儲存庫所支援的 pip 命令，以及不支援的特定命令。

主題

- 與儲存庫互動的支援命令
- 支援的用戶端命令

與儲存庫互動的支援命令

本節列出 `pip`命令，其中`pip`用戶端向所設定的登錄檔提出一或多個請求。已驗證這些命令在針對 CodeArtifact 儲存庫調用時可正常運作。

Command	描述
安裝	安裝套件。
下載	下載套件。

CodeArtifact 不會實作 `pip search`。如果您已`pip`設定 CodeArtifact 儲存庫，執行 `pip search`會搜尋並顯示來自 PyPI 的套件。

支援的用戶端命令

這些命令不需要與儲存庫進行任何直接互動，因此 CodeArtifact 不需要執行任何動作來支援儲存庫。

Command	描述
解除安裝	解除安裝套件。
凍結	以要求格式輸出已安裝的套件。
清單	列出已安裝的套件。
show	顯示已安裝套件的相關資訊。
檢查	確認已安裝的套件具有相容的相依性。
組態	管理本機和全域組態。
滾輪	根據您的需求建置車輪。
雜湊	套件封存的運算雜湊。
完成	協助完成命令。
debug	顯示適用於偵錯的資訊。
help	顯示 命令的說明。

從上游和外部連線請求 Python 套件

從 pypi.org 匯入 Python 套件版本時，CodeArtifact 會匯入該套件版本中的所有資產。雖然大多數 Python 套件都包含少量資產，但有些套件包含超過 100 個資產，通常支援多個硬體架構和 Python 解譯器。

對於現有的套件版本，新資產通常會發佈到 pypi.org。例如，有些專案會在發行新版本的 Python 時發佈新資產。從 CodeArtifact 安裝 Python 套件並搭配 時`pip install`，會更新 CodeArtifact 儲存庫中保留的套件版本，以反映來自 pypi.org：// 的最新資產集。

同樣地，如果上游 CodeArtifact 儲存庫中的套件版本沒有新的資產，則在`pip install`執行時，這些資產會保留在目前的 CodeArtifact 儲存庫中。

Yanked 套件版本

部分 pypi.org 中的套件版本標示為 yanked，它會與套件安裝程式（例如 pip) 通訊，除非版本是唯一符合版本指標（使用 ==或 ===) 的版本。如需詳細資訊，請參閱 PEP_592。

如果 CodeArtifact 中的套件版本最初是從外部連線至 pypi.org 擷取，當您從 CodeArtifact 儲存庫安裝套件版本時，CodeArtifact 會確保從 pypi.org 擷取套件版本的更新偏擺中繼資料。

如何知道套件版本是否已取消

若要檢查 CodeArtifact 中的套件版本是否遭到取消，您可以嘗試使用 安裝套件版本`pip install packageName===packageVersion`。如果套件版本已取消，您會收到類似以下的警告訊息：

```
WARNING: The candidate selected for download or install is a yanked version
```

若要檢查套件版本是否在 pypi.org：//www.microsoft.com/microsoft.com/microsoft.microsoft.com/microsoft.com/microsoft.microsoft.com/microsoft.microsoft.microsoft.com/microsoft.microsoft.microsoft.microsoft.microsoft.com/soft.microsoft.microsoft.micro；若要查看套件版本的 pypi.org`https://pypi.org/project/packageName/packageVersion/`.com/

在私有套件上設定偏擺狀態

CodeArtifact 不支援為直接發佈至 CodeArtifact 儲存庫的套件設定偏擺中繼資料。

為什麼 CodeArtifact 未擷取套件版本的最新偏擺中繼資料或資產？

通常，CodeArtifact 可確保從 CodeArtifact 儲存庫擷取 Python 套件版本時，偏擺的中繼資料會up-to-date，且具有 pypi.org：//www. 上的最新值。此外，套件版本中的資產清單也會以 https：// pypi.org 和任何上游 CodeArtifact 儲存庫上的最新設定來保持最新狀態。無論您是第一次安裝套件版本，而且 CodeArtifact 從 pypi.org 將其匯入 CodeArtifact 儲存庫，或是先前已安裝套件，都是如此。不過，在某些情況下，套件管理員用戶端，例如 pip，不會從 pypi.org 或上游儲存庫提取最新的偏擺中繼資料。反之，CodeArtifact 會傳回已存放在儲存庫中的資料。本節說明發生這種情況的三種方式：

上游組態：如果使用 disassociate-external-connection 從儲存庫或其上游移除與 pypi.org 的外部連線，則不會再從 pypi.org 重新整理偏擺的中繼資料。同樣地，如果您移除上游儲存庫，來自已移除儲存庫和已移除儲存庫上游的資產將無法再供目前的儲存庫使用。如果您使用 CodeArtifact 套件原始伺服器控制來防止提取特定套件的新版本，則同樣情況是如此 — 設定`upstream=BLOCK`會封鎖重新整理偏擺的中繼資料。

套件版本狀態：如果您將套件版本的狀態設定為 Published或 以外的任何項目Unlisted，將不會重新整理套件版本的偏擺中繼資料和資產。同樣地，如果您要擷取特定套件版本 （例如 torch 2.0.1)，且上游儲存庫中存在狀態不是 Published或 的相同套件版本Unlisted，這也會封鎖上游儲存庫到目前儲存庫的偏擺中繼資料和資產傳播。這是因為其他套件版本狀態表示版本不再在任何儲存庫中使用。

直接發佈：如果您將特定套件版本直接發佈至 CodeArtifact 儲存庫，這將防止套件版本的受阻中繼資料和資產重新整理從其上游儲存庫和 pypi.org 進行。例如，假設您從套件版本 下載資產torch 2.0.1，例如 torch-2.0.1-cp311-none-macosx_11_0_arm64.whl，使用 Web 瀏覽器，然後使用 Twine 做為 將其發佈到您的 CodeArtifact 儲存庫torch 2.0.1。CodeArtifact 會透過直接發佈至您的儲存庫來追蹤套件版本是否進入網域，而不是從與 pypi.org 的外部連線或上游儲存庫。在此情況下，CodeArtifact 不會讓偏擺中繼資料與上游儲存庫或 pypi.org 保持同步。如果您發佈torch 2.0.1到上游儲存庫，也是如此 — 套件版本的存在會阻止將偏擺中繼資料和資產傳播到更深入上游圖形的儲存庫。

搭配 Ruby 使用 CodeArtifact

這些主題說明如何搭配 CodeArtifact 使用 RubyGems 和 Bundler 工具來安裝和發佈 Ruby Gem。

> **Note**
>
> CodeArtifact 建議使用 Ruby 3.3 或更新版本，且不適用於 Ruby 2.6 或更新版本。

主題

- 透過 CodeArtifact 設定和使用 RubyGems 和 Bundler
- RubyGems 命令支援
- Bundler 相容性

透過 CodeArtifact 設定和使用 RubyGems 和 Bundler

在 CodeArtifact 中建立儲存庫後，您可以使用 RubyGems (`gem`) 和 Bundler (`bundle`) 來安裝和發佈 Gem。本主題說明如何設定套件管理員以驗證和使用 CodeArtifact 儲存庫。

使用 CodeArtifact 設定 RubyGems (**`gem`**) 和 Bundler (**`bundle`**)

若要使用 RubyGems (`gem`) 或 Bundler (`bundle`) 將 Gem 發佈至 AWS CodeArtifact 或取用 Gem，您必須先使用 CodeArtifact 儲存庫資訊來設定 Gem 套件，包括存取它的登入資料。請依照下列其中一個程序的步驟，使用 CodeArtifact 儲存庫端點資訊和登入資料來設定 和 `gem` `bundle` CLI 工具。

使用主控台指示設定 RubyGems 和 Bundler

您可以使用 主控台中的組態指示，將 Ruby 套件管理員連線至 CodeArtifact 儲存庫。主控台指示提供自訂命令，您可以執行這些命令來設定套件管理員，而不需要尋找和填寫 CodeArtifact 資訊。

1. 開啟位於 https://console.aws.amazon.com/codesuite/codeartifact/home 的 AWS CodeArtifact 主控台。
2. 在導覽窗格中，選擇儲存庫，然後選擇您要用於安裝或推送 Ruby Gem 套件的儲存庫。
3. 選擇檢視連線指示。
4. 選擇您的作業系統。

5. 選擇您要使用 CodeArtifact 儲存庫設定的 Ruby 套件管理員用戶端。
6. 遵循產生的說明，設定套件管理員用戶端從儲存庫安裝 Ruby Gem 套件或將 Ruby Gem 套件發佈到儲存庫。

手動設定 RubyGems 和 Bundler

如果您無法或不想使用主控台的組態指示，您可以使用下列指示，手動將 Ruby 套件管理員連線至 CodeArtifact 儲存庫。

1. 在命令列中，使用下列命令來擷取 CodeArtifact 授權字符，並將其存放在 環境變數中。

   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

   macOS and Linux

   ```
   export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

   Windows

   - Windows （使用預設命令 shell)：

     ```
     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. 若要將 Ruby Gem 套件發佈到您的儲存庫，請使用下列命令來擷取 CodeArtifact 儲存庫的端點，並將其儲存在RUBYGEMS_HOST環境變數中。gem CLI 使用此環境變數來判斷 Gem 的發佈位置。

> **Note**
>
> 或者，您可以使用 `gem push`命令為儲存庫端點提供 `--host`選項，而不是使用 `RUBYGEMS_HOST`環境變數。

- 將 *my_domain* 取代為您的 CodeArtifact 網域名稱。
- 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
- 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

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

下列命令會擷取儲存庫端點、修剪結尾 `/`，然後將它們存放在 環境變數中。

- Windows （使用預設命令 shell)：

```
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("/")
```

下列 URL 是範例儲存庫端點：

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/
```

> **Note**
>
> 若要使用雙堆疊端點，請使用 codeartifact.*region*.on.aws端點。

3. 若要將 Ruby Gem 套件發佈至您的儲存庫，您必須編輯 ~/.gem/credentials 檔案以包含您的身分驗證字符，以使用 RubyGems 驗證 CodeArtifact。如果~/.gem/目錄或~/.gem/credentials檔案不存在，請建立目錄和檔案。

   macOS and Linux

   ```
   echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/
   credentials
   ```

   Windows

   - Windows （ 使用預設命令 shell)：

     ```
     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. 若要使用 從儲存庫gem安裝 Ruby Gem，您必須將儲存庫端點資訊和身分驗證字符新增至 .gemrc 檔案。您可以將其新增至全域檔案 (~/.gemrc) 或專案.gemrc檔案。您必須新增至 的 CodeArtifact 資訊.gemrc是儲存庫端點和身分驗證字符的組合。其格式如下：

   ```
   https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
   west-2.amazonaws.com/ruby/my_repo/
   ```

   - 對於身分驗證字符，您可以使用先前步驟中設定CODEARTIFACT_AUTH_TOKEN的環境變數。
   - 若要擷取儲存庫端點，您可以讀取先前設定RUBYGEMS_HOST的環境變數值，也可以使用下列get-repository-endpoint命令，視需要取代這些值：

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint
 --output text
```

在您擁有端點之後，請使用文字編輯器將 aws:${CODEARTIFACT_AUTH_TOKEN}@ 新增至適當的位置。建立儲存庫端點和身分驗證字符字串後，請使用 echo命令將其新增至 .gemrc 檔案的 :sources:區段，如下所示：

> **Warning**
>
> CodeArtifact 不支援使用 gem sources -add命令將儲存庫新增為來源。您必須直接將來源新增至 檔案。

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 （ 使用預設命令 shell)：

```
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. 若要使用 Bundler，您必須執行下列`bundle config`命令，以儲存庫端點 URL 和身分驗證字符來設定 Bundler：

   macOS and Linux

   ```
   bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
   ```

   Windows

   - Windows （使用預設命令 shell)：

     ```
     bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
     ```

   - Windows PowerShell：

     ```
     bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
     ```

現在您已使用 CodeArtifact 儲存庫設定 RubyGems (`gem`) 和 Bundler (`bundle`)，您可以使用它們在其中發佈和使用 Ruby Gem。

從 CodeArtifact 安裝 Ruby Gem

使用下列程序，透過 `gem`或 CLI 工具從 CodeArtifact `bundle` 儲存庫安裝 Ruby Gem。

使用 安裝 Ruby Gem 套件 **`gem`**

您可以使用 RubyGems (`gem`) CLI，從 CodeArtifact 儲存庫快速安裝特定版本的 Ruby Gem。

使用 從 CodeArtifact 儲存庫安裝 Ruby Gem **`gem`**

1. 如果您尚未執行，請依照中的步驟使用 CodeArtifact 設定 RubyGems (`gem`) 和 Bundler (`bundle`)設定 `gem` CLI，以使用具有適當登入資料的 CodeArtifact 儲存庫。

   > **Note**
   >
   > 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 使用以下命令從 CodeArtifact 安裝 Ruby Gem：

```
gem install my_ruby_gem --version 1.0.0
```

使用 安裝 Ruby Gem 套件 **bundle**

您可以使用 Bundler (bundle) CLI 安裝在 中設定的 Ruby GemGemfile。

使用 從 CodeArtifact 儲存庫安裝 Ruby Gem **bundle**

1. 如果您尚未執行，請依照 中的步驟使用 CodeArtifact 設定 RubyGems (gem) 和 Bundler (bundle)設定 bundle CLI，以使用具有適當登入資料的 CodeArtifact 儲存庫。

   > **Note**
   >
   > 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 將 CodeArtifact 儲存庫端點 URL 新增至您的 Gemfile，source以從 CodeArtifact 儲存庫及其上游安裝設定的 Ruby Gem。

   ```
   source "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   ruby/my_repo/"

   gem 'my_ruby_gem'
   ```

3. 使用下列命令來安裝 Ruby Gem 套件，如 中所指定Gemfile：

   ```
   bundle install
   ```

將 Ruby Gem 套件發佈至 CodeArtifact

使用下列程序，使用 CLI 將 Ruby Gem 套件發佈至 CodeArtifact gem 儲存庫。

1. 如果您尚未執行，請依照中的步驟使用 CodeArtifact 設定 RubyGems (gem) 和 Bundler (bundle)設定 gem CLI，以使用具有適當登入資料的 CodeArtifact 儲存庫。

> **Note**
>
> 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 使用下列命令將 Ruby Gem 套件發佈至 CodeArtifact 儲存庫。請注意，如果您未設定RUBYGEMS_HOST環境變數，則必須在 `--host`選項中提供 CodeArtifact 儲存庫端點。

```
gem push --key codeartifact_api_key my_ruby_gem-0.0.1.gem
```

RubyGems 命令支援

CodeArtifact 支援 `gem install`和 `gem push`命令。CodeArtifact 不支援下列gem命令：

- `gem fetch`
- `gem info --remote`
- `gem list --remote`
- `gem mirror`
- `gem outdated`
- `gem owner`
- `gem query`
- `gem search`
- `gem signin`
- `gem signout`
- `gem sources --add`
- `gem sources --update`
- `gem specification --remote`
- `gem update`
- `gem yank`

Bundler 相容性

本指南包含 CodeArtifact 與 Bundler 相容性的相關資訊。

Bundler 相容性

AWS CodeArtifact 建議 Bundler 的 2.4.11 或更高版本。如果您在安裝時遇到問題，請將 Bundler CLI 更新至最新版本。

Bundler 版本支援

在低於 2.4.11 的 Bundler 版本中，在 Bundler 決定查詢完整索引 之前，Gemfile 中可以定義的相依性上限為 500 個`specs.4.8.gz`。由於 CodeArtifact 不支援完整索引，因此在使用低於 2.4.11 的 Bundler 版本時，指定超過 500 個相依性將無法與 CodeArtifact 搭配使用。

若要使用 CodeArtifact 在您的 Gemfile 中定義超過 500 個相依性，請將 Bundler 更新至版本 2.4.11 或更高版本。

Bundler 操作支援

CodeArtifact 對 RubyGems 的支援不包含 Bundler Compact Index APIs（不支援 `/versions` API)。CodeArtifact 僅支援相依性 API。

此外，CodeArtifact 不支援各種規格 APIs，例如 `specs.4.8.gz`。

搭配 Swift 使用 CodeArtifact

這些主題說明如何使用 Swift Package Manager 搭配 CodeArtifact 來安裝和發佈 Swift 套件。

> **Note**
>
> CodeArtifact 支援 Swift 5.8 和更新版本，以及 Xcode 14.3 和更新版本。
> CodeArtifact 建議使用 Swift 5.9 和更新版本，以及 Xcode 15 和更新版本。

主題

- 使用 CodeArtifact 設定 Swift Package Manager
- 使用和發佈 Swift 套件
- Swift 套件名稱和命名空間標準化
- Swift 疑難排解

使用 CodeArtifact 設定 Swift Package Manager

若要使用 Swift Package Manager 將套件發佈至 AWS CodeArtifact 或使用套件，您必須先設定登入資料以存取 CodeArtifact 儲存庫。使用 CodeArtifact 登入資料和儲存庫端點設定 Swift Package Manager CLI 的建議方法是使用 `aws codeartifact login`命令。您也可以手動設定 Swift Package Manager。

使用登入命令設定 Swift

使用 `aws codeartifact login`命令來設定具有 CodeArtifact 的 Swift Package Manager。

> **Note**
>
> 若要使用登入命令，需要 Swift 5.8 或更新版本，建議使用 Swift 5.9 或更新版本。

`aws codeartifact login` 命令將執行下列動作：

1. 從 CodeArtifact 擷取身分驗證字符，並將其存放在您的環境中。憑證的儲存方式取決於環境的作業系統：
   a. macOS：在 macOS Keychain 應用程式中建立項目。

b. Linux 和 Windows：在 ~/.netrc 檔案中建立項目。

在所有作業系統中，如果存在登入資料項目，此命令會以新的字符取代該項目。

2. 擷取 CodeArtifact 儲存庫端點 URL，並將其新增至 Swift 組態檔案。命令會將儲存庫端點 URL 新增至位於 的專案層級組態檔案/path/to/project/.swiftpm/configuration/registries.json。

> **Note**
>
> aws codeartifact login 命令會呼叫必須從包含 Package.swift 檔案的目錄執行的swift package-registry命令。因此，aws codeartifact login命令必須在 Swift 專案內執行。

使用登入命令設定 Swift

1. 導覽至包含專案Package.swift檔案的 Swift 專案目錄。
2. 執行下列 aws codeartifact login 命令。

   如果您要存取您擁有之網域中的儲存庫，則不需要包含 --domain-owner。如需詳細資訊，請參閱跨帳戶網域。

```
aws codeartifact login --tool swift --domain my_domain \
--domain-owner 111122223333 --repository my_repo \
[--namespace my_namespace]
```

--namespace 選項會將應用程式設定為只在套件位於指定的命名空間中時，才會使用 CodeArtifact 儲存庫中的套件。CodeArtifact 命名空間與範圍同義，用於將程式碼組織到邏輯群組，並防止在程式碼庫包含多個程式庫時可能發生的名稱衝突。

呼叫 後的預設授權期間login為 12 小時，login必須呼叫 以定期重新整理字符。如需使用 login命令建立的授權字符的詳細資訊，請參閱 使用 login命令建立的字符。

不使用登入命令設定 Swift

雖然建議您使用 aws codeartifact login命令設定 Swift，但您也可以手動更新 Swift Package Manager 組態，在不使用登入命令的情況下設定 Swift Package Manager。

在下列程序中，您將使用 AWS CLI 執行下列動作：

1. 從 CodeArtifact 擷取身分驗證字符，並將其存放在您的環境中。憑證的儲存方式取決於環境的作業系統：

   a. macOS：在 macOS Keychain 應用程式中建立項目。

   b. Linux 和 Windows：在 `~/.netrc` 檔案中建立項目。

2. 擷取 CodeArtifact 儲存庫端點 URL。

3. 在`~/.swiftpm/configuration/registries.json`組態檔案中，使用儲存庫端點 URL 和身分驗證類型新增項目。

設定 Swift 而不使用登入命令

1. 在命令列中，使用下列命令來擷取 CodeArtifact 授權字符，並將其存放在 環境變數中。

   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

   macOS and Linux

   ```
   export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
   domain my_domain --domain-owner 111122223333 --query authorizationToken --output
    text`
   ```

   Windows

   - Windows （使用預設命令 shell)：

     ```
     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. 執行下列命令，取得 CodeArtifact 儲存庫的端點。您的儲存庫端點用來將 Swift Package Manager 指向您的儲存庫，以取用或發佈套件。

   - 以您的 CodeArtifact 網域名稱取代 *my_domain*。
   - 以網域擁有者 AWS 的帳戶 ID 取代 *111122223333*。如果您要存取您擁有之網域中的儲存庫，則不需要包含 `--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 將 *my_repo* 取代為您的 CodeArtifact 儲存庫名稱。

   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 （使用預設命令 shell)：

     ```
     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
     ```

   下列 URL 是範例儲存庫端點。

   ```
   https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   swift/my_repo/
   ```

   > **Note**
   >
   > 若要使用雙堆疊端點，請使用 `codeartifact.region.on.aws`端點。

> **Important**
>
> 用於設定 Swift Package Manager 時，您必須附加`login`到儲存庫 URL 端點的結尾。此程序的 命令會為您完成此操作。

3. 這兩個值存放在環境變數中時，請使用 `swift package-registry login`命令將其傳遞給 Swift，如下所示：

   macOS and Linux

   ```
   swift package-registry login ${CODEARTIFACT_REPO}login --token
    ${CODEARTIFACT_AUTH_TOKEN}
   ```

   Windows

   - Windows （使用預設命令 shell)：

     ```
     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. 接下來，更新應用程式所使用的套件登錄檔，以便從 CodeArtifact 儲存庫提取任何相依性。此命令必須在您嘗試解析套件相依性的專案目錄中執行：

   macOS and Linux

   ```
   $ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
   ```

   Windows

   - Windows （使用預設命令 shell)：

     ```
     $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
     ```

   - Windows PowerShell：

```
$ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
```

`--scope` 選項會將應用程式設定為僅在 CodeArtifact 儲存庫位於指定範圍內時，才會取用其套件。範圍與 CodeArtifact 命名空間同義，用於將程式碼組織到邏輯群組，並防止在程式碼庫包含多個程式庫時可能發生的名稱衝突。

5. 您可以在專案目錄中執行下列命令，檢視專案層級`.swiftpm/configuration/registries.json`檔案的內容，以確認組態已正確設定：

```
$ 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
}
```

現在您已使用 CodeArtifact 儲存庫設定 Swift Package Manager，您可以使用它來在其中發佈和使用 Swift 套件。如需詳細資訊，請參閱使用和發佈 Swift 套件。

使用和發佈 Swift 套件

從 CodeArtifact 取用 Swift 套件

使用下列程序從 AWS CodeArtifact 儲存庫使用 Swift 套件。

從 CodeArtifact 儲存庫使用 Swift 套件

1. 如果您尚未執行，請依照中的步驟使用 CodeArtifact 設定 Swift Package Manager設定 Swift Package Manager，以使用具有適當登入資料的 CodeArtifact 儲存庫。

> **Note**
>
> 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 編輯應用程式專案資料夾中Package.swift的檔案，以更新專案要使用的套件相依性。

   a. 如果Package.swift檔案不包含dependencies區段，請新增區段。

   b. 在 Package.swift 檔案的 dependencies區段中，新增您想要使用的套件，方法是新增其套件識別符。套件識別符包含以句號分隔的範圍和套件名稱。如需範例，請參閱後續步驟之後的程式碼片段。

      > **Tip**
      >
      > 若要尋找套件識別符，您可以使用 CodeArtifact 主控台。尋找您要使用的特定套件版本，並參考套件版本頁面上的安裝捷徑指示。

   c. 如果Package.swift檔案不包含targets區段，請新增區段。

   d. 在 targets區段中，新增使用相依性所需的目標。

      下列程式碼片段是範例程式碼片段，顯示Package.swift檔案中已設定 dependencies和targets區段：

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
    ],
...
```

3. 現在已設定所有項目，請使用下列命令從 CodeArtifact 下載套件相依性。

```
swift package resolve
```

在 Xcode 中使用 CodeArtifact 的 Swift 套件

使用下列程序從 Xcode 中的 CodeArtifact 儲存庫使用 Swift 套件。

從 Xcode 中的 CodeArtifact 儲存庫使用 Swift 套件

1. 如果您尚未執行，請依照中的步驟使用 CodeArtifact 設定 Swift Package Manager設定 Swift Package Manager，以使用具有適當登入資料的 CodeArtifact 儲存庫。

   > **Note**
   >
   > 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 在 Xcode 中將套件新增為專案中的相依性。

   a. 選擇檔案 > 新增套件。

   b. 使用搜尋列搜尋您的套件。您的搜尋格式必須為 `package_scope.package_name`。

   c. 找到後，選擇套件，然後選擇新增套件。

   d. 驗證套件後，請選擇您要新增做為相依性的套件產品，然後選擇新增套件。

如果您使用 CodeArtifact 儲存庫搭配 Xcode 時遇到問題，請參閱 Swift 疑難排解 以取得常見問題和可能的修正。

將 Swift 套件發佈至 CodeArtifact

CodeArtifact 建議使用 Swift 5.9 或更新版本，並使用 `swift package-registry publish`命令來發佈 Swift 套件。如果您使用的是舊版，則必須使用 Curl 命令將 Swift 套件發佈至 CodeArtifact。

使用 **`swift package-registry publish`**命令發佈 CodeArtifact 套件

使用下列程序搭配 Swift 5.9 或更新版本，將 Swift 套件發佈至具有 Swift Package Manager 的 CodeArtifact 儲存庫。

1. 如果您尚未執行，請依照 中的步驟使用 CodeArtifact 設定 Swift Package Manager設定 Swift Package Manager，以使用具有適當登入資料的 CodeArtifact 儲存庫。

> **Note**
>
> 產生的授權字符有效期為 12 小時。如果自建立後已經過了 12 小時，您將需要建立新的 。

2. 導覽至包含套件`Package.swift`檔案的 Swift 專案目錄。
3. 執行下列`swift package-registry publish`命令來發佈套件。命令會建立套件來源封存，並將其發佈到您的 CodeArtifact 儲存庫。

```
swift package-registry publish packageScope.packageName packageVersion
```

例如：

```
swift package-registry publish myScope.myPackage 1.0.0
```

4. 您可以在 主控台中檢查或使用 `aws codeartifact list-packages`命令，確認套件已發佈並存在於儲存庫中，如下所示：

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

您可以使用 `aws codeartifact list-package-versions`命令列出單一版本的套件，如下所示：

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

使用 Curl 發佈 CodeArtifact 套件

雖然建議使用 Swift 5.9 或更新版本隨附的 `swift package-registry publish`命令，但您也可以使用 Curl 將 Swift 套件發佈至 CodeArtifact。

使用下列程序，將 Swift 套件發佈至具有 Curl 的 AWS CodeArtifact 儲存庫。

1. 如果您尚未建立，請依照中的步驟建立和更新 `CODEARTIFACT_AUTH_TOKEN`和 `CODEARTIFACT_REPO`環境變數使用 CodeArtifact 設定 Swift Package Manager。

> **Note**
>
> 身分驗證字符的有效期為 12 小時。如果CODEARTIFACT_AUTH_TOKEN環境變數在建立後已經過 12 小時，您將需要使用新的登入資料重新整理環境變數。

2. 首先，如果您尚未建立 Swift 套件，您可以執行下列命令來執行此操作：

```
mkdir testDir && cd testDir
swift package init
git init .
swift package archive-source
```

3. 使用下列 Curl 命令將 Swift 套件發佈至 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. 您可以在 主控台中檢查或使用 `aws codeartifact list-packages`命令，確認套件已發佈並存在於儲存庫中，如下所示：

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

   您可以使用 `aws codeartifact list-package-versions`命令列出單一版本的套件，如下所示：

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

從 GitHub 擷取 Swift 套件並重新發佈至 CodeArtifact

使用下列程序從 GitHub 擷取 Swift 套件，並將其重新發佈至 CodeArtifact 儲存庫。

從 GitHub 擷取 Swift 套件並重新發佈至 CodeArtifact

1. 如果您尚未執行，請依照 中的步驟使用 CodeArtifact 設定 Swift Package Manager設定 Swift Package Manager，以使用具有適當登入資料的 CodeArtifact 儲存庫。

   > **Note**
   >
   > 產生的授權字符有效期為 12 小時。如果自建立權杖後已經過 12 小時，您將需要建立新的權杖。

2. 使用以下`git clone`命令複製您要擷取並重新發佈的 Swift 套件的 git 儲存庫。如需有關複製 GitHub 儲存庫的資訊，請參閱 GitHub 文件中的複製儲存庫。

   ```
   git clone repoURL
   ```

3. 導覽至您剛複製的儲存庫：

   ```
   cd repoName
   ```

4. 建立套件並將其發佈至 CodeArtifact。

   a. 建議：如果您使用的是 Swift 5.9 或更新版本，您可以使用下列`swift package-registry publish`命令來建立套件，並將其發佈至您設定的 CodeArtifact 儲存庫。

      ```
      swift package-registry publish packageScope.packageName versionNumber
      ```

      例如：

      ```
      swift package-registry publish myScope.myPackage 1.0.0
      ```

   b. 如果您使用的 Swift 版本早於 5.9，則必須使用 `swift archive-source`命令來建立套件，然後使用 Curl 命令來發佈套件。

      i. 如果您尚未設定 `CODEARTIFACT_AUTH_TOKEN`和 `CODEARTIFACT_REPO`環境變數，或已超過 12 小時，請遵循中的步驟不使用登入命令設定 Swift。

      ii. 使用 `swift package archive-source`命令建立 Swift 套件：

```
swift package archive-source
```

如果成功，您將在命令列Created *package_name*.zip中看到 。

iii. 使用下列 Curl 命令將 Swift 套件發佈至 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. 您可以在 主控台中檢查或使用 `aws codeartifact list-packages`命令，確認套件已發佈並存在於儲存庫中，如下所示：

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

您可以使用 `aws codeartifact list-package-versions`命令列出單一版本的套件，如下所示：

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Swift 套件名稱和命名空間標準化

CodeArtifact 會先標準化套件名稱和命名空間，再儲存它們，這表示 CodeArtifact 中的名稱可能與發佈套件時提供的名稱不同。

套件名稱和命名空間標準化：CodeArtifact 會將所有字母轉換為小寫，以標準化 Swift 套件名稱和命名空間。

套件版本標準化：CodeArtifact 不會標準化 Swift 套件版本。請注意，CodeArtifact 僅支援語意版本控制 2.0 版模式，如需語意版本控制的詳細資訊，請參閱語意版本控制 2.0.0。

非標準化套件名稱和命名空間可與 API 和 CLI 請求搭配使用，因為 CodeArtifact 會對這些請求的輸入執行標準化。例如，`--package myPackage`和 的輸入`--namespace myScope`會標準化，並傳回具有標準化套件名稱 `mypackage`和 命名空間的套件`myscope`。

您必須在 ARNs 中使用標準化名稱，例如在 IAM 政策中。

若要尋找套件的標準化名稱，請使用 `aws codeartifact list-packages`命令。如需詳細資訊，請參閱列出套件名稱。

Swift 疑難排解

以下資訊可協助您對 Swift 和 CodeArtifact 的常見問題進行疑難排解。

即使設定 Swift Package Manager 之後，我還是在 Xcode 中收到 401 錯誤

問題：當您嘗試從 CodeArtifact 儲存庫新增套件做為 Xcode 中 Swift 專案的相依性時，即使已遵循將 Swift 連線至 CodeArtifact 的指示，也會收到 401 未經授權的錯誤。

可能的修正：這可能是由 macOS Keychain 應用程式的問題所造成，而您的 CodeArtifact 登入資料會存放在該應用程式。若要修正此問題，建議您開啟 Keychain 應用程式並刪除所有 CodeArtifact 項目，並依照 中的指示，再次使用 CodeArtifact 儲存庫設定 Swift Package Manager使用 CodeArtifact 設定 Swift Package Manager。

由於密碼的金鑰鏈提示，Xcode 掛在 CI 機器上

問題：當您嘗試從 CodeArtifact 提取 Swift 套件作為連續整合 (CI) 伺服器上 Xcode 組建的一部分時，例如使用 GitHub 動作時，CodeArtifact 的身分驗證可能會停止，最終失敗，並顯示類似以下的錯誤訊息：

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

可能的修正：這是因為憑證未儲存至 CI 機器上的金鑰鏈，且 Xcode 僅支援儲存在金鑰鏈中的憑證。若要修正此問題，建議您使用下列步驟手動建立金鑰鏈項目：

1. 準備金鑰鏈。

```
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. 取得 CodeArtifact 身分驗證字符和您的儲存庫端點。

```
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. 手動建立 Keychain 項目。

```
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}"
```

如需此錯誤和解決方案的詳細資訊，請參閱 https://github.com/apple/swift-package-manager/issues/7236。

搭配一般套件使用 CodeArtifact

這些主題說明如何使用 AWS CodeArtifact 來使用和發佈一般套件。

主題

- 一般套件概觀
- 一般套件支援的命令
- 發佈和使用一般套件

一般套件概觀

您可以使用`generic`套件格式上傳任何類型的檔案，以在 CodeArtifact 儲存庫中建立套件。一般套件不會與任何特定的程式設計語言、檔案類型或套件管理生態系統相關聯。這對於儲存和版本控制任意建置成品非常有用，例如應用程式安裝程式、機器學習模型、組態檔案等。

一般套件包含套件名稱、命名空間、版本，以及一或多個資產（或檔案）。一般套件可與單一 CodeArtifact 儲存庫中的其他格式套件一起存在。

您可以使用 AWS CLI 或 SDK 來使用一般套件。如需使用一般套件的完整 AWS CLI 命令清單，請參閱一般套件支援的命令。

一般套件限制

- 永遠不會從上游儲存庫擷取它們。它們只能從發佈到其中的儲存庫取得。
- 他們無法宣告要從 ListPackageVersionDependencies 傳回或顯示在 中的相依性 AWS Management Console 。
- 他們可以存放 README 和 LICENSE 檔案，但 CodeArtifact 無法解譯這些檔案。這些檔案中的資訊不會從 GetPackageVersionReadme 或 DescribePackageVersion 傳回，也不會出現在 中 AWS Management Console。
- 如同 CodeArtifact 中的所有套件，資產大小和每個套件的資產數量都有限制。如需 CodeArtifact 中限制和配額的詳細資訊，請參閱 AWS CodeArtifact 中的配額。
- 其中包含的資產名稱必須遵循下列規則：
  - 資產名稱可以使用 Unicode 字母和數字。具體而言，允許這些 Unicode 字元類別：小寫字母 (`Ll`)、修飾詞字母 (`Lm`)、其他字母 (`Lo`)、大寫字母 (`Lt`)、大寫字母 (`Lu`)、字母編號 (`Nl`) 和小數位數 ()`Nd`。

- 允許使用以下特殊字元：~!@^&()-_+[]{};,.
- 資產無法命名為 .或 ..
- 空格是唯一允許的空格字元。資產名稱不能以空格字元開頭或結尾，或包含連續空格。

一般套件支援的命令

您可以使用 AWS CLI 或 SDK 來使用一般套件。下列 CodeArtifact 命令適用於一般套件：

- copy-package-versions （請參閱 在儲存庫之間複製套件）
- delete-package （請參閱 刪除套件 (AWS CLI)）
- delete-package-versions （請參閱 刪除套件版本 (AWS CLI)）
- describe-package
- describe-package-version （請參閱 檢視和更新套件版本詳細資訊和相依性）
- dispose-package-versions （請參閱 處置套件版本）
- get-package-version-asset （請參閱 下載套件版本資產）
- list-package-version-assets （請參閱 列出套件版本資產）
- list-package-versions （請參閱 列出套件版本）
- list-packages （請參閱 列出套件名稱）
- publish-package-version （請參閱 發佈一般套件）
- put-package-origin-configuration （請參閱 編輯套件原始伺服器控制項）

> **Note**
>
> 您可以使用publish原始伺服器控制設定來允許或封鎖在儲存庫中發佈一般套件名稱。不過，upstream設定不適用於一般套件，因為無法從上游儲存庫擷取這些套件。

- update-package-versions-status （請參閱 更新套件版本狀態）

發佈和使用一般套件

若要發佈一般套件版本及其相關資產，請使用 publish-package-version命令。您可以使用 list-package-version-asset命令列出一般套件的資產，並使用 下載。 get-package-version-asset下列主題包含step-by-step指示，以使用這些命令發佈一般套件或下載一般套件資產。

發佈一般套件

一般套件包含套件名稱、命名空間、版本，以及一或多個資產（或 檔案）。本主題示範如何發佈名為 的套件`my-package`，其命名空間為 `my-ns`、版本 `1.0.0`，並包含一個名為 的資產`asset.tar.gz`。

先決條件：

- AWS Command Line Interface 使用 CodeArtifact 設定（請參閱 使用 AWS CodeArtifact 設定)
- 擁有 CodeArtifact 網域和儲存庫（請參閱 開始使用 AWS CLI)

發佈一般套件

1. 使用下列命令，為您要上傳至套件版本的每個檔案產生 SHA256 雜湊，並將 值放在 環境變數中。此值會用作完整性檢查，以確認檔案內容在最初傳送後未變更。

   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. 呼叫 `publish-package-version` 以上傳資產並建立新的套件版本。

   > **Note**
   >
   > 如果您的套件包含多個資產，您可以呼叫`publish-package-version`一次，讓每個資產上傳。包含每個呼叫 的`--unfinished`引數`publish-package-version`，上傳最終資產時除外。省略 `--unfinished`會將套件版本的狀態設定為 `Published`，並防止其他資產上傳到其中。

或者，--unfinished針對每次對 的呼叫包含 publish-package-version，然後使用 Publishedupdate-package-versions-status命令將套件版本的狀態設定為 。

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%
```

以下將顯示輸出。

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

列出一般套件資產

若要列出包含在一般套件中的資產，請使用 `list-package-version-assets`命令。如需詳細資訊，請參閱[列出套件版本資產](#)。

下列範例列出套件 `1.0.0`版本的資產`my-package`。

列出套件版本資產

- 呼叫 `list-package-version-assets` 以列出包含在一般套件中的資產。

  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
  ```

  以下將顯示輸出。

  ```
  {
      "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"
}
```

下載一般套件資產

若要從一般套件下載資產，請使用 get-package-version-asset命令。如需詳細資訊，請參閱下載套件版本資產。

下列範例會將資產asset.tar.gz從 套件1.0.0版本下載my-package到目前工作目錄，至名為 的檔案asset.tar.gz。

下載套件版本資產

- 呼叫 從一般套件get-package-version-asset下載資產。

  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
  ```

以下將顯示輸出。

```
{
    "assetName": "asset.tar.gz",
    "packageVersion": "1.0.0",
    "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
}
```

搭配 CodeBuild 使用 CodeArtifact CodeBuild

這些主題說明如何在 AWS CodeBuild 建置專案中使用 CodeArtifact 儲存庫中的套件。

主題

- 在 CodeBuild 中使用 npm 套件
- 在 CodeBuild 中使用 Python 套件
- 在 CodeBuild 中使用 Maven 套件
- 在 CodeBuild 中使用 NuGet 套件
- 相依性快取

在 CodeBuild 中使用 npm 套件

下列步驟已使用 CodeBuild 所提供 Docker 映像中列出的作業系統進行測試。

使用 IAM 角色設定許可

在 CodeBuild 中使用 CodeArtifact 的 npm 套件時，需要這些步驟。 CodeBuild

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/iam/：//www. 開啟 IAM 主控台。
2. 在導覽窗格中，選擇角色。在角色頁面上，編輯 CodeBuild 組建專案所使用的角色。此角色必須具有下列許可。

```
{
  "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**
>
> 如果您也想要使用 CodeBuild 發佈套件，請新增 **codeartifact:PublishPackageVersion**許可。

如需詳細資訊，請參閱《IAM 使用者指南》中的修改角色。

登入並使用 npm

若要從 CodeBuild 使用 npm 套件，請從專案 的 `pre-build`區段執行 `login`命令`buildspec.yaml`，以設定 從 CodeArtifact npm 擷取套件。如需詳細資訊，請參閱使用 npm 進行身分驗證。

成功執行 `login`後，您可以從 `build`區段執行npm命令來安裝 npm 套件。

Linux

> **Note**
>
> 只有在您使用較舊的 CodeBuild 映像 AWS CLI `pip3 install awscli --upgrade --user`時，才需要使用 升級 。如果您使用的是最新的映像版本，則可以移除該行。

```
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
```

在 CodeBuild 中使用 Python 套件

下列步驟已使用 CodeBuild 所提供 Docker 映像中列出的作業系統進行測試。

使用 IAM 角色設定許可

在 CodeBuild 中使用來自 CodeArtifact 的 Python 套件時，需要這些步驟。

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/iam/：//www. 開啟 IAM 主控台。
2. 在導覽窗格中，選擇角色。在角色頁面上，編輯 CodeBuild 組建專案所使用的角色。此角色必須具有下列許可。

```
{
  "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**
>
> 如果您也想要使用 CodeBuild 發佈套件，請新增 **`codeartifact:PublishPackageVersion`**許可。

如需詳細資訊，請參閱《IAM 使用者指南》中的修改角色。

登入並使用 pip 或 Twine

若要從 CodeBuild 使用 Python 套件，請從專案`buildspec.yaml`檔案的 `pre-build`區段執行 `login`命令，以設定 從 CodeArtifact `pip` 擷取套件。如需詳細資訊，請參閱搭配 Python 使用 CodeArtifact 。

成功執行 `login` 後，您可以從 `build`區段執行`pip`命令，以安裝或發佈 Python 套件。

Linux

> **Note**
>
> 只有在您使用較舊的 CodeBuild 映像 AWS CLI `pip3 install awscli --upgrade --user`時，才需要使用 升級 。如果您使用的是最新的映像版本，則可以移除該行。

使用 安裝 Python 套件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
```

若要使用 發佈 Python 套件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

使用 安裝 Python 套件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
```

若要使用 發佈 Python 套件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
```

在 CodeBuild 中使用 Maven 套件

使用 IAM 角色設定許可

在 CodeBuild 中使用來自 CodeArtifact 的 Maven 套件時，需要這些步驟。

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/iam/：// 開啟 IAM 主控台。
2. 在導覽窗格中，選擇角色。在角色頁面上，編輯 CodeBuild 組建專案所使用的角色。此角色必須具有下列許可。

```
{
  "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**
>
> 如果您也想要使用 CodeBuild 發佈套件，請新增 **codeartifact:PublishPackageVersion**和 **codeartifact:PutPackageMetadata**許可。

如需詳細資訊，請參閱《IAM 使用者指南》中的修改角色。

使用 gradle 或 mvn

若要搭配 `gradle`或 使用 Maven 套件`mvn`，請將 CodeArtifact 驗證字符存放在 環境變數中，如在 環境變數中傳遞驗證字符中所述。以下是範例。

> **Note**
>
> 只有在您使用較舊的 CodeBuild 映像 AWS CLI `pip3 install awscli --upgrade --user`時，才需要使用 升級 。如果您使用的是最新的映像版本，則可以移除該行。

```
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`
```

若要使用 Gradle：

如果您參考了 Gradle `build.gradle`檔案中的 `CODEARTIFACT_AUTH_TOKEN` 變數，如使用 CodeArtifact 搭配 Gradle 所述，您可以從 `buildspec.yml`建置區段叫用 Gradle 建置。

```
build:
  commands:
    - gradle build
```

若要使用 mvn：

您必須依照搭配使用 CodeArtifact 與 mvn 中的指示，設定 Maven 組態檔案 (`settings.xml` 和 `pom.xml`)。

在 CodeBuild 中使用 NuGet 套件

下列步驟已使用 CodeBuild 所提供 Docker 映像中列出的作業系統進行測試。

主題

- 使用 IAM 角色設定許可
- 使用 NuGet 套件
- 使用 NuGet 套件建置
- 發佈 NuGet 套件

使用 IAM 角色設定許可

在 CodeBuild 中使用 CodeArtifact 中的 NuGet 套件時，需要這些步驟。 CodeBuild

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/iam/：//www. 開啟 IAM 主控台。
2. 在導覽窗格中，選擇角色。在角色頁面上，編輯 CodeBuild 組建專案所使用的角色。此角色必須具有下列許可。

```
{
  "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**
>
> 如果您也想要使用 CodeBuild 發佈套件，請新增 **`codeartifact:PublishPackageVersion`**許可。

如需詳細資訊，請參閱《IAM 使用者指南》中的修改角色。

使用 NuGet 套件

若要從 CodeBuild 使用 NuGet 套件，請在專案的 `buildspec.yaml` 檔案中包含下列項目。

1. 在 `install`區段中，安裝 CodeArtifact 登入資料提供者來設定命令列工具，例如 `msbuild`和 `dotnet`，以建置套件並將其發佈至 CodeArtifact。
2. 在 `pre-build`區段中，將 CodeArtifact 儲存庫新增至 NuGet 組態。

請參閱下列`buildspec.yaml`範例。如需詳細資訊，請參閱搭配 NuGet 使用 CodeArtifact 。

安裝登入資料提供者並新增儲存庫來源後，您可以從 `build`區段執行 NuGet CLI 工具命令，以使用 NuGet 套件。

Linux

若要使用 使用 NuGet 套件`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

若要使用 使用 NuGet 套件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
```

使用 NuGet 套件建置

若要從 CodeBuild 使用 NuGet 套件建置 ，請在專案的 buildspec.yaml 檔案中包含下列項目。

1. 在 `install`區段中，安裝 CodeArtifact 登入資料提供者來設定命令列工具，例如 `msbuild`和 `dotnet`，以建置套件並將其發佈至 CodeArtifact。
2. 在 `pre-build`區段中，將 CodeArtifact 儲存庫新增至 NuGet 組態。

請參閱下列`buildspec.yaml`範例。如需詳細資訊，請參閱搭配 NuGet 使用 CodeArtifact 。

安裝登入資料提供者並新增儲存庫來源後，您可以從 `dotnet build` `build`區段執行 NuGet CLI 工具命令，例如 。

Linux

若要使用 建置 NuGet 套件`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
```

若要使用 建置 NuGet 套件`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

若要使用 建置 NuGet 套件`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
```

若要使用 建置 NuGet 套件`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
```

發佈 NuGet 套件

若要從 CodeBuild 發佈 NuGet 套件，請在專案的 `buildspec.yaml` 檔案中包含下列項目。

1. 在 `install`區段中，安裝 CodeArtifact 登入資料提供者來設定命令列工具，例如 `msbuild`和 `dotnet`，以建置套件並將其發佈至 CodeArtifact。
2. 在 `pre-build`區段中，將 CodeArtifact 儲存庫新增至 NuGet 組態。

請參閱下列`buildspec.yaml`範例。如需詳細資訊，請參閱搭配 NuGet 使用 CodeArtifact 。

安裝登入資料提供者並新增儲存庫來源後，您可以從 `build`區段執行 NuGet CLI 工具命令，並發佈 NuGet 套件。

Linux

若要使用 發佈 NuGet 套件`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

若要使用 發佈 NuGet 套件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
```

相依性快取

您可以在 CodeBuild 中啟用本機快取，以減少每個組建需要從 CodeArtifact 擷取的相依性數目。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的在 中建置快取 AWS CodeBuild。啟用自訂本機快取後，請將快取目錄新增至專案的 buildspec.yaml 檔案。

例如，如果您使用的是 mvn，請使用下列項目。

```
cache:
  paths:
    - '/root/.m2/**/*'
```

對於其他工具，請使用此表格中顯示的快取資料夾。

工具	快取目錄
mvn	/root/.m2/**/*

工具	快取目錄
gradle	/root/.gradle/caches/**/*
pip	/root/.cache/pip/**/*
npm	/root/.npm/**/*
nuget	/root/.nuget/**/*
yarn (classic)	/root/.cache/yarn/**/*

監控 CodeArtifact

監控是維護 CodeArtifact 和其他 AWS 解決方案可靠性、可用性和效能的重要部分。 AWS 提供下列監控工具來監看 CodeArtifact、在發生錯誤時回報，並適時採取自動動作：

- 您可以使用 Amazon EventBridge 自動化您的 AWS 服務，並自動回應系統事件，例如應用程式可用性問題或資源變更。 AWS 服務的事件會以接近即時的方式傳送到 EventBridge。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南和 CodeArtifact 事件格式和範例。
- 您可以使用 Amazon CloudWatch 指標，依操作檢視 CodeArtifact 用量。CloudWatch 指標包含對 CodeArtifact 提出的所有請求，且請求會依帳戶顯示。您可以透過導覽至 Usage/By AWS Resource AWS 命名空間，在 CloudWatch 指標中檢視這些指標。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 指標。 Amazon CloudWatch

主題

- 監控 CodeArtifact 事件
- 使用事件啟動 CodePipeline 執行
- 使用 事件來執行 Lambda 函數

監控 CodeArtifact 事件

CodeArtifact 已與 Amazon EventBridge 整合，Amazon EventBridge 是一項可自動化和回應事件的服務，包括 CodeArtifact 儲存庫中的變更。您可以建立事件的規則，並設定事件符合規則時會發生的情況。EventBridge 之前被稱為 CloudWatch Events。

事件可以觸發下列動作：

- 叫用 AWS Lambda 函數。
- 啟用 AWS Step Functions 狀態機器。
- 通知 Amazon SNS 主題或 Amazon SQS 佇列。
- 在 中啟動管道 AWS CodePipeline。

CodeArtifact 會在建立、修改或刪除套件版本時建立事件。以下是 CodeArtifact 事件的範例：

- 發佈新的套件版本 （ 例如，執行 `npm publish`)。

- 將新的資產新增至現有的套件版本 （例如，將新的 JAR 檔案推送至現有的 Maven 套件）。
- 使用 將套件版本從一個儲存庫複製到另一個儲存庫`copy-package-versions`。如需詳細資訊，請參閱在儲存庫之間複製套件。
- 使用 刪除套件版本`delete-package-versions`。如需詳細資訊，請參閱刪除套件或套件版本。
- 使用 刪除套件版本`delete-package`。每個已刪除的套件版本都會發佈一個事件。如需詳細資訊，請參閱刪除套件或套件版本。
- 從上游儲存庫擷取套件版本時，在下游儲存庫中保留套件版本。如需詳細資訊，請參閱在 CodeArtifact 中使用上游儲存庫。
- 將套件版本從外部儲存庫擷取到 CodeArtifact 儲存庫。如需詳細資訊，請參閱將 CodeArtifact 儲存庫連線至公有儲存庫。

事件會交付到擁有網域的帳戶，以及管理儲存庫的帳戶。例如，假設 帳戶`111111111111`擁有網域`my_domain`。帳戶會在`my_domain`名為 的 中`222222222222`建立儲存庫`repo2`。當新套件版本發佈至 時`repo2`，兩個帳戶都會收到 EventBridge 事件。網域擁有帳戶 (111111111111) 會接收網域中所有儲存庫的事件。如果單一帳戶同時擁有網域和其中的儲存庫，則只會傳送單一事件。

下列主題說明 CodeArtifact 事件格式。它們會示範如何設定 CodeArtifact 事件，以及如何將事件與其他 AWS 服務搭配使用。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon EventBridge 入門。

CodeArtifact 事件格式和範例

以下是事件欄位和描述，以及 CodeArtifact 事件的範例。

CodeArtifact 事件格式

所有 CodeArtifact 事件都包含下列欄位。

事件欄位	描述
version	事件格式的版本。目前只有單一版本 `0`。
id	事件的唯一識別符。
詳細資訊類型	事件的類型。這會決定 `detail` 物件中的欄位。`detail-type` 目前支援的是 `CodeArtifact Package Version State Change` 。

事件欄位	描述
source	事件的來源。對於 CodeArtifact，它會是 `aws.codeartifact` 。
帳戶	接收事件之帳戶 AWS 的帳戶 ID。
time	觸發事件的確切時間。
region	觸發事件的區域。
resources	包含已變更之套件 ARN 的清單。清單包含一個項目。如需套件 ARN 格式的資訊，請參閱 授予套件的寫入存取權。
domainName	包含包含套件之儲存庫的網域。
domainOwner	網域擁有者的帳戶 AWS ID。
repositoryName	包含套件的儲存庫。
repositoryAdministrator	儲存庫管理員的帳戶 AWS ID。
packageFormat	觸發事件的套件格式。
packageNamespace	觸發事件的套件命名空間。
packageName	觸發事件的套件名稱。
packageVersion	觸發事件的套件版本。
packageVersionState	事件觸發時的套件版本狀態。可能值為 `Unfinished` 、`Published` 、`Unlisted`、`Archived` 和 `Disposed`。
packageVersionRevision	在事件觸發時，唯一識別資產狀態和套件版本中繼資料的值。如果套件版本已修改（例如，透過將另一個 JAR 檔案新增至 Maven 套件），則`packageVersionRevision` 變更。

事件欄位	描述
change.assetsAdded	新增至觸發事件之套件的資產數目。資產的範例為 Maven JAR 檔案或 Python 滾輪。
change.assetsRemoved	從觸發事件的套件中移除的資產數目。
changes.assetsUpdated	在觸發事件的套件中修改的資產數目。
changes.metadataUpdated	`true` 如果事件包含修改後的套件層級中繼資料，則設定為 的布林值。例如，事件可能會修改 Maven `pom.xml` 檔案。
changes.statusChanged	如果`packageVersionStatus` 修改`true`事件的 ，則設定為 的布林值 （例如，如果從 `packageVersionStatus` 變更為 `Unfinished` `Published`)。
operationType	描述套件版本變更的高階類型。可能的值為 `Created`、`Updated` 和 `Deleted`。
sequenceNumber	整數，指定套件的事件編號。套件上的每個事件都會遞增 ，`sequenceNumber` 以便事件可以循序排列。事件可以將 遞增`sequenceNumber` 任何整數。 **Note** EventBridge 事件可能會收到順序不正確。 `sequenceNumber` 可用來判斷其實際順序。
eventDeduplicationId	用來區分重複 EventBridge 事件的 ID。在極少數情況下，EventBridge 可能會針對單一事件或排程時間觸發相同的規則多次。或者，它可能會針對指定的觸發規則多次調用相同的目標。

CodeArtifact 事件範例

以下是發佈 npm 套件時，可能會觸發的 CodeArtifact 事件範例。

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

使用事件啟動 CodePipeline 執行

此範例示範如何設定 Amazon EventBridge 規則， AWS CodePipeline 以便在 CodeArtifact 儲存庫中的套件版本發佈、修改或刪除時開始執行。

主題

- 設定 EventBridge 許可
- 建立 EventBridge 規則
- 建立 EventBridge 規則目標

設定 EventBridge 許可

您必須新增 EventBridge 的許可，才能使用 CodePipeline 叫用您建立的規則。若要使用 AWS Command Line Interface (AWS CLI) 新增這些許可，請遵循《 AWS CodePipeline 使用者指南》中為 CodeCommit 來源 (CLI) 建立 CloudWatch Events 規則中的步驟 1。

建立 EventBridge 規則

若要建立規則，請使用 `put-rule`命令搭配 `--name`和 `--event-pattern` 參數。事件模式會指定與每個事件內容相符的值。如果模式符合事件，則會觸發目標。例如，下列模式符合`my_domain`網域中`myrepo`儲存庫的 CodeArtifact 事件。

```
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"]}}'
```

建立 EventBridge 規則目標

下列命令會將目標新增至規則，以便在事件符合規則時觸發 CodePipeline 執行。針對 `RoleArn` 參數，指定本主題稍早所建立角色的 Amazon Resource Name (ARN)。

```
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'
```

使用 事件來執行 Lambda 函數

此範例說明如何設定 EventBridge 規則，在 CodeArtifact 儲存庫中的套件版本發佈、修改或刪除時啟動 AWS Lambda 函數。

如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的教學課程：使用 EventBridge 排程 AWS Lambda 函數。 EventBridge

主題

- 建立 EventBridge 規則
- 建立 EventBridge 規則目標
- 設定 EventBridge 許可

建立 EventBridge 規則

若要建立啟動 Lambda 函數的規則，請使用 `put-rule`命令搭配 `--name`和 `--event-pattern`選項。下列模式指定`my_domain`網域中任何儲存庫`@types`範圍內的 npm 套件。

```
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"]}}'
```

建立 EventBridge 規則目標

下列命令會將目標新增至規則，在事件符合規則時執行 Lambda 函數。針對 `arn` 參數，指定 Lambda 函數的 Amazon Resource Name (ARN)。

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

設定 EventBridge 許可

使用 `add-permission`命令來授予規則呼叫 Lambda 函數的許可。針對 `--source-arn` 參數，指定您在此範例中稍早建立之規則的 ARN。

```
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
```

CodeArtifact 中的安全性

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構是專為滿足最安全敏感組織的需求而建置。

安全是 AWS 與您之間的共同責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用於 CodeArtifact 的合規計劃，請參閱合規計劃的 AWS 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 CodeArtifact 時套用共同責任模型。下列主題說明如何設定 CodeArtifact 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 CodeArtifact 資源。

主題

- 資料保護 in AWS CodeArtifact
- 監控 CodeArtifact
- AWS CodeArtifact 的合規驗證
- AWS CodeArtifact 身分驗證和字符
- 彈性 in AWS CodeArtifact
- 基礎設施安全 in AWS CodeArtifact
- 相依性替代攻擊
- Identity and Access Management for AWS CodeArtifact

資料保護 in AWS CodeArtifact

AWS 共同責任模型適用於 AWS CodeArtifact 中的資料保護。如此模型所述， AWS 負責保護執行所有 的 全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 CodeArtifact 或使用主控台、API AWS CLI或 AWS SDKs的其他 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

資料加密

加密是 CodeArtifact 安全性的重要部分。有些加密，例如傳輸中的資料，是預設提供的，不需要您執行任何動作。其他加密，例如靜態資料，您可以在建立專案或建置時設定 。

- 靜態資料加密 - 存放在 CodeArtifact 中的所有資產都是使用 AWS KMS keys (KMS 金鑰） 加密。這包括所有儲存庫中所有套件中的所有資產。每個網域都會使用一個 KMS 金鑰來加密其所有資產。根據預設，會使用 AWS 受管 KMS 金鑰，因此您不需要建立 KMS 金鑰。如果需要，您可以使用您建立和設定的客戶受管 KMS 金鑰。如需詳細資訊，請參閱AWS Key Management Service 《 使用者指南》中的建立金鑰和AWS 金鑰管理服務概念。您可以在建立網域時指定客戶管理的 KMS 金鑰。如需詳細資訊，請參閱在 CodeArtifact 中使用網域。
- 傳輸中的資料加密 - 客戶與 CodeArtifact 之間，以及 CodeArtifact 與其下游相依性之間使用 TLS 加密保護的所有通訊。

流量隱私權

您可以透過設定 CodeArtifact 使用界面虛擬私有雲端 (VPC) 端點來改善 CodeArtifact 網域及其所含資產的安全性。若要這樣做，您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。如需詳細資訊，請參

閱使用 Amazon VPC 端點。如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱 AWS PrivateLink 和透過 PrivateLink 存取 AWS 服務。

監控 CodeArtifact

監控是維護 AWS CodeArtifact 和 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點失敗時更輕鬆地偵錯。 AWS 提供下列項目來監控 CodeArtifact 資源和回應潛在事件：

主題

- 使用 記錄 CodeArtifact API 呼叫 AWS CloudTrail

使用 記錄 CodeArtifact API 呼叫 AWS CloudTrail

CodeArtifact 已與 整合AWS CloudTrail，此服務提供由使用者、角色或 CodeArtifact 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 CodeArtifact 的所有 API 呼叫擷取為事件，包括套件管理員用戶端的呼叫。

如果您建立線索，則可以將 CloudTrail 事件持續交付至 Amazon Simple Storage Service (Amazon S3) 儲存貯體，包括 CodeArtifact 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的事件歷史記錄檢視最新事件。您可以使用 CloudTrail 所收集的資訊，判斷對 CodeArtifact 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「AWS CloudTrail 使用者指南」。

CloudTrail 中的 CodeArtifact 資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 CodeArtifact 中發生時，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他 AWS 服務事件。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html。

若要持續記錄您 AWS 帳戶中的事件，包括 CodeArtifact 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您也可以設定其他 AWS 服務，以進一步分析 CloudTrail 日誌中收集的事件資料並對其採取行動。如需詳細資訊，請參閱下列主題：

- 為您的 AWS 帳戶建立線索

- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知

當 AWS 您的帳戶中啟用 CloudTrail 記錄時，對 CodeArtifact 動作發出的 API 呼叫會在 CloudTrail 日誌檔案中追蹤，並在其中與其他 AWS 服務記錄一起寫入。CloudTrail 會根據期間與檔案大小，決定何時建立與寫入新檔案。

CloudTrail 會記錄所有 CodeArtifact 動作。例如，呼叫 `ListRepositories` (在 、 `aws codeartifact list-repositories`) AWS CLI、 `CreateRepository` (`aws codeartifact create-repository`) 和 `ListPackages`(`aws codeartifact list-packages`) 動作除了套件管理員用戶端命令之外，還會在 CloudTrail 日誌檔案中產生項目。套件管理員用戶端命令通常會對伺服器發出多個 HTTP 請求。每個請求都會產生單獨的 CloudTrail 日誌事件。

CloudTrail 日誌的跨帳戶交付

最多三個不同的帳戶會收到單一 API 呼叫的 CloudTrail 日誌：

- 提出請求的帳戶，例如，呼叫 的帳戶`GetAuthorizationToken`。
- 儲存庫管理員帳戶 - 例如，管理`ListPackages`呼叫之儲存庫的帳戶。
- 網域擁有者的帳戶，例如，擁有網域的帳戶，其中包含呼叫 API 的儲存庫。

對於對網域而非特定儲存庫執行動作`ListRepositoriesInDomain`的 APIs，只有呼叫帳戶和網域擁有者的帳戶會收到 CloudTrail 日誌。對於未針對任何資源授權`ListRepositories`的 APIs，只有發起人的帳戶會收到 CloudTrail 日誌。

了解 CodeArtifact 日誌檔案項目

CloudTrail 日誌檔案可以包含一或多個日誌項目。每一項目均列出多個 JSON 格式的事件。一個日誌事件為任何來源提出的單一請求，並包含請求動作、動作的日期和時間、請求參數等資訊。日誌項目並非公有 API 呼叫的有序堆疊追蹤，因此不會以任何特定順序顯示。

主題

- 範例：呼叫 GetAuthorizationToken API 的日誌項目
- 範例：用於擷取 npm 套件版本的日誌項目

範例：呼叫 GetAuthorizationToken API 的日誌項目

建立的日誌項目會在 `requestParameters` 欄位中`GetAuthorizationToken`包含網域名稱。

```
{
  "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"
}
```

範例：用於擷取 npm 套件版本的日誌項目

所有套件管理員用戶端提出的請求，包括**npm**用戶端，都會在 `requestParameters` 欄位中記錄其他資料，包括網域名稱、儲存庫名稱和套件名稱。URL 路徑和 HTTP 方法會記錄在 `additionalEventData` 欄位中。


```
{
   "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"
}
```

AWS CodeArtifact 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃範圍內，請參閱AWS 服務 合規計劃範圍內的 ，然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS 合規計劃。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載 中的 AWS Artifact報告。

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源以協助合規：

- 安全合規與治理 - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- HIPAA 合格服務參考 – 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- AWS 合規資源 – 此工作手冊和指南集合可能適用於您的產業和位置。
- AWS 客戶合規指南 – 透過合規的角度了解共同責任模型。本指南摘要說明跨多個架構 （包括國家標準與技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 保護 AWS 服務 並映射指引至安全控制的最佳實務。
- 《 AWS Config 開發人員指南》中的使用 規則評估資源 – AWS Config 服務會評估資源組態是否符合內部實務、產業準則和法規。
- AWS Security Hub – 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「Security Hub 控制參考」。
- Amazon GuardDuty – 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- AWS Audit Manager – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，並符合法規和業界標準。

AWS CodeArtifact 身分驗證和字符

CodeArtifact 要求使用者向 服務進行身分驗證，以發佈或取用套件版本。您必須使用 AWS 憑證建立授權字符，以驗證 CodeArtifact 服務。若要建立授權字符，您必須擁有正確的許可。如需建立授權字符所需的許可，請參閱 中的 GetAuthorizationToken項目AWS CodeArtifact 許可參考。如需 CodeArtifact 許可的一般資訊，請參閱 How AWS CodeArtifact 可與 IAM 搭配使用。

若要從 CodeArtifact 擷取授權字符，您必須呼叫 GetAuthorizationToken API。您可以使用 login或 get-authorization-token命令來 AWS CLI呼叫 GetAuthorizationToken 。

> **Note**
>
> 根使用者無法呼叫 GetAuthorizationToken。

- aws codeartifact login：此命令可讓您輕鬆地設定常用套件管理員，以在單一步驟中使用 CodeArtifact。呼叫 會使用 login 擷取字符，GetAuthorizationToken並使用字符和正確的 CodeArtifact 儲存庫端點設定套件管理員。支援套件管理員如下：
  - dotnet
  - npm
  - nuget
  - pip
  - Swift
  - 雙身
- aws codeartifact get-authorization-token：對於 不支援的套件管理員login，您可以以get-authorization-token直接呼叫 ，然後視需要使用字符設定套件管理員，例如，將套件管理員新增至組態檔案或將其儲存為環境變數。

CodeArtifact 授權字符在預設的 12 小時內有效。權杖可以設定 15 分鐘到 12 小時的生命週期。當生命週期過期時，您必須擷取另一個字符。字符生命週期會在呼叫 get-authorization-token login或 之後開始。

如果在擔任角色時get-authorization-token呼叫 login或 ，您可以將 值設定為 --duration-seconds ，將字符的生命週期設定為等於角色工作階段持續時間的剩餘時間0。否則，字符生命週期與角色的最大工作階段持續時間無關。例如，假設您呼叫 sts assume-role並指定 15 分鐘的工作階段持續時間，然後呼叫 login來擷取 CodeArtifact 授權字符。在此情況下，即使超過 15 分鐘的工作階

段持續時間，字符仍在整個 12 小時期間內有效。如需控制工作階段持續時間的資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用 `login`命令建立的字符

`aws codeartifact login` 命令會使用 擷取字符`GetAuthorizationToken`，並使用字符和正確的 CodeArtifact 儲存庫端點設定您的套件管理員。

下表說明 `login`命令的參數。

參數	必要	描述
`--tool`	是	要驗證的套件管理員。可能的值為 `dotnet`、`npm`、`pip`、`nuget`、`swift`和 `twine`。
`--domain`	是	儲存庫所屬的網域名稱。
`--domain-owner`	否	網域擁有者的 ID。如果存取您未驗證 AWS 之帳戶所擁有的網域，則需要此參數。如需詳細資訊，請參閱「跨帳戶網域」。
`--repository`	是	要驗證的儲存庫名稱。
`--duration-seconds`	否	登入資訊有效的時間，以秒為單位。最小值為 900*，最大值為 43200。
`--namespace`	否	將命名空間與您的儲存庫工具建立關聯。
`--dry-run`	否	僅列印將執行的命令，以將您的工具與儲存庫連線，而無需對組態進行任何變更。

*在擔任角色`login`時呼叫 時，值 0 也有效。`login` 使用 呼叫 `--duration-seconds 0` 會建立生命週期等於擔任角色之工作階段持續時間中剩餘時間的字符。

下列範例示範如何使用 `login`命令擷取授權字符。

```
aws codeartifact login \
    --tool dotnet | npm | nuget | pip | swift | twine \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo
```

如需如何搭配 npm 使用 `login`命令的特定指引，請參閱 搭配 CodeArtifact 設定和使用 npm。對於 Python，請參閱 搭配 Python 使用 CodeArtifact 。

呼叫 **GetAuthorizationToken** API 所需的許可

呼叫 CodeArtifact `GetAuthorizationToken` API 需要 `sts:GetServiceBearerToken`和 `codeartifact:GetAuthorizationToken`許可。

若要搭配 CodeArtifact 儲存庫使用套件管理員，您的 IAM 使用者或角色必須允許 `sts:GetServiceBearerToken`。雖然 `sts:GetServiceBearerToken`可以新增至 CodeArtifact 網域資源政策，但該許可在該政策中沒有作用。

使用 **GetAuthorizationToken** API 建立的字符

您可以呼叫 `get-authorization-token` 從 CodeArtifact 擷取授權字符。

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text
```

您可以使用 `--duration-seconds`引數變更字符的有效時間長度。最小值為 900，最大值為 43200。下列範例會建立會持續 1 小時 (3600 秒） 的字符。

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 3600
```

如果在擔任角色get-authorization-token時呼叫 ，字符生命週期與角色的工作階段持續時間上限無關。您可以將 設定為 --duration-seconds0，將權杖設定為在擔任角色的工作階段持續時間過期時過期。

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 0
```

如需詳細資訊，請參閱下列文件：

- 如需字符和環境變數的指引，請參閱 使用 環境變數傳遞身分驗證字符。
- 對於 Python 使用者，請參閱 設定不含登入命令的 pip或 使用 CodeArtifact 設定和使用雙身。
- 對於 Maven 使用者，請參閱 搭配 Gradle 使用 CodeArtifact 或 將 CodeArtifact 與 mvn 搭配使用。
- 如需 npm 使用者，請參閱 不使用登入命令設定 npm。

使用 環境變數傳遞身分驗證字符

AWS CodeArtifact 使用 GetAuthorizationToken API 提供的授權字符來驗證和授權來自 Maven 和 Gradle 等建置工具的請求。如需這些身分驗證字符的詳細資訊，請參閱 使用 GetAuthorizationToken API 建立的字符。

您可以將這些身分驗證權杖存放在可由建置工具讀取的環境變數中，以取得從 CodeArtifact 儲存庫擷取套件或將套件發佈至其中所需的權杖。

基於安全考量，此方法偏好將字符存放在檔案，而該檔案可能由其他使用者或程序讀取，或意外簽入來源控制。

1. 如中所述設定您的 AWS 登入資料安裝或升級 ，然後設定 AWS CLI。
2. 設定 CODEARTIFACT_AUTH_TOKEN 環境變數：

> **Note**
>
> 在某些情況下，您不需要包含 --domain-owner引數。如需詳細資訊，請參閱跨帳戶網域。

- macOS 或 Linux：

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

- Windows （ 使用預設命令 shell)：

```
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
```

撤銷 CodeArtifact 授權字符

當已驗證的使用者建立權杖來存取 CodeArtifact 資源時，該權杖會持續到其可自訂的存取期間結束為止。預設存取期間為 12 小時。在某些情況下，您可能想要在存取期間過期之前撤銷對權杖的存取。您可以依照這些指示撤銷 CodeArtifact 資源的存取權。

如果您使用臨時安全登入資料建立存取權杖，例如擔任的角色或聯合身分使用者存取，您可以透過更新 IAM 政策來拒絕存取來撤銷存取。如需詳細資訊，請參閱《IAM 使用者指南》中的停用暫時安全登入資料的許可。

如果您使用長期 IAM 使用者登入資料來建立存取權杖，則必須修改使用者的政策以拒絕存取，或刪除 IAM 使用者。如需詳細資訊，請參閱變更 IAM 使用者的許可或刪除 IAM 使用者。

彈性 in AWS CodeArtifact

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置的。 AWS 區域提供多個實體隔離且隔離的可用區域，這些區域以低延遲、高輸送量和高度備援的網路連接。AWS CodeArtifact 會在多個可用區域中運作，並將成品資料和中繼資料存放在 Amazon S3 和 Amazon DynamoDB 中。您的加密資料會以備援方式存放在多個設施和每個設施中的多個裝置，使其具有高可用性和高耐用性。

如需 AWS 區域和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

基礎設施安全 in AWS CodeArtifact

作為受管服務， AWS CodeArtifact 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 CodeArtifact。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 以產生暫時安全憑證以簽署請求。

相依性替代攻擊

套件管理員可簡化封裝和共用可重複使用程式碼的程序。這些套件可能是組織開發的私有套件，用於其應用程式，或者可能是公有套件，通常是在組織外部開發並由公有套件儲存庫分發的開放原始碼套件。請求套件時，開發人員依賴其套件管理員來擷取其相依性的新版本。相依性替代攻擊也稱為相依性混淆攻擊，利用套件管理員通常無法區分套件的合法版本與惡意版本。

相依性替代攻擊屬於稱為軟體供應鏈攻擊的駭客子集。軟體供應鏈攻擊是一種利用軟體供應鏈中任何位置漏洞的攻擊。

相依性替代攻擊可以針對任何同時使用內部開發的套件和從公有儲存庫擷取的套件。攻擊者會識別內部套件名稱，然後在公有套件儲存庫中以策略方式放置同名的惡意程式碼。一般而言，惡意程式碼會發佈在版本編號較高的套件中。套件管理員會從這些公有摘要擷取惡意程式碼，因為他們認為惡意程式套件是套件的最新版本。這會導致所需套件與惡意套件之間的「混淆」或「替代」，導致程式碼遭到入侵。

為了防止相依性替代攻擊， AWS CodeArtifact 提供套件原始伺服器控制。套件原始伺服器控制是控制套件如何新增至儲存庫的設定。這些控制項可用來確保套件版本無法直接發佈至您的儲存庫並從公有來源擷取，保護您免受相依性替代攻擊。原始伺服器控制可以在個別套件和多個套件上設定，方法是在套件群組上設定原始伺服器控制。如需套件原始伺服器控制以及如何變更它們的詳細資訊，請參閱 編輯套件原始伺服器控制項和 套件群組原始伺服器控制。

Identity and Access Management for AWS CodeArtifact

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （ 登入 ） 和授權 （ 具有許可 ） 來使用 CodeArtifact 資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- How AWS CodeArtifact 可與 IAM 搭配使用
- AWS CodeArtifact 的身分型政策範例
- 使用標籤控制 CodeArtifact 資源的存取
- AWS CodeArtifact 許可參考
- 疑難排解 AWS CodeArtifact 身分和存取

目標對象

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，取決於您在 CodeArtifact 中執行的工作。

服務使用者 – 如果您使用 CodeArtifact 服務來執行任務，您的管理員會為您提供所需的登入資料和許可。當您使用更多 CodeArtifact 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 CodeArtifact 中的功能，請參閱 疑難排解 AWS CodeArtifact 身分和存取。

服務管理員 – 如果您在公司負責 CodeArtifact 資源，您可能擁有 CodeArtifact 的完整存取權。您的任務是判斷服務使用者應存取哪些 CodeArtifact 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何搭配 CodeArtifact 使用 IAM，請參閱 How AWS CodeArtifact 可與 IAM 搭配使用。

IAM 管理員 – 如果您是 IAM 管理員，建議您了解撰寫政策以管理 CodeArtifact 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的 CodeArtifact 身分型政策範例，請參閱 AWS CodeArtifact 的身分型政策範例。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 （ 登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分身分身分登入 。 AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

視您的使用者類型而定，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱AWS 登入 《 使用者指南》中的如何登入您的 AWS 帳戶 。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的 憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的適用於 API 請求的AWS Signature 第 4 版。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的 IAM 中的AWS 多重要素驗證。

AWS 帳戶 根使用者

建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

根據最佳實務， 要求人類使用者，包括需要管理員存取權的使用者，使用身分提供者的聯合 AWS 服務 身分來使用臨時憑證來存取 。

聯合身分是來自您企業使用者目錄、Web 身分提供者、 AWS Directory Service、Identity Center 目錄或任何使用透過身分來源提供的憑證 AWS 服務 存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接和同步到您自己的身分來源中的一組使用者 AWS 帳戶 和群組，以便

在所有 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是 中的身分 AWS 帳戶 ，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色 （主控台）。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以將政策直接連接到資源（而不是使用角色做為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。
  - 服務連結角色 – 服務連結角色是一種連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體，並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源建立關聯時， AWS 會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交

集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。

- 服務控制政策 SCPs) – SCPs是 JSON 政策，可指定 in. 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶 之多個 的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs的詳細資訊，包括 AWS 服務 支援 RCPs 清單，請參閱AWS Organizations《 使用者指南》中的資源控制政策 (RCPs)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

How AWS CodeArtifact 可與 IAM 搭配使用

在您使用 IAM 管理 CodeArtifact 的存取權之前，請先了解哪些 IAM 功能可與 CodeArtifact 搭配使用。

您可以搭配 AWS CodeArtifact 使用的 IAM 功能

IAM 功能	CodeArtifact 支援
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是

IAM 功能	CodeArtifact 支援
政策條件索引鍵 (服務特定)	否
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	是
服務角色	否
服務連結角色	否

若要全面了解 CodeArtifact 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

CodeArtifact 的身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

CodeArtifact 的身分型政策範例

若要檢視 CodeArtifact 身分型政策的範例，請參閱 AWS CodeArtifact 的身分型政策範例。

CodeArtifact 中的資源型政策

支援資源型政策：是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源

的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

CodeArtifact 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 CodeArtifact 動作清單，請參閱《服務授權參考》中的AWS CodeArtifact 定義的動作。

CodeArtifact 中的政策動作在動作之前使用以下字首：

```
codeartifact
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "codeartifact:action1",
      "codeartifact:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `Describe` 文字的所有動作，請包含以下動作：

```
"Action": "codeartifact:Describe*"
```

若要檢視 CodeArtifact 身分型政策的範例，請參閱 AWS CodeArtifact 的身分型政策範例。

CodeArtifact 的政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 CodeArtifact 資源類型及其 ARNs，請參閱《服務授權參考》中的AWS CodeArtifact 定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱AWS CodeArtifact 定義的動作。若要查看在政策中指定 CodeArtifact 資源 ARNs 的範例，請參閱 AWS CodeArtifact 資源和操作。

CodeArtifact 的政策條件索引鍵

支援服務特定的政策條件金鑰：否

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值， 會使用邏輯OR操作 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

> **Note**
>
> AWS CodeArtifact 不支援下列 AWS 全域條件內容金鑰：
>
> - Referer
> - UserAgent

若要查看 CodeArtifact 條件索引鍵的清單，請參閱《服務授權參考》中的AWS CodeArtifact 的條件索引鍵。若要了解您可以使用條件金鑰的動作和資源，請參閱AWS CodeArtifact 定義的動作。

若要檢視 CodeArtifact 身分型政策的範例，請參閱 AWS CodeArtifact 的身分型政策範例。

CodeArtifact ACLs

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

ABAC 搭配 CodeArtifact

支援 ABAC (政策中的標籤)：部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 中 AWS，這些屬性稱為標籤。您可以將標籤連接至 IAM 實體 （使用者或角色） 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的使用 ABAC 授權定義許可。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

如需標記 CodeArtifact 資源的詳細資訊，包括根據資源上的標籤限制資源存取的範例身分型政策，請參閱 使用標籤控制 CodeArtifact 資源的存取。

搭配 CodeArtifact 使用臨時登入資料

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱《AWS 服務 IAM 使用者指南》中的 使用 IAM。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則會使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取 時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的從使用者切換至 IAM 角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議使用您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

CodeArtifact 的跨服務主體許可

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱轉發存取工作階段。

有兩個 CodeArtifact API 動作需要呼叫委託人擁有其他服務的許可：

1. `GetAuthorizationToken` 需要 `sts:GetServiceBearerToken`與 `codeartifact:GetAuthorizationToken`。
2. `CreateDomain`提供非預設加密金鑰時， 需要 KMS 金鑰`kms:CreateGrant`上的 `kms:DescribeKey`和 以及 `codeartifact:CreateDomain`。

如需 CodeArtifact 中動作所需許可和資源的詳細資訊，請參閱 AWS CodeArtifact 許可參考。

CodeArtifact 的服務角色

支援服務角色：否

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。

> **Warning**
>
> 變更服務角色的許可可能會中斷 CodeArtifact 功能。只有在 CodeArtifact 提供指引時，才能編輯服務角色。

CodeArtifact 的服務連結角色

支援服務連結角色：否

服務連結角色是連結至 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

AWS CodeArtifact 的身分型政策範例

根據預設，使用者和角色沒有建立或修改 CodeArtifact 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

如需 CodeArtifact 定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARNs 格式，請參閱《服務授權參考》中的AWS CodeArtifact 的動作、資源和條件索引鍵。

主題

- 政策最佳實務
- 使用 CodeArtifact 主控台
- AWS CodeArtifact 的 AWS 受管 （預先定義） 政策
- 允許使用者檢視自己的許可
- 允許使用者取得儲存庫和網域的相關資訊
- 允許使用者取得特定網域的相關資訊
- 允許使用者取得特定儲存庫的相關資訊

- 限制授權字符持續時間

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 CodeArtifact 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 使用服務動作 AWS 服務，您也可以使用條件來授予存取，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 CodeArtifact 主控台

若要存取 AWS CodeArtifact 主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 中 CodeArtifact 資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍然可以使用 CodeArtifact 主控台，也請將 `AWSCodeArtifactAdminAccess`或 `AWSCodeArtifactReadOnlyAccess` AWS 受管政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

AWS CodeArtifact 的 AWS 受管 （預先定義） 政策

AWS 透過提供由 建立和管理的獨立 IAM 政策，解決許多常見的使用案例 AWS。這些 AWS 受管政策會授予常見使用案例的必要許可，因此您可以避免調查需要哪些許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS Managed Policies (AWS 受管政策)。

下列 AWS 受管政策是 AWS CodeArtifact 的專屬政策，您可以將其連接至您帳戶中的使用者。

- `AWSCodeArtifactAdminAccess` – 提供 CodeArtifact 的完整存取權，包括管理 CodeArtifact 網域的許可。

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

- `AWSCodeArtifactReadOnlyAccess` – 提供 CodeArtifact 的唯讀存取權。

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

若要建立和管理 CodeArtifact 服務角色，您還必須連接名為 的 AWS 受管政策`IAMFullAccess`。

您也可以建立自己的自訂 IAM 政策，以允許 CodeArtifact 動作和資源的許可。您可以將這些自訂政策連接至需要這些許可的 IAM 使用者或群組。

允許使用者檢視自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上完成此動作的許可，或使用 AWS CLI 或 AWS API 以程式設計方式完成此動作的許可。

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

允許使用者取得儲存庫和網域的相關資訊

下列政策允許 IAM 使用者或角色列出和描述任何類型的 CodeArtifact 資源，包括網域、儲存庫、套件和資產。此政策也包含 `codeArtifact:ReadFromRepository`許可，允許主體從 CodeArtifact 儲存庫擷取套件。它不允許建立新的網域或儲存庫，也不允許發佈新的套件。

呼叫 `GetAuthorizationToken` API 需要 `codeartifact:GetAuthorizationToken`和 `sts:GetServiceBearerToken`許可。

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

允許使用者取得特定網域的相關資訊

以下顯示許可政策的範例，允許使用者只列出名稱為 123456789012的任何網域us-east-2的 區域中的網域my。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codeartifact:ListDomains",
         "Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/my*"
      }
   ]
}
```

允許使用者取得特定儲存庫的相關資訊

以下顯示許可政策範例，允許使用者取得以 結尾的儲存庫相關資訊test，包括其中套件的相關資訊。使用者將無法發佈、建立或刪除資源。

呼叫 GetAuthorizationToken API 需要 codeartifact:GetAuthorizationToken和 sts:GetServiceBearerToken許可。

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

限制授權字符持續時間

使用者必須使用授權字符向 CodeArtifact 驗證身分，才能發佈或取用套件版本。授權字符僅在設定的生命週期內有效。字符的預設生命週期為 12 小時。如需授權字符的詳細資訊，請參閱 AWS CodeArtifact 身分驗證和字符。

擷取字符時，使用者可以設定字符的生命週期。授權字符生命週期的有效值為 0，以及介於 900(15 分鐘） 和 43200(12 小時） 之間的任何數字。的值0會建立持續時間等於使用者角色臨時登入資料的字符。

管理員可以使用連接到使用者或群組的許可政策中的 sts:DurationSeconds條件索引鍵，限制授權字符的生命週期有效值。如果使用者嘗試建立生命週期超過有效值的授權字符，則字符建立將會失敗。

下列範例政策會限制 CodeArtifact 使用者所建立授權字符的可能持續時間。

範例政策：將字符生命週期限制為剛好 12 小時 (43200 秒）

使用此政策，使用者只能建立生命週期為 12 小時的授權字符。

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

```
}
```

範例政策：將字符生命週期限制在 15 分鐘到 1 小時之間，或等於使用者的臨時登入資料期間

使用此政策，使用者將能夠建立介於 15 分鐘到 1 小時之間的有效字符。使用者也可以透過0為 指定 來建立權杖，以持續其角色的暫時登入資料持續時間--durationSeconds。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codeartifact:*",
            "Resource": "*"
        },
        {
            "Sid": "sts",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "NumericLessThanEquals": {
                    "sts:DurationSeconds": 3600
                },
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

使用標籤控制 CodeArtifact 資源的存取

IAM 使用者政策陳述式中的條件是您用來指定 CodeArtifact 動作所需資源許可的語法的一部分。在條件中使用標記是控制資源和請求的存取權限的方式之一。如需標記 CodeArtifact 資源的資訊，請參閱標記 資源。本主題討論的是標記型的存取控制。

設計 IAM 政策時，您可能會透過授予對特定資源的存取來設定精細許可。隨著您管理的資源數量增加，此任務變得越來越困難。標記資源並在政策陳述式條件中使用標籤，可讓此任務更輕鬆。您可以對具有特定標籤的任何資源大量授予存取。然後，您會在建立期間或之後，對相關資源重複套用此標籤。

可以將標記連接到資源或在請求中將標記傳遞至支援標記的服務。在 CodeArtifact 中，資源可以有標籤，而某些動作可以包含標籤。在建立 IAM 政策時，可使用標記條件鍵來控制以下項目：

- 哪些使用者可以根據網域或儲存庫資源已有的標籤，對網域或儲存庫資源執行動作。
- 可在動作請求中傳遞的標籤。
- 請求中是否可使用特定的標籤鍵。

關於標籤條件索引鍵的完整語法和語義，請參閱 IAM 使用者指南中的使用標籤控制存取。

> **Important**
>
> 在資源上使用標籤來限制動作時，標籤必須位於動作操作所在的資源上。例如，若要拒絕具有標籤的`DescribeRepository`許可，標籤必須位於每個儲存庫上，而非網域上。AWS CodeArtifact 許可參考 如需 CodeArtifact 中的動作清單及其操作的資源，請參閱 。

標籤型存取控制範例

下列範例示範如何在 CodeArtifact 使用者的政策中指定標籤條件。

Example 1：根據請求中的標籤限制動作

`AWSCodeArtifactAdminAccess` 受管使用者政策可讓使用者不受限制地對任何資源執行 CodeArtifact 動作。

除非請求包含特定標籤，否則下列政策會限制此能力，並拒絕未經授權的使用者建立儲存庫的許可。為此，如果請求未指定名為 的標籤`costcenter`，且具有其中一個值1或 ，則會拒絕該`CreateRepository`動作2。除了受管使用者政策之外，客戶的管理員必須將此 IAM 政策連接到未授權的 IAM 使用者。

```
{
  "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：根據資源標籤限制動作

AWSCodeArtifactAdminAccess 受管使用者政策可讓使用者不受限制地對任何資源執行 CodeArtifact 動作。

下列政策會限制此能力，並拒絕未經授權的使用者對指定網域中的儲存庫執行動作的許可。在作法上，如果資源有名為 Key1 的標記，且值為 Value1 或 Value2，則拒絕某些動作。(aws:ResourceTag 條件索引鍵用於依據資源上的標籤來控制資源的存取。) 除了受管使用者政策之外，客戶的管理員必須將此 IAM 政策連接到未授權的 IAM 使用者。

```
{
  "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：根據資源標籤允許動作

下列政策會授予使用者許可，以對 CodeArtifact 中的儲存庫和套件執行動作，並取得相關資訊。

若要這樣做，如果儲存庫具有名為 Key1且值為 的標籤，則允許特定動作Value1。(aws:RequestTag 條件索引鍵用來控制哪些標籤可在 IAM 請求中傳遞。) aws:TagKeys 條件可確保標籤索引鍵區分大小寫。此政策適用於未連接 AWSCodeArtifactAdminAccess 受管使用者政策的 IAM 使用者。受管政策可讓使用者不受限制地對任何資源執行 CodeArtifact 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact:ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
```

```
      }
    }
  ]
}
```

Example 4：根據請求中的標籤允許動作

下列政策授予使用者在 CodeArtifact 中指定網域中建立儲存庫的許可。

若要這樣做，如果請求中的建立資源 API 指定名為 的標籤Key1，其值為 ，則允許CreateRepository和 TagResource動作Value1。(aws:RequestTag 條件索引鍵用來控制哪些標籤可在 IAM 請求中傳遞。) aws:TagKeys 條件可確保標籤索引鍵區分大小寫。此政策適用於未連接 AWSCodeArtifactAdminAccess 受管使用者政策的 IAM 使用者。受管政策可讓使用者不受限制地對任何資源執行 CodeArtifact 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:CreateRepository",
        "codeartifact:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

AWS CodeArtifact 許可參考

AWS CodeArtifact 資源和操作

在 AWS CodeArtifact 中，主要資源是網域。在政策中，您使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。儲存庫也是 資源，且具有與其相關聯的 ARNs。如需詳細資訊，請參閱《》中的 Amazon Resource Name (ARNs)Amazon Web Services 一般參考。

資源類型	ARN 格式
網域	arn:aws:codeartifact: *region-ID* :*account-ID* :domain/*my_domain*
儲存庫	arn:aws:codeartifact: *region-ID* :*account-ID* :repository/ *my_domain* /*my_repo*
套件群組	arn:aws:codeartifact: *region-ID* :*account-ID* :package-group/ *my_domain* /*encoded_package_group_pattern*
具有命名空間的套件	arn:aws:codeartifact: *region-ID* :*account-ID* :package/ *my_domain* /*my_repo*/*package-format* /*namespace* /*my_package*
不含命名空間的套件	arn:aws:codeartifact: *region-ID* :*account-ID* :package/ *my_domain* /*my_repo*/*package-format* //*my_package*
所有 CodeArtifact 資源	arn:aws:codeartifact:*
指定 AWS 區域中指定帳戶擁有的所有 CodeArtifact 資源	arn:aws:codeartifact: *region-ID* :*account-ID* :*

您指定的資源 ARN 取決於您要控制存取的動作。

您可以使用其 ARN，在陳述式中指出特定網域 (*myDomain*)，如下所示。

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

您可以使用其 ARN，在陳述式中指出特定儲存庫 (*myRepo*)，如下所示。

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

若要在單一陳述式中指定多項資源，請使用逗號分隔他們的 ARN。下列陳述式適用於特定網域中的所有套件和儲存庫。

```
"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**
>
> 許多 AWS 服務會將冒號 (:) 或正斜線 (/) 視為 ARNs中的相同字元。不過，CodeArtifact 在資源模式和規則中使用完全相符的項目。在建立事件模式時，請務必使用正確的字元，使這些字元符合資源中的 ARN 語法。

AWS CodeArtifact API 操作和許可

當您設定存取控制和撰寫可連接到 IAM 身分的許可政策 （身分型政策） 時，您可以使用下表做為參考。

您可以在 AWS CodeArtifact 政策中使用 AWS全局條件索引鍵來表示條件。如需清單，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

您可以在政策的 `Action` 欄位中指定動作。若要指定動作，請使用 `codeartifact:` 字首，後面接著 API 操作名稱 (例如 `codeartifact:CreateDomain` 和 `codeartifact:AssociateExternalConnection`)。若要在單一陳述式中指定多個動作，請用逗號加以分隔 (例如 `"Action": [ "codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection" ]`)。

使用萬用字元

您可以使用或不使用萬用字元 (*)，指定 ARN 做為政策之 `Resource` 欄位中的資源值。您可以使用萬用字元指定多個動作或資源。例如， `codeartifact:*`會指定所有 CodeArtifact 動作，並`codeartifact:Describe*`指定以字詞 開頭的所有 CodeArtifact 動作`Describe`。

套件群組 ARNs

> **Note**
>
> 本節說明套件群組 ARNs和模式編碼如何資訊化。建議您從主控台複製 ARNs，或使用 `DescribePackageGroup` API 擷取 ARNs，而不是編碼模式和建構 ARNs。

IAM 政策使用萬用字元 *來比對多個 IAM 動作或多個資源。套件群組模式也使用 * 字元。為了更輕鬆地撰寫符合單一套件群組的 IAM 政策，套件群組 ARN 格式會使用套件群組模式的編碼版本。

具體而言，套件群組 ARN 格式如下：

```
arn:aws:codeartifact:region:account-ID:package-
group/my_domain/encoded_package_group_pattern
```

其中編碼的套件群組模式是套件群組模式，某些特殊字元會取代為其百分比編碼值。下列清單包含字元及其對應的百分比編碼值：

- * : %2a
- $: %24
- % : %25

例如，網域根套件群組 (/*) 的 ARN 會是：

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

請注意，清單中未包含的字元無法編碼，且 ARNs 區分大小寫，因此*必須編碼為 %2a，而非 %2A。

疑難排解 AWS CodeArtifact 身分和存取

使用以下資訊來協助您診斷和修正使用 CodeArtifact 和 IAM 時可能遇到的常見問題。

主題

- 我無權在 CodeArtifact 中執行動作
- 我想要允許 以外的人員 AWS 帳戶 存取我的 CodeArtifact 資源

我無權在 CodeArtifact 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 codeartifact:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codeartifact:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 codeartifact:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 以外的人員 AWS 帳戶 存取我的 CodeArtifact 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 CodeArtifact 是否支援這些功能，請參閱 How AWS CodeArtifact 可與 IAM 搭配使用。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 中提供存取權給 IAM 使用者。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有的 。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

使用 Amazon VPC 端點

您可以設定 CodeArtifact 使用界面虛擬私有雲端 (VPC) 端點來改善 VPC 的安全性。

VPC 端點使用 AWS PrivateLink的一項服務可讓您透過私有 IP 地址存取 CodeArtifact APIs。 AWS PrivateLink 會限制 VPC 和 CodeArtifact 與 AWS 網路之間的所有網路流量。當您使用介面 VPC 端點時，您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 VPC 端點。

Important

- VPC 端點不支援跨AWS 區域請求。請務必在計劃對 CodeArtifact 發出 API AWS 呼叫的相同區域中建立端點。
- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組，必須允許從 VPC 的私有子網路，透過 443 埠傳入的連線。

主題

- 建立 CodeArtifact 的 VPC 端點
- 建立 Amazon S3 閘道端點
- 從 VPC 使用 CodeArtifact
- 為 CodeArtifact 建立 VPC 端點政策

建立 CodeArtifact 的 VPC 端點

若要為 CodeArtifact 建立虛擬私有雲端 (VPC) 端點，請使用 Amazon EC2 `create-vpc-endpoint` AWS CLI 命令。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的界面 VPC 端點 (AWS PrivateLink)。

需要兩個 VPC 端點，以便對 CodeArtifact 的所有請求都位於 AWS 網路中。第一個端點用於呼叫 CodeArtifact APIs（例如 `GetAuthorizationToken`和 `CreateRepository`)。

```
com.amazonaws.region.codeartifact.api
```

第二個端點用於使用套件管理員和建置工具（例如 npm 和 Gradle) 存取 CodeArtifact 儲存庫。

```
com.amazonaws.region.codeartifact.repositories
```

下列命令會建立端點以存取 CodeArtifact 儲存庫。

```
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
```

下列命令會建立端點來存取套件管理員和建置工具。

```
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**
>
> 建立`codeartifact.repositories`端點時，您必須使用 `--private-dns-enabled`選項建立私有 DNS 主機名稱。如果您在建立`codeartifact.repositories`端點時無法或不想建立私有 DNS 主機名稱，則必須遵循額外的組態步驟，從 VPC 使用套件管理員搭配 CodeArtifact。如需更多資訊，請參閱在沒有私有 DNS 的情況下使用`codeartifact.repositories`端點。

建立 VPC 端點之後，您可能需要使用安全群組規則執行更多組態，才能搭配 CodeArtifact 使用端點。如需 Amazon VPC 中安全群組的詳細資訊，請參閱安全群組。

如果您在連線至 CodeArtifact 時遇到問題，您可以使用 VPC Reachability Analyzer 工具來偵錯問題。如需詳細資訊，請參閱什麼是 VPC Reachability Analyzer？

建立 Amazon S3 閘道端點

CodeArtifact 使用 Amazon Simple Storage Service (Amazon S3) 來存放套件資產。若要從 CodeArtifact 提取套件，您必須為 Amazon S3 建立閘道端點。當您的建置或部署程序從 CodeArtifact 下載套件時，必須存取 CodeArtifact 以取得套件中繼資料，以及 Amazon S3 下載套件資產（例如 Maven `.jar` 檔案）。

> **Note**
>
> 使用 Python 或 Swift 套件格式時，不需要 Amazon S3 端點。

若要為 CodeArtifact 建立 Amazon S3 閘道端點，請使用 Amazon EC2 `create-vpc-endpoint` AWS CLI 命令。建立端點時，您必須選取 VPC 的路由表。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的閘道 VPC 端點。

下列命令會建立 Amazon S3 端點。

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
  --route-table-ids routetableid
```

AWS CodeArtifact 的最低 Amazon S3 儲存貯體許可

Amazon S3 閘道端點會使用 IAM 政策文件來限制服務的存取權限。若要僅允許 CodeArtifact 的最低 Amazon S3 儲存貯體許可，請限制 CodeArtifact 在您為端點建立 IAM 政策文件時所使用的 Amazon S3 儲存貯體存取權。

下表說明政策中應該參考的 Amazon S3 儲存貯體，以允許存取每個區域的 CodeArtifact。

區域	Amazon S3 儲存貯體 ARN
us-east-1	arn：aws：s3：：assets-193858265520-us-east-1
us-east-2	arn：aws：s3：：assets-250872398865-us-east-2
us-west-2	arn：aws：s3：：assets-787052242323-us-west-2
eu-west-1	arn：aws：s3：：assets-438097961670-eu-west-1
eu-west-2	arn：aws：s3：：assets-247805302724-eu-west-2

區域	Amazon S3 儲存貯體 ARN
eu-west-3	arn：aws：s3：：assets-762466490029-eu-west-3
eu-north-1	arn：aws：s3：：assets-611884512288-eu-north-1
eu-south-1	arn：aws：s3：：assets-484130244270-eu-south-1
eu-central-1	arn：aws：s3：：assets-769407342218-eu-central-1
ap-northeast-1	arn：aws：s3：：assets-660291247815-ap-northeast-1
ap-southeast-1	arn：aws：s3：：assets-421485864821-ap-southeast-1
ap-southeast-2	arn：aws：s3：：assets-860415559748-ap-southeast-2
ap-south-1	arn：aws：s3：：assets-681137435769-ap-south-1

您可以使用 `aws codeartifact describe-domain`命令來擷取 CodeArtifact 網域所使用的 Amazon S3 儲存貯體。

```
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"
  }
}
```

範例

下列範例說明如何提供us-east-1區域中 CodeArtifact 操作所需的 Amazon S3 儲存貯體存取權。對於其他區域，請根據上表，使用您區域的正確許可 ARN 更新Resource項目。

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]
    }
  ]
}
```

從 VPC 使用 CodeArtifact

如果您無法或不想在 中建立的 com.amazonaws.*region*.codeartifact.repositories VPC 端點上啟用私有 DNS建立 CodeArtifact 的 VPC 端點，您必須為儲存庫端點使用不同的組態，才能從 VPC 使用 CodeArtifact。如果com.amazonaws.*region*.codeartifact.repositories端點未啟用私有 DNS，請依照 中的指示在沒有私有 DNS 的情況下使用codeartifact.repositories端點設定 CodeArtifact。

在沒有私有 DNS 的情況下使用**codeartifact.repositories**端點

如果您無法或不想在 中建立的 com.amazonaws.*region*.codeartifact.repositories VPC 端點上啟用私有 DNS建立 CodeArtifact 的 VPC 端點，則必須遵循這些指示，使用正確的 CodeArtifact URL 設定套件管理員。

1. 執行下列命令來尋找要用來覆寫主機名稱的 VPC 端點。

```
$ aws ec2 describe-vpc-endpoints --filters Name=service-
name,Values=com.amazonaws.region.codeartifact.repositories \
  --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
```

輸出看起來如下。

```
[
  [
    "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"
  ]
]
```

2. 更新 VPC 端點路徑以包含套件格式、您的 CodeArtifact 網域名稱和 CodeArtifact 儲存庫名稱。請參閱以下範例。

```
https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-
west-2.vpce.amazonaws.com/format/d/domain_name-domain_owner/repo_name
```

從範例端點取代下列欄位。

- *##*：將 取代為有效的 CodeArtifact 套件格式，例如 npm或 pypi。
- *domain_name*：將 取代為 CodeArtifact 網域，其中包含託管套件的 CodeArtifact 儲存庫。
- *domain_owner*：將 取代為 CodeArtifact 網域擁有者的 ID，例如 111122223333。
- *repo_name*：將 取代為託管套件的 CodeArtifact 儲存庫。

下列 URL 是範例 npm 儲存庫端點。

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. 將套件管理員設定為使用上一個步驟中更新的 VPC 端點。您必須在不使用 CodeArtifact `login`命令的情況下設定套件管理員。如需每個套件格式的組態指示，請參閱下列文件。

- npm： 不使用登入命令設定 npm
- nuget：在沒有登入命令的情況下設定 nuget 或 dotnet
- pip： 設定不含登入命令的 pip

- 雙身： 使用 CodeArtifact 設定和使用雙身
- Gradle： 搭配 Gradle 使用 CodeArtifact
- mvn： 將 CodeArtifact 與 mvn 搭配使用

為 CodeArtifact 建立 VPC 端點政策

若要為 CodeArtifact 建立 VPC 端點政策，請指定下列項目：

- 可執行動作的主體。
- 可執行的動作。
- 可對其執行動作的資源。

下列範例政策指定帳戶 123456789012 中的主體可以呼叫 `GetAuthorizationToken` API 並從 CodeArtifact 儲存庫擷取套件。

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
         "AWS": "arn:aws:iam::123456789012:root"
       }
    }
  ]
}
```

使用 建立 CodeArtifact 資源 AWS CloudFormation

CodeArtifact 已與 整合 AWS CloudFormation，這項服務可協助您建立和設定 AWS 資源的模型，讓您可減少建立和管理資源和基礎設施的時間。您可以建立範本來描述您想要的所有 AWS 資源，並 AWS CloudFormation 負責為您佈建和設定這些資源。

使用 時 AWS CloudFormation，您可以重複使用範本來持續且重複地設定 CodeArtifact 資源。只需描述您的資源一次，然後在多個帳戶和 AWS 區域中逐一佈建相同的資源。

CodeArtifact 和 AWS CloudFormation 範本

若要佈建和設定 CodeArtifact 和相關服務的資源，您必須了解 AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation 設計工具來協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的什麼是 AWS CloudFormation 設計工具？。

CodeArtifact 支援在其中建立網域、儲存庫和套件群組 AWS CloudFormation。如需詳細資訊，包括 JSON 和 YAML 範本的範例，請參閱AWS CloudFormation 《 使用者指南》中的下列主題：

- AWS::CodeArtifact::Domain
- AWS::CodeArtifact::Repository
- AWS::CodeArtifact::PackageGroup

防止刪除 CodeArtifact 資源

CodeArtifact 儲存庫包含重要的應用程式相依性，如果遺失，可能無法輕鬆重新建立。若要保護 CodeArtifact 資源免於在透過 CloudFormation 管理 CodeArtifact 資源時意外刪除，請在所有網域`DeletionPolicy`和儲存庫`Retain`上包含值為 的 和 `UpdateRetainPolicy` 屬性。如果從堆疊範本中移除資源，或意外刪除整個堆疊，這將防止刪除。下列 YAML 程式碼片段顯示具有這些屬性的基本網域和儲存庫：

```
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
```

如需這些屬性的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的 DeletionPolicy 和 UpdateReplacePolicy。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation 命令列界面使用者指南

故障診斷 AWS CodeArtifact

以下資訊可協助您對 CodeArtifact 的常見問題進行疑難排解。

如需特定格式問題的疑難排解資訊，請參閱下列主題：

- Maven 疑難排解
- Swift 疑難排解

我無法檢視通知

問題：您進入開發人員工具主控台後，在 Settings (設定) 底下選擇 Notifications (通知) 時，出現許可錯誤。

可能的修正：雖然通知是開發人員工具主控台的功能，但 CodeArtifact 目前不支援通知。CodeArtifact 的受管政策都未包含允許使用者檢視或管理通知的許可。如果您在開發人員工具主控台中使用其他 服務，且這些服務支援通知，則這些服務的受管政策會包含檢視和管理這些服務通知所需的許可。

標記 資源

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。每個 AWS 標籤有兩個部分：

- 標籤鍵 (例如，CostCenter、Environment、Project 或 Secret)。標籤鍵會區分大小寫。
- 一個名為標籤值 (例如，111122223333、Production 或團隊名稱) 的選用欄位。忽略標籤值基本上等同於使用空字串。與標籤鍵相同，標籤值會區分大小寫。

這些合稱為鍵值組。

標籤可協助您識別和組織 AWS 資源。許多 AWS 服務支援標籤，因此您可以對來自不同服務的資源指派相同的標籤，以指出資源是相關的。例如，您可以將相同的標籤指派給指派給 AWS CodeBuild 專案的儲存庫。

如需使用標籤的秘訣和最佳實務，請參閱標記 AWS 資源的最佳實務白皮書。

您可以在 CodeArtifact 中標記下列資源類型：

- 在 CodeArtifact 中標記儲存庫
- 在 CodeArtifact 中標記網域

您可以使用 主控台 AWS CLI、CodeArtifact APIs或 AWS SDKs來：

- 當您建立標籤時，將標籤新增至網域或儲存庫*。
- 新增、管理和移除網域或儲存庫的標籤。

* 在主控台中建立網域或儲存庫時，您無法將標籤新增至該網域或儲存庫。

除了使用標籤識別、組織和追蹤您的資源之外，您還可以在 IAM 政策中使用標籤，以協助控制誰可以檢視資源並與之互動。如需以標籤為基礎的存取政策範例，請參閱使用標籤控制 CodeArtifact 資源的存取。

CodeArtifact 成本分配與標籤

您可以使用標籤在 CodeArtifact 中配置儲存體和請求成本。

在 CodeArtifact 中分配資料儲存成本

資料儲存成本與網域相關聯，因此若要配置 CodeArtifact 儲存成本，您可以使用套用至網域的任何標籤。如需將標籤新增至網域的詳細資訊，請參閱 在 CodeArtifact 中標記網域。

在 CodeArtifact 中分配請求成本

大多數的請求用量都與儲存庫相關，因此，若要配置 CodeArtifact 請求成本，您可以使用任何套用至儲存庫的標籤。如需將標籤新增至儲存庫的詳細資訊，請參閱 在 CodeArtifact 中標記儲存庫。

有些請求類型與網域而非儲存庫相關聯，因此與請求相關的請求用量和成本會配置給網域上的標籤。判斷請求類型是否與網域或儲存庫相關聯的最佳方法是使用服務授權參考中AWS CodeArtifact 定義的動作資料表。在動作欄中尋找請求類型，並查看對應資源類型欄中的值。如果資源類型是網域，則該類型的請求將向網域收費。如果資源類型是儲存庫或套件，則該類型的請求將向儲存庫收費。有些動作會顯示這兩種資源類型，對於那些動作，計費的資源取決於請求中傳遞的值。

AWS CodeArtifact 中的配額

下表說明 CodeArtifact 中的資源配額。若要檢視資源配額以及 CodeArtifact 的服務端點清單，請參閱中的AWS 服務配額Amazon Web Services 一般參考。

您可以請求提高下列 CodeArtifact 資源配額的服務配額。 CodeArtifact 如需請求提高服務配額的詳細資訊，請參閱 AWS Service Quotas。

名稱	預設	可調整	描述
資產檔案大小	所有受支援的區域：5 GB	是	每個資產的檔案大小上限。
每個套件版本的資產	每個受支援的區域：150	否	每個套件版本的資產數目上限。
每秒 CopyPackageVersions 請求數	每個受支援的區域：5	是	每秒可對 CopyPackageVersions 進行的呼叫數上限。
每個儲存庫的直接上游	每個受支援的區域：10	否	每個儲存庫的直接上游儲存庫數量上限。
每個 AWS 帳戶的網域數	每個受支援的區域：10	是	每個 AWS 帳戶可建立的網域數量上限。
每秒 GetAuthorizationToken 請求數	每個受支援的區域：40	是	每秒擷取的授權字符數目上限。
每秒 GetPackageVersionAsset 請求數	每個受支援的區域：50	是	每秒可對 GetPackageVersionAsset 進行的呼叫數上限。
ListPackageVersionAssets 每秒請求數	每個受支援的區域：200	是	每秒可對 ListPackageVersionAssets 進行的最大呼叫數。

名稱	預設	可調整	描述
ListPackageVersions 每秒請求數	每個受支援的區域：200	是	每秒可對 ListPackageVersions 進行的最大呼叫數。
每秒 ListPackages 請求數	每個受支援的區域：200	是	每秒可對 ListPackages 進行的最大呼叫數。
每秒 PublishPackageVersion 請求數	每個受支援的區域：10	是	每秒可對 PublishPackageVersion 進行的最大呼叫數。
每秒從單一 AWS 帳戶讀取請求	每個支援的區域：800	是	每秒來自一個 AWS 帳戶的讀取請求數目上限。
每個網域的儲存庫	每個受支援的區域：1,000	是	每個網域可建立的儲存庫數量上限。
使用單一身分驗證字符的每秒請求數	每個支援的區域：1，200	否	使用單一身分驗證字符的每秒請求數上限。
每個 IP 地址沒有身分驗證字符的請求	每個支援的區域：600	否	來自單一 IP 地址且沒有身分驗證字符的每秒請求數上限。
搜尋的上游儲存庫	每個受支援的區域：25	否	解析套件時搜尋的上游儲存庫數量上限。
每秒從單一 AWS 帳戶寫入請求	每個受支援的區域：100	是	每秒來自一個 AWS 帳戶的寫入請求數目上限。

Note

一般而言，對 CodeArtifact 提出的每個讀取請求都算作一個請求計入配額。不過，對於 Ruby 套件格式，對 `/api/v1/dependencies`操作的單一讀取請求可以請求多個套件的資料。

例如，請求可能看起來像 `https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1,gem2.gem3`。在此範例中，請求會計算為針對配額的三個請求。
請注意，多個請求僅適用於服務配額，不適用於計費。在此範例中，您只需為一個請求付費，但它算作對服務配額的三個請求。

AWS CodeArtifact 使用者指南文件歷史記錄

下表說明 CodeArtifact 文件的重要變更。

變更	描述	日期
新增設定和使用 Cargo with CodeArtifact 的文件	CodeArtifact 現在支援貨運箱。新增了有關設定貨運以使用 CodeArtifact 儲存庫的指引文件。如需詳細資訊，請參閱搭配貨運使用 CodeArtifact 。	2024 年 6 月 20 日
新增設定和使用 Ruby 搭配 CodeArtifact 的文件	CodeArtifact 現在支援 Ruby Gem 套件。新增了有關設定 Ruby 套件管理員使用 CodeArtifact 儲存庫的指引文件。如需詳細資訊，請參閱搭配 Ruby 使用 CodeArtifact 。	2024 年 4 月 30 日
新增使用客戶受管金鑰建立網域的 AWS KMS 金鑰政策範例	新增了可用於建立客戶受管 KMS 金鑰以加密 CodeArtifact 網域中資產的金鑰政策範例。如需詳細資訊，請參閱範例 AWS KMS 金鑰政策。	2024 年 4 月 18 日
新增支援啟動套件群組的文件。	新增有關在 CodeArtifact 中管理和使用套件群組的文件。如需詳細資訊，請參閱在 CodeArtifact 中使用套件群組。	2024 年 3 月 21 日
已將其他有效的套件管理員新增至 aws codeartifact 登入命令的文件。	已將 `dotnet`、 `nuget`和 `swift` 新增至要與 `aws codeartifact login` 命令搭配使用的有效套件管理員清單。如需詳細資訊，請參閱	2024 年 2 月 18 日

	AWS CodeArtifact 身分驗證和字符。	
在 Swift 疑難排解文件中新增了有關在 CI 機器上掛起 Xcode 的項目	已新增因密碼的金鑰鏈提示而可能導致 Xcode 在 CI 機器上懸置的問題的相關資訊，包括解決方案。如需詳細資訊，請參閱由於密碼的金鑰鏈提示，Xcode 掛在 CI 機器上。	2024 年 2 月 6 日
新增有關使用 npm 8.x 或更高版本對慢速 npm 套件安裝時間進行故障診斷的資訊	新增有關從 CodeArtifact 處理慢速 npm 套件安裝時間的資訊，這可能會導致建置時間變慢。如需詳細資訊，請參閱使用 npm 8.x 或更高版本對慢速安裝進行故障診斷。	2023 年 12 月 29 日
更新 CodeArtifact 中 Python 套件資產和中繼資料行為的相關資訊	更新 CodeArtifact 儲存庫如何保留和重新整理 Python 套件版本資產和中繼資料的相關資訊。如需詳細資訊，請參閱從上游和外部連線請求 Python 套件。	2023 年 12 月 14 日
有關監控 CodeArtifact 的重組文件	重新整理有關監控 CodeArtifact 事件的資訊，並新增有關使用 Amazon CloudWatch 指標檢視 CodeArtifact 請求的資訊。如需詳細資訊，請參閱監控 CodeArtifact。	2023 年 12 月 14 日
新增使用 管理 CodeArtifact 資源的詳細資訊 AWS CloudFormation	新增了有關使用 CloudFormation 管理 CodeArtifact 資源的文件參考和連結，包括有關防止使用 CloudFormation 管理 CodeArtifact 資源刪除的章節。如需詳細資訊，請參閱防止刪除 CodeArtifact 資源。	2023 年 12 月 7 日

新增詳細說明 CodeArtifact 支援 AWS KMS 外部金鑰存放區 (XKS) 的文件	新增章節，其中包含 CodeArtifact 支援 KMS 金鑰的相關資訊，包括搭配 CodeArtifact 使用 XKS 金鑰。如需詳細資訊，請參閱CodeArtifact 支援的 AWS KMS 金鑰類型。	2023 年 10 月 31 日
更新現有並新增故障診斷文件	新增 Maven 故障診斷主題，並在一般故障診斷主題中包含 Swift 和 Maven 故障診斷文件的連結。如需詳細資訊，請參閱故障診斷 AWS CodeArtifact。	2023 年 9 月 28 日
更新文件以包含 Swift Package Manager 發佈命令	Swift 5.9 引入了`swift package-registry publish`命令，以建立 Swift 套件並將其發佈至套件儲存庫。已更新 Swift 文件，以包含使用該命令的指示。如需詳細資訊，請參閱搭配 Swift 使用 CodeArtifact 。	2023 年 9 月 25 日
新增使用 Swift 設定 CodeArtifact 的文件	CodeArtifact 現在支援 Swift 套件。新增了有關設定 Swift 使用 CodeArtifact 儲存庫的指引文件。如需詳細資訊，請參閱搭配 Swift 使用 CodeArtifact 。	2023 年 9 月 20 日
新增 CodeArtifact 如何處理偏擺 Python 套件版本的指引	已新增文件，說明如何判斷 Python 套件版本是否中斷、CodeArtifact 如何處理中斷套件版本，以及常見問題的答案。如需詳細資訊，請參閱Yanked 套件版本。	2023 年 8 月 2 日

修正 Yarn 文件中的不正確命令列命令	修正擷取 CodeArtifact 授權字符，並將其存放在 Yarn 文件環境變數中的不正確命令列命令。	2023 年 7 月 20 日
Python 文件的次要新增和小錯誤修正	在各自的文件中新增 pip 和 Twine 資訊，並修正將 `codeartifact login` 命令與 Twine 搭配使用時會發生的情況。如需詳細資訊，請參閱搭配 CodeArtifact 設定和使用 pip 和 使用 CodeArtifact 設定和使用雙身。	2023 年 7 月 14 日
修正 CodeBuild 文件中不正確的 dotnet 命令	更正在 CodeBuild 中使用 NuGet 套件文件中的`dotnet add package`命令。	2023 年 7 月 13 日
Updated AWS CodeArtifact 和 AWS Identity and Access Management 文件	CodeArtifact 文件中的 IAM 進行了大修，以增加與其他 AWS 服務的文件的清晰度和一致性。請參閱 Identity and Access Management for AWS CodeArtifact。	2023 年 5 月 24 日
新增有關 yanked Python 套件版本的資訊	新增 CodeArtifact 如何保留偏擺 Python 套件版本中繼資料的相關資訊，如需詳細資訊，請參閱 Yanked 套件版本。	2023 年 4 月 11 日
新增 Clojure 支援的相關資訊	新增 Clojure 支援的相關資訊，包括管理 Clojure 專案的相依性。如需詳細資訊，請參閱搭配 deps.edn 使用 CodeArtifact。	2023 年 3 月 21 日

新增一般套件發佈的相關資訊	新增一般套件以及如何使用發佈和下載套件內容的相關資訊 AWS CLI。如需詳細資訊，請參閱搭配一般套件使用 CodeArtifact 、發佈和使用一般套件及一般套件支援的命令。	2023 年 3 月 10 日
新增發佈資產大小限制的相關資訊	已將 區段新增至套件發佈，以說明要發佈的資產大小限制。	2022 年 6 月 21 日
重構外部連線文件	移動外部連線文件並重新組織，以專注於使用者的最終目標，也就是將其 CodeArtifact 儲存庫連線至公有套件儲存庫。也針對實現該目標的不同方法新增了更多指引和資訊。如需詳細資訊，請參閱將 CodeArtifact 儲存庫連線至公有儲存庫。	2022 年 5 月 9 日
已更新 Amazon CloudWatch Events 的 CodeArtifact 事件資訊	已將詳細資訊新增至 `account` 欄位，並新增 `repositoryAdministrator` 欄位。如需詳細資訊，請參閱CodeArtifact 事件格式和範例。	2022 年 3 月 7 日

新增從沒有私有 DNS 的 VPC 使用 CodeArtifact 的組態指示	如果您無法或不想在 `codeartifact.repositories` VPC 端點上啟用私有 DNS，您必須為儲存庫端點使用不同的組態，才能從 VPC 使用 CodeArtifact。如需更多資訊，請參閱在沒有私有 DNS 的情況下使用 `codeartifact.repositories` 端點。	2022 年 2 月 8 日
新增更新套件版本狀態的深入文件	已將更新套件版本狀態文件擴展至自己的主題。新增更新套件版本狀態的文件，包括必要的 IAM 許可、各種案例的範例 AWS CLI 命令，以及可能的錯誤。如需更多資訊，請參閱更新套件版本狀態。	2021 年 9 月 1 日
使用更深入的許可資訊更新複製套件版本文件	已新增呼叫 `aws codeartifact copy-package-versions` 命令所需的 IAM 和資源型政策許可的詳細資訊，以將套件版本從一個儲存庫複製到 CodeArtifact 中相同網域中的另一個儲存庫。除了詳細資訊之外，現在還有來源和目的地儲存庫所需的資源型政策範例。如需更多資訊，請參閱複製套件所需的 IAM 許可。	2021 年 8 月 25 日

更新在 IntelliJ IDEA 中執行 Gradle 建置的文件	更新在 IntelliJ IDEA 中執行 Gradle 建置的文件，其中包含設定 Gradle 從 CodeArtifact 擷取外掛程式的步驟。也新增了 選項，為每次新執行建立新的 CodeArtifact 授權字符，並對 進行內嵌呼叫`aws codeartifact get-authorization-token` 。如需更多資訊，請參閱在 IntelliJ IDEA 中執行 Gradle 建置。	2021 年 8 月 23 日
新增設定和使用 Yarn 搭配 AWS CodeArtifact 的文件	新增了使用 CodeArtifact 設定和使用 Yarn 1.X 和 Yarn 2.X 來管理 npm 套件的文件。如需更多資訊，請參閱設定和使用 Yarn 搭配 CodeArtifact。	2021 年 7 月 30 日
AWS CodeArtifact 現在支援 NuGet 套件	CodeArtifact 使用者現在可以發佈和使用 NuGet 套件。新增了有關設定和使用 Visual Studio 和 NuGet 命令列工具的文件，例如 `nuget`和 `dotnet`搭配 CodeArtifact 儲存庫。如需更多資訊，請參閱搭配 NuGet 使用 CodeArtifact 。	2020 年 11 月 19 日
在 AWS CodeArtifact 中標記資源	在 AWS CodeArtifact 中新增了有關標記儲存庫和網域的文件。請參閱 標記 資源。	2020 年 10 月 30 日

CodeArtifact 現在支援 AWS CloudFormation	CodeArtifact 使用者現在可以使用 AWS CloudFormation 範本來建立 CodeArtifact 儲存庫和網域。使用 建立 CodeArtifact 資源 AWS CloudFormation 如需詳細資訊，請參閱 和 以開始使用。	2020 年 10 月 8 日
新增建立 Amazon S3 閘道端點以搭配 Amazon VPC 使用 CodeArtifact 的相關資訊	新增使用 Amazon EC2 命令建立 Amazon S3 閘道端點的相關資訊。 Amazon EC2 AWS CLI 本文件也包含 CodeArtifact 與 Amazon VPC 環境搭配使用所需的特定許可的相關資訊。請參閱 建立 Amazon S3 閘道端點。	2020 年 8 月 12 日
使用 curl 發佈 Maven 成品，並發佈第三方 Maven 成品	新增 使用 curl 發佈和 的指引發佈第三方成品。	2020 年 8 月 10 日
一般可用性 (GA) 版本	CodeArtifact 使用者指南的初始版本。	2020 年 6 月 10 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。