CodeArtifact 使用者指南

CodeArtifact

CodeArtifact: CodeArtifact 使用者指南

Table of Contents

什麼是 AWS CodeArtifact？

AWS CodeArtifact 是一種安全、可高度擴充的受管理成品儲存庫服務，可協助組織儲存和共用軟體套件以進行應用程式開發。您可以使 CodeArtifact 用流行的構建工具和軟件包管理器，例如 NuGet CLI，Maven，搖籃，npm，紗線，pip 和麻線。 CodeArtifact 協助您減少管理自己的成品儲存系統的需求，或是擔心擴充其基礎架構的問題。您可以儲存在 CodeArtifact 儲存庫中的套件數量或總大小沒有限制。

您可以創建您的私有 CodeArtifact 存儲庫和外部，公共存儲庫，如 npmjs.com 或 Maven 中央之間的連接。 CodeArtifact 然後，當軟件包管理器請求時，將根據需求從公共存儲庫中獲取和存儲軟件包。這樣可以更方便地使用應用程式所使用的開放原始碼相依性，並有助於確保它們始終可用於建置和開發。您也可以將私人套件發佈到存 CodeArtifact 放庫。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。

如何 CodeArtifact 工作？

CodeArtifact 將軟體套件儲存在儲存庫中。儲存庫是多層的 — 單一儲存庫可以包含任何受支援類型的套件。每個 CodeArtifact 儲存庫都是單一 CodeArtifact 網域的成員。建議您針對組織使用一個或多個存放庫的一個生產網域。例如，您可以將每個存放庫用於不同的開發團隊。然後，您可以在開發團隊之間探索並共用儲存庫中的套件。

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存放庫。您也可以將開放原始碼套件匯入到儲存庫中，方法是使用外部連接至公共儲存庫，例如 npmjs、 NuGet 畫廊、Maven Central 或 PyPI。如需詳細資訊，請參閱 將 CodeArtifact 存儲庫 Connect 到公共存儲庫。

您可以讓一個儲存庫中的套件可供相同網域中的另一個存放庫使用。若要這麼做，請將一個存放庫設定為另一個存放庫的上游。上游存放庫可用的所有套件版本也可供下游存放庫使用。此外，下游存放庫可透過外部連線至公用存放庫使用的所有套裝軟體。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。

CodeArtifact 要求使用者透過服務進行驗證，才能發佈或使用套件版本。您必須使用憑據創建授權令牌來對 CodeArtifact 服務進行身份驗 AWS 證。 CodeArtifact 儲存庫中的套件無法公開使用。如需中的驗證和存取權的詳細資訊 CodeArtifact，請參閱AWS CodeArtifact 身份驗證和令牌。

AWS CodeArtifact 概念

以下是使用時需要了解的一些概念和術語 CodeArtifact。

主題

資產

資產是存儲在 CodeArtifact 與一個包版本，如 NPM `.tgz` 文件或 Maven POM 和 JAR 文件相關聯的單個文件。

網域

儲存庫會彙總到稱為網域的較高層級實體。所有套件資產和中繼資料都儲存在網域中，但它們會透過儲存庫使用。給定的包資產（例如 Maven JAR 文件）在每個域中存儲一次，無論它存在多少個存儲庫。網域中的所有資產和中繼資料都會使用 () 中 AWS Key Management Service 儲存的相同 AWS KMS key (KMS 金鑰AWS KMS) 加密。

每個存放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原則。使用這種方法，您可以決定哪些帳戶可以存取網域中的儲存庫，以及可以使用哪些公用存放庫做為套件的來源。

雖然一個組織可以有多個網域，但我們建議使用包含所有已發行成品的單一生產網域。如此一來，團隊就可以在整個組織中尋找並共用套件。

儲存庫

CodeArtifact 儲存庫包含一組套件版本，每個套件版本都會對應至一組資產。儲存庫是多層的 — 單一儲存庫可以包含任何受支援類型的套件。每個存儲庫公開端點，用於使用像 nuget CLI，npm CLI，Maven CLI（mvn）和 pip 等工具獲取和發布包。每個網域最多可以建立 1,000 個儲存庫。

套件

套件是解決相依性和安裝軟體所需的軟體套件和中繼資料。在中 CodeArtifact，套件包含套件名稱、選擇性命名空間 (例如 @types in) @types/node、一組套件版本，以及封裝層級中繼資料 (例如 npm 標籤)。

AWS CodeArtifact 支持 NPM，PyPI，Maven，斯威夫特 NuGet，紅寶石，通用包格式。

Package 群組

P@@ ackage 件群組可用來將組態套用至符合使用套件格式、套件命名空間和套件名稱的已定義模式的多個套件。您可以使用套件群組，更方便地為多個套件設定套件原始碼控制項。Package 來源控制項可用來封鎖或允許擷取或發佈新套件版本，以保護使用者免於遭受稱為相依性替代攻擊的惡意動作。

Package 命名空間

某些套件格式支援階層式套件名稱，將套件組織成邏輯群組，並協助避免名稱衝突。例如，npm 支持作用域。如需詳細資訊，請參閱 npm 範圍文件。npm 套件`@types/node`具有的範圍`@types`和名稱`node`。`@types`範圍內還有許多其他套件名稱。在中 CodeArtifact，範圍（「類型」）被稱為包命名空間和名稱（「節點」）被稱為包名稱。對於 Maven 包，包命名空間對應於 Maven 的組 ID。Maven 套件`org.apache.logging.log4j:log4j`具有的群組 ID (套件命名空間) `org.apache.logging.log4j` 和文件 ID (套件名稱)。`log4j`對於泛型套件，需要命名空間。某些套件格式 (例如 PyPI) 不支援階層式名稱，其概念類似於 npm 範圍或 Maven 群組 ID。如果沒有分組軟件包名稱的方法，避免名稱衝突可能會更加困難。

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`。不同套件格式的版本號碼格式和語意會有所不同。例如，npm 套件版本必須符合語意版本化規格。在中 CodeArtifact，套件版本包含版本識別碼、套件版本層級中繼資料以及一組資產。

Package 版本修訂

套件版本修訂是一個字串，可識別套件版本的一組特定資產和中繼資料。每次更新套件版本時，都會建立新的封裝版本修訂。例如，您可以發布 Python 包版本的源代碼發行歸檔（sdist），然後將包含編譯代碼的 Python 輪子添加到相同的版本。當您發佈操控盤時，會建立新的封裝版本修訂版本。

上游儲存庫

如果可以從下游存放庫的存放庫端點存取其中的套件版本，則其中一個存放庫是另一個存放庫的上游。這種方法從客戶端的角度有效地合併兩個存儲庫的內容。使用 CodeArtifact，您可以在兩個存儲庫之間創建上游關係。

我該如何開始使用 CodeArtifact？

建議您完成下列步驟：

1. CodeArtifact 通過閱讀了解更多信息AWS CodeArtifact 概念。
2. 按照中的 AWS 帳戶步驟設定您的 AWS CLI、和 IAM 使用者設定使用 AWS CodeArtifact。
3. 請 CodeArtifact 遵循中的指示來使用入門 CodeArtifact。

設定使用 AWS CodeArtifact

如果您已經註冊 Amazon Web Services（AWS），則可以 AWS CodeArtifact立即開始使用。您可以開啟 CodeArtifact 主控台，選擇 [建立網域和存放庫]，然後依照啟動精靈中的步驟建立您的第一個網域和存放庫。

如果您 AWS 尚未註冊，或需要建立第一個網域和儲存庫的協助，請完成下列工作以進行設定以使用 CodeArtifact：

主題

- 註冊成為 AWS
- 安裝或升級，然後設定 AWS CLI
- 佈建 IAM 使用者
- 安裝您的軟件包管理器或構建工具

註冊成為 AWS

當您註冊 Amazon Web Services (AWS) 時，只需支付您使用的服務和資源 (包括在內) 的費用 AWS CodeArtifact。

如果您已有 AWS 帳戶，請跳至下一個工作，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。

若要建立 AWS 帳戶

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊時 AWS 帳戶，會建立AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

安裝或升級，然後設定 AWS CLI

若要從本機開發電腦上的 AWS Command Line Interface (AWS CLI) 呼叫 CodeArtifact 命令，您必須安裝 AWS CLI.

如果您安 AWS CLI 裝的是較舊版本，則必須對其進行升級，才能使用這些 CodeArtifact命令。CodeArtifact 下列 AWS CLI 版本提供指令：

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`指令。

若要安裝和設定 AWS CLI

1. 按照安裝中 AWS CLI 的說明進行安裝或升級 AWS Command Line Interface。
2. 使用 AWS CLI配置命令配置，如下所示。

   ```
   aws configure
   ```

   出現提示時，請指定要搭配 AWS 使用之 IAM 使用者的存取金鑰和 AWS 秘密存取金鑰 CodeArtifact。當系統提示您輸入預設 AWS 區域 名稱時，請指定要在其中建立配管的「區域」(Region)，例如`us-east-2`。系統提示您輸入預設輸出格式時，請指定 `json`。

   > **Important**
   >
   > 配置時 AWS CLI，系統會提示您指定 AWS 區域。選擇「區域」和「端點」中列出的其中一個支援的區域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，此命令將顯示可用命 CodeArtifact 令的列表。

接下來，您可以建立 IAM 使用者並授與該使用者的存取權 CodeArtifact。如需詳細資訊，請參閱 佈建 IAM 使用者。

佈建 IAM 使用者

請依照下列指示準備要使用的 IAM 使用者 CodeArtifact。

若要提供 Aniam 使用者

1. 建立 IAM 使用者，或使用與 AWS 帳戶. 如需詳細資訊，請參閱 IAM 使用者指南中的建立 AWS IAM 使用者和 IAM 政策概觀。
2. 授與 IAM 使用者存取權限 CodeArtifact。

   - 選項 1：建立自訂 IAM 政策。使用自訂 IAM 政策，您可以提供所需的最低許可，並變更身份驗證 Token 的持續時間。如需詳細資訊和範例政策，請參閱以身分識別為基礎的原則範例 AWS CodeArtifact。
   - 選項 2：使用受AWSCodeArtifactAdminAccess AWS 管理的策略。下列程式碼片段顯示此政策的內容。

   > **Important**
   >
   > 此原則授予對所有 CodeArtifact API 的存取權。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"
                }
            }
        }
    ]
}
```

呼叫 CodeArtifact `GetAuthorizationToken` API 需要`sts:GetServiceBearerToken`權限。此 API 會傳回使用套件管理員 (例如`npm`或`pip`搭配) 時必須使用的權杖 CodeArtifact。若要將套件管理員與 CodeArtifact 儲存庫搭配使用，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要搭配使用的套件管理員或建置工具 CodeArtifact，請參閱安裝您的軟件包管理器或構建工具。

安裝您的軟件包管理器或構建工具

若要從中發行或使用套件 CodeArtifact，您必須使用套件管理員。每個封裝類型都有不同的封裝管理員。下列清單包含一些您可以搭配使用的套件管理員 CodeArtifact。如果您尚未安裝，請為您要使用的套件類型安裝套件管理員。

- 對於 NPM，請使用 npm CLI 或 PNPM。
- 對於 Maven，使用阿帕奇 Maven（`mvn`）或搖籃。
- 對於 Python，使用 pip 安裝軟件包並麻線發布軟件包。
- 對於 NuGet，使 T oolkit for Visual Studio 中的視覺工作室或核或網點網路 CLI。
- 對於一般套件，請使用AWS CLI或 SDK 來發佈和下載封裝內容。

後續步驟

接下來的步驟將取決於您要搭配使用的套件類型 CodeArtifact，以及 CodeArtifact 資源的狀態。

如果您是第一次 CodeArtifact 為自己、您的團隊或組織開始使用，請參閱下列文件以取得一般入門資訊以及建立所需資源的協助。

- 開始使用主控台
- AWS CLI 使用入門

如果您的資源已經建立，而且您已準備好將套件管理員設定為將套件推送至儲存庫或從 CodeArtifact 儲存庫安裝套件，請參閱與套件類型或套件管理員對應的文件。

- 使用 npm 的 CodeArtifact
- 使用CodeArtifact與蟒蛇
- CodeArtifact 與 Maven 一起使用
- 使用CodeArtifact取代為NuGet
- CodeArtifact 搭配泛型套件使用

入門 CodeArtifact

在本入門教學中，您可以使用 CodeArtifact 建立下列項目：

- 一個名為的域`my-domain`。
- 儲存庫`my-repo`包含於`my-domain`。
- 儲存庫`npm-store`包含於`my-domain`。所以此`npm-store`有一個外部連接到 npm 公共存儲庫。此連線是用來擷取 npm 套件到`my-repo`儲存庫。

開始本教學課程之前，我們建議您檢閱 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 價錢。

主題

- 先決條件
- 開始使用主控台
- AWS CLI 使用入門

先決條件

您可以使用AWS Management Console或AWS Command Line Interface(AWS CLI。若要遵循教學課程，您必須先完成以下必要條件：

- 完成「設定使用 AWS CodeArtifact」中的步驟。
- 安裝 npm 如需詳細資訊，請參閱下載和安裝 Node.js 和故宮在 npm 文檔中。

開始使用主控台

執行下列步驟，以開始使用 CodeArtifact 使用AWS Management Console。本指南`npm`套件管理員，如果您使用不同的套件管理員，您將需要修改下列一些步驟。

1. 前往登入AWS Management Console並開啟AWS CodeArtifact 主控台https://console.aws.amazon.com/codesuite/codeartifact/start。如需詳細資訊，請參閱 設定使用 AWS CodeArtifact。
2. 選擇 Create repository (建立儲存庫)。
3. In儲存庫名稱下一步**my-repo**。
4. (選擇性) 輸入儲存庫描述，輸入儲存庫的選用描述。
5. In公共上游儲存庫下一步npm-repo建立一組儲存庫npmjs那是從你的上游my-repo儲存庫。

   CodeArtifact 分配名稱npm-store到這個存儲庫為你。上游儲存庫中所有可用的套件npm-store也可供其下游存放庫使用，my-repo。
6. 選擇 Next (下一步)。
7. InAWS 帳戶，選擇這個 AWS 帳戶。
8. In網域名稱下一步**my-domain**。
9. 展開 Additional configuration (其他組態)。
10. 您必須使用AWS KMS key(KMS 金鑰) 可加密您網域中的所有資產。您可以使用AWS 受管金鑰或您管理的 KMS 金鑰：

    - 選擇AWS 受管金鑰如果你想使用默認AWS 受管金鑰。
    - 選擇客戶受管金鑰如果您希望使用您管理的 KMS 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 受管金鑰和客戶受管金鑰中的AWS Key Management Service開發人員指南。
11. 選擇 Next (下一步)。
12. InReview and create (檢閱和建立)，回顧什麼 CodeArtifact 正在為您創造。

    - Package顯示如何my-domain、my-repo，以及npm-store是相關的。
    - 步驟 1：建立儲存庫顯示相關詳細資訊my-repo和npm-store。
    - 步驟 2：選取網域顯示相關詳細資訊my-domain。

    當您準備好時，請選擇建立儲存庫。
13. 在「」my-repo頁面，選擇檢視連線指示(下一步)，然後選擇NPM。
14. 使用AWS CLI執行login指令顯示於使用此配置你的 npm 客戶端AWS CLI CodeArtifact命令。

```
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`，確保您AWS CLI已配置並且您的預設區域名稱設為您建立儲存庫時所在的區域，請參閱設定 AWS 命令列界面。

如需詳細資訊，請參閱配置和使用 npm CodeArtifact

15. 使用 npm CLI 來安裝 NPM 套件。例如，要安裝流行的 npm 軟件包`lodash`，使用下列命令。

```
npm install lodash
```

16. 返回 CodeArtifact 主控台。如果您的my-repo儲存庫是開放的，重新整理頁面。否則，在導覽窗格中，選擇儲存庫(下一步)，然後選擇my-repo。

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

> **Note**
>
> 安裝套件與擷取到存放庫之間可能會有延遲。

17. 避免AWS收費，刪除您在此教學課程中使用的資源：

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

    a. 從導覽窗格中，選擇儲存庫。

    b. 選擇npm-repo，選擇刪除，然後按照步驟刪除存放庫。

c. 選擇my-repo，選擇刪除，然後按照步驟刪除存放庫。

d. 從導覽窗格中，選擇網域。

e. 選擇My 網域，選擇刪除，然後依照步驟刪除網域。

AWS CLI 使用入門

執行下列步驟，以開始使用 CodeArtifact 使用AWS Command Line Interface(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npm套件管理員，如果您使用不同的套件管理員，您將需要修改下列一些步驟。

1. 使用 AWS CLI 執行 create-domain 命令。

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

即會在輸出中顯示 JSON 格式化資料，並且會顯示有關您新網域的詳細資訊。

```
{
    "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，確保您AWS CLI已配置並且您的預設區域名稱設為您建立儲存庫時所在的區域，請參閱設定 AWS 命令列界面。

2. 使用create-repository命令在您的域中創建一個存儲庫。

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

即會在輸出中顯示 JSON 格式化資料，並且會顯示有關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命令為您創建一個上游存儲庫my-repo儲存庫。

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

即會在輸出中顯示 JSON 格式化資料，並且會顯示有關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命令將外部連接添加到 npm 公共倉庫npm-store儲存庫。

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

JSON 格式的資料會出現在儲存庫及其新外部連線的輸出中。

```
{
```

```
    "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 存儲庫 Connect 到公共存儲庫。

5. 使用update-repository指令以關聯npm-store儲存庫做為上游儲存庫my-repo儲存庫。

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

JSON 格式的資料會出現在您的輸出中，包括其新上游儲存庫的詳細資訊。

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

```
}
```

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命令來配置你的 npm 軟件包管理器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 配置和使用 npm CodeArtifact。

7. 使用 npm CLI 來安裝 NPM 套件。例如，要安裝流行的 npm 軟件包lodash，使用下列命令。

```
npm install lodash
```

8. 使用list-packages命令來查看您剛剛安裝的軟件包my-repo儲存庫。

> **Note**
>
> 之間可能會有延遲npm installinstall 命令完成，並且當軟件包在存儲庫中可見時。如需從公用儲存庫擷取套件時一般延遲的詳細資訊，請參閱外部連延延。

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

JSON 格式的資料會出現在您安裝的輸出中。

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

```
}
```

您現在有三個 CodeArtifact 資源：

- 網域my-domain。
- 儲存庫my-repo包含於my-domain。這個存儲庫有一個可供它使用的 npm 軟件包。
- 儲存庫npm-store包含於my-domain。此存儲庫具有與公共 npm 存儲庫的外部連接，並與my-repo儲存庫。

9. 避免AWS收費，刪除您在此教學課程中使用的資源：

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

a. 使用delete-repository命令來刪除npm-store儲存庫。

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

即會在輸出中顯示 JSON 格式化資料，並且會顯示有關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npm-store儲存庫。

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

即會在輸出中顯示 JSON 格式化資料，並且會顯示有關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my-domain儲存庫。

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

JSON 格式的資料會出現在已刪除網域的輸出中。

```
{
    "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
    }
}
```

使用儲存庫 CodeArtifact

這些主題說明如何使用主 CodeArtifact 控台 AWS CLI、和 CodeArtifact API 來建立、列出、更新和刪除存放庫。

主題

建立 儲存庫

由於中的所有套件 CodeArtifact 都儲存在儲存庫中，因此若要使用 CodeArtifact，您必須建立一個套件。您可以使用 CodeArtifact 主控台 AWS Command Line Interface (AWS CLI) 或建立存放庫 AWS CloudFormation。每個儲存庫都與您建立時使用的 AWS 帳戶相關聯。您可以有多個存儲庫，並在域中創建和分組它們。當您建立存放庫時，它不會包含任何套件。存儲庫是 polyglot，這意味著單個存儲庫可以包含任何受支持類型的軟件包。

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目上限，請參閱配額 AWS CodeArtifact。如果您達到允許存儲庫的最大數量，則可以刪除存儲庫以騰出更多空間。

一個存儲庫可以有一個或多個與其相關聯的 CodeArtifact 存儲庫作為上游存儲庫。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 使用中的上游存儲庫 CodeArtifact。

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

> **Note**
>
> 建立存放庫之後，您無法變更其名稱、關聯 AWS 帳戶或網域。

主題

創建一個存儲庫（控制台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [儲存庫]，然後選擇 [建立存放庫]。
3. 在存放庫名稱中，輸入存放庫的名稱。
4. (選擇性) 在存放庫說明中，輸入存放庫的選擇性說明。
5. (選擇性) 在「發佈上游儲存庫」中，新增將儲存庫連接至套件授權單位的中繼儲存庫，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。如果其他 AWS 帳戶擁有網域，請選擇不同的 AWS 帳戶。
8. 在「網域」中，選擇要在其中建立存放庫的網域。

   如果帳戶中沒有網域，您必須建立網域。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。

   - 如果要使用預設值，請選擇 AWS 受管金鑰 AWS 受管金鑰。
   - 如果您要使用您管理的 KMS 金鑰，請選擇「客戶管理金鑰」。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 受管金鑰和客戶管理的金鑰。

9. 選擇下一步。
10. 在「檢閱並建立」中，檢閱 CodeArtifact 正在為您建立的內容。

    - Package 流程會顯示網域和儲存庫的連線方式。
    - 第 1 步：創建存儲庫顯示有關存儲庫和將創建的可選上游存儲庫的詳細信息。
    - 步驟 2：選擇域顯示有關的詳細信息my_domain。

    準備就緒後，請選擇 [建立儲存庫]。

建立儲存庫 (AWS CLI)

使用指create-repository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。建立儲存庫時，每個儲存庫都會與您驗證的 AWS 帳戶相關聯。

使用標籤建立儲存庫

若要使用標籤建立儲存庫，請將--tags參數新增至您的create-domain指令。

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

使用上游存儲庫創建一個存儲庫

您可以在建立存放庫時指定一或多個上游存放庫。

```
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**
>
> 若要使用上游建立存放庫，您必須擁有上游存放庫上AssociateWithDownstreamRepository動作的權限。

若要在建立儲存庫之後將上游新增至儲存庫，請參閱添加或刪除上游存儲庫（控制台）和新增或移除上游儲存庫 (AWS CLI)。

連接到儲存庫

設定設定檔和憑證以對 AWS 帳戶進行驗證之後，請決定要在中使用哪個存放庫 CodeArtifact。您有下列選項：

- 建立 儲存庫。如需詳細資訊，請參閱建立存放庫。
- 使用您帳戶中已存在的儲存庫。您可以使用該`list-repositories`命令查找在您的 AWS 帳戶中創建的存儲庫。如需詳細資訊，請參閱 列出儲存庫。
- 在不同 AWS 帳戶中使用儲存庫。如需詳細資訊，請參閱存放庫原則。

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。

- CodeArtifact 與 Maven 一起使用
- CodeArtifact 與 npm 一起使用
- CodeArtifact 搭配使用 NuGet
- CodeArtifact 與 Python 一起使用

刪除儲存庫

您可以使用 CodeArtifact 控制台或刪除存放庫 AWS CLI。刪除儲存庫之後，您就無法再將套件推送至該儲存庫或從中提取套件。儲存庫中的所有套件都會變成永久無法使用且無法還原。您可以使用相同的名稱建立儲存庫，但其內容將為空。

主題

- 刪除存放庫（控制台）
- 刪除存放庫 (AWS CLI)

刪除存放庫（控制台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
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。

刪除存放庫 (AWS CLI)

使用指`delete-repository`令刪除存放庫。

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

列出儲存庫

使用本主題中的命令列出 AWS 帳戶或網域中的儲存庫。

列出 AWS 帳戶中的存儲庫

使用此命令列出您 AWS 帳戶中的所有存儲庫。

```
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`--max-results`和`--next-token`參數`list-repositories`來分頁回應。對於`--max-results`，請指定介於 1 到 1000 之間的整數，以指定單一頁面中傳回的結果數目。它的默認值是 50。若要傳回後續頁面，請`list-repositories`再次執行，並將先前指令輸出中接收到的`nextToken`值傳遞給`--next-token`。如果不使用該`--next-token`選項，則始終返回結果的第一頁。

列出網域中的儲存庫

用`list-repositories-in-domain`於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
```

輸出顯示某些存儲庫由不同的 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--max-results和--next-token參數list-repositories-in-domain來分頁回應。對於--max-results，請指定介於 1 到 1000 之間的整數，以指定單一頁面中傳回的結果數目。它的默認值是 50。若要傳回後續頁面，請list-repositories-in-domain再次執行，並將先前指令輸出中接收到的nextToken值傳遞給--next-token。如果不使用該--next-token選項，則始終返回結果的第一頁。

要在更緊湊的列表中輸出存儲庫名稱，請嘗試以下命令。

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

輸出範例：

```
repo1
repo2
repo3
```

下列範例除了儲存庫名稱之外，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
```

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

檢視或修改儲存區域組態

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

> **Note**
>
> 建立存放庫之後，您無法變更其名稱、關聯 AWS 帳戶或網域。

主題

檢視或修改儲存庫組態 (主控台)

您可以使用 CodeArtifact 控制台查看有關存儲庫的詳細信息並更新。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [存放庫]，然後選擇您要檢視或修改的存放庫名稱。
3. 展開「詳細資料」以查看下列內容

   - 存放庫的網域。選擇域名以了解更多信息。
   - 存放庫的資源策略。選擇套用儲存區域原則來新增一個。
   - 存儲庫的 Amazon 資源名稱（ARN）。
   - 如果您的存放庫具有外部連接，則可以選擇連接以進一步了解它。一個存放庫只能有一個外部連接。如需詳細資訊，請參閱 將 CodeArtifact 存儲庫 Connect 到公共存儲庫。
   - 如果您的存儲庫具有上游存儲庫，則可以選擇一個來查看其詳細信息。一個存儲庫最多可以有 10 個直接上游存儲庫。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。

   > **Note**
   >
   > 存儲庫可以具有外部連接或上游存儲庫，但不能同時具有兩者。

4. 在「套件」中，您可以看到此儲存庫可用的任何套件。選擇一個套件以了解更多信息。
5. 選擇 [檢視連線指示]，然後選擇套件管理員以了解如何設定它 CodeArtifact。
6. 選擇套用儲存區域原則來更新或新增資源原則至您的儲存區域。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。

- 存放庫描述。
- 與儲存庫相關聯的標籤。
- 如果您的存放庫具有外部連接，則可以更改其連接到哪個公共存儲庫。否則，您可以將一個或多個現有存儲庫添加為上游存儲庫。請求包裹 CodeArtifact 時，按照您希望它們優先順序排列它們。如需詳細資訊，請參閱 上游存放庫優先順序。

檢視或修改儲存區域組態 (AWS CLI)

若要在中檢視儲存庫的目前組態 CodeArtifact，請使用`describe-repository`指令。

```
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。若要新增或變更儲存庫的上游關係，請使用`update-repository`指令。

```
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**
>
> 若要新增上游存放庫，您必須擁有上游存放庫上AssociateWithDownstreamRepository動作的權限。

若要移除儲存庫的上游關係，請使用空白清單作為--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 使用以資源為基礎的權限來控制存取。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，讓其他 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原則和以身分識別為基礎的原則和以資源為基

建立資源策略以授與讀取存取權

資源策略是 JSON 格式的文本文件。檔案必須指定主參與者 (actor)、一或多個動作，以及效果 (Allow或Deny)。例如，下列資源策略會授與帳號從存放庫下載套件的123456789012權限。

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

因為只會針對其所附加儲存庫的作業評估原則，因此您不需要指定資源。由於資源是隱含的，所以您可以將設定Resource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原則。網域原則必須至少授codeartifact:GetAuthorizationToken與sts:GetServiceBearerToken權限給主體。如需授與跨帳戶存取權的完整網域原則範例，請參閱此網域原則範例處。

> **Note**
>
> 動codeartifact:ReadFromRepository作只能在存放庫資源上使用。您無法將套件的 Amazon 資源名稱 (ARN) 做為資源，以允許讀取存放庫中套件子集的動

作。codeartifact:ReadFromRepository指定的主體可以讀取儲存庫中的所有套件，也可以不讀取其中一個套件。

由於存放庫中指定的唯一動作是ReadFromRepository，帳戶中的使用者和角色1234567890可以從存放庫下載套件。但是，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ReadFromRepository而言，除了從存放庫下載套件的使用者也需要以其他方式與其互動之外，您還必須在下列原則中授與權限。

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

設定策略

建立原則文件之後，請使用以下put-repository-permissions-policy指令將其附加至儲存庫：

```
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，會在評估權限時忽略存放庫上的資源策略。如此可確保網域的擁有者無法將自己鎖定在存放庫之外，因此他們無法更新資源原則。

> **Note**
>
> 您無法將使用資源策略更新儲存庫上資源策略的權限授與其他 AWS 帳號，因為呼叫時會忽略資源策略 put-repository-permissions-policy。

輸出範例：

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

命令的輸出包含存放庫資源的 Amazon 資源名稱 (ARN)、政策文件的完整內容以及修訂識別碼。您可以put-repository-permissions-policy使用--policy-revision選項將修訂識別元傳遞給。這可確保文件的已知修訂版本會被覆寫，而不是由其他作者設定的較新版本。

閱讀政策

使用get-repository-permissions-policy指令讀取原則文件的現有版本。若要格式化輸出以提高可讀性，請使用--output和與 Python json.tool 模組--query policy.document一起使用。

```
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`指令從儲存庫刪除原則。

```
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。

授與主參與者的讀取存取權

當您將帳號的 root 使用者指定為策略文件中的主參與者時，即授與該帳號中所有使用者和角色的存取權。若要限制對選取的使用者或角色的存取，請在策略的`Principal`區段中使用其 ARN。例如，使用以下指令授與帳戶`bob`中的 IAM 使用者讀取存取權限`123456789012`。

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

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

授與封裝的寫入存取權

此codeartifact:PublishPackageVersion動作可用來控制發佈新版封裝的權限。與此動作搭配使用的資源必須是封裝。 CodeArtifact 封裝 ARN 的格式如下。

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

下列範例會顯示 NPM 套件的 ARN，其範圍@parity和名稱ui位於網域my_domain的my_repo儲存庫中。

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

沒有範圍的 npm 套件的 ARN 具有命名空間欄位的空字串。例如，以下是 ARN，適用於沒有範圍且名稱react位於網域my_domain中的my_repo儲存庫中的套件。

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

下列策略授與帳號123456789012權限，以便在my_repo存放庫@parity/ui中發行版本。

```
{
    "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**
>
> 要授予發布 Maven 和 NuGet 包版本的權限，除了添加以下權限codeartifact:PublishPackageVersion。
>
> 1. NuGet：codeartifact:ReadFromRepository並指定儲存庫資源
> 2. 釋界：codeartifact:PutPackageMetadata

由於此原則將網域和存放庫指定為資源的一部分，因此只有在附加至該存放庫時才允許發行。

授與存放庫的寫入權限

您可以使用萬用字元來授與存放庫中所有套件的寫入權限。例如，使用下列原則授與帳號權限，以寫入my_repo存放庫中所有套件。

```
{
    "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/*"
        }
    ]
```

```
}
```

標記儲存庫 CodeArtifact

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在中將標籤套用至儲存庫 CodeArtifact。如需有關資 CodeArtifact 源標記、使用案例、標籤索引鍵和值條件約束以及支援的資源類型的資訊，請參閱標記資源。

您可以在建立存放庫時使用 CLI 指定標籤。您可以使用主控台或 CLI 新增或移除標籤，以及更新存放庫中標籤的值。每個儲存庫最多可以新增 50 個標籤。

主題

- 標記儲存庫 (CLI)
- 標記儲存庫 (主控台)

標記儲存庫 (CLI)

您可以使用 CLI 來管理存放庫標籤。

主題

- 將標籤新增至存放庫 (CLI)
- 檢視存放庫 (CLI) 的標籤
- 編輯存放庫 (CLI) 的標籤
- 從存放庫 (CLI) 移除標籤

將標籤新增至存放庫 (CLI)

您可以使用主控台或標 AWS CLI 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**
>
> 要獲取存儲庫的 ARN，請運行以下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。######## my_domain ############## my_ repo ####### ## key1 ########### value1####### key 2 ############ 2#

```
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，則此命令沒有輸出。

檢視存放庫 (CLI) 的標籤

請遵循下列步驟 AWS CLI 來使用檢視存放庫的 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**
>
> 要獲取存儲庫的 ARN，請運行以下describe-repository命令：
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

######### my_domain ### ARN ####### my_repo ############## ##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"
    }
}
```

編輯存放庫 (CLI) 的標籤

請依照下列步驟 AWS CLI 來使用編輯存放庫的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。

在終端機或命令列上，執行指tag-resource令，指定要更新標籤的儲存庫的 ARN，並指定標籤鍵和標籤值。

> **Note**
>
> 要獲取存儲庫的 ARN，請運行以下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，則此命令沒有輸出。

從存放庫 (CLI) 移除標籤

請依照下列步驟使 AWS CLI 用從存放庫移除標籤。

> **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。

在終端機或命令列上，執行指untag-resource令，指定要移除標籤的儲存庫的 ARN，以及要移除之標籤的標籤金鑰。

> **Note**
>
> 要獲取存儲庫的 ARN，請運行以下`describe-repository`命令：
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

```
####### my_domain ########## 1 # key2 #### my_ repo ###########
```

```
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，則此命令沒有輸出。移除標籤後，您可以使用`list-tags-for-resource`指令檢視儲存庫上的剩餘標籤。

標記儲存庫 (主控台)

您可以使用主控台或 CLI 以標記資源。

主題

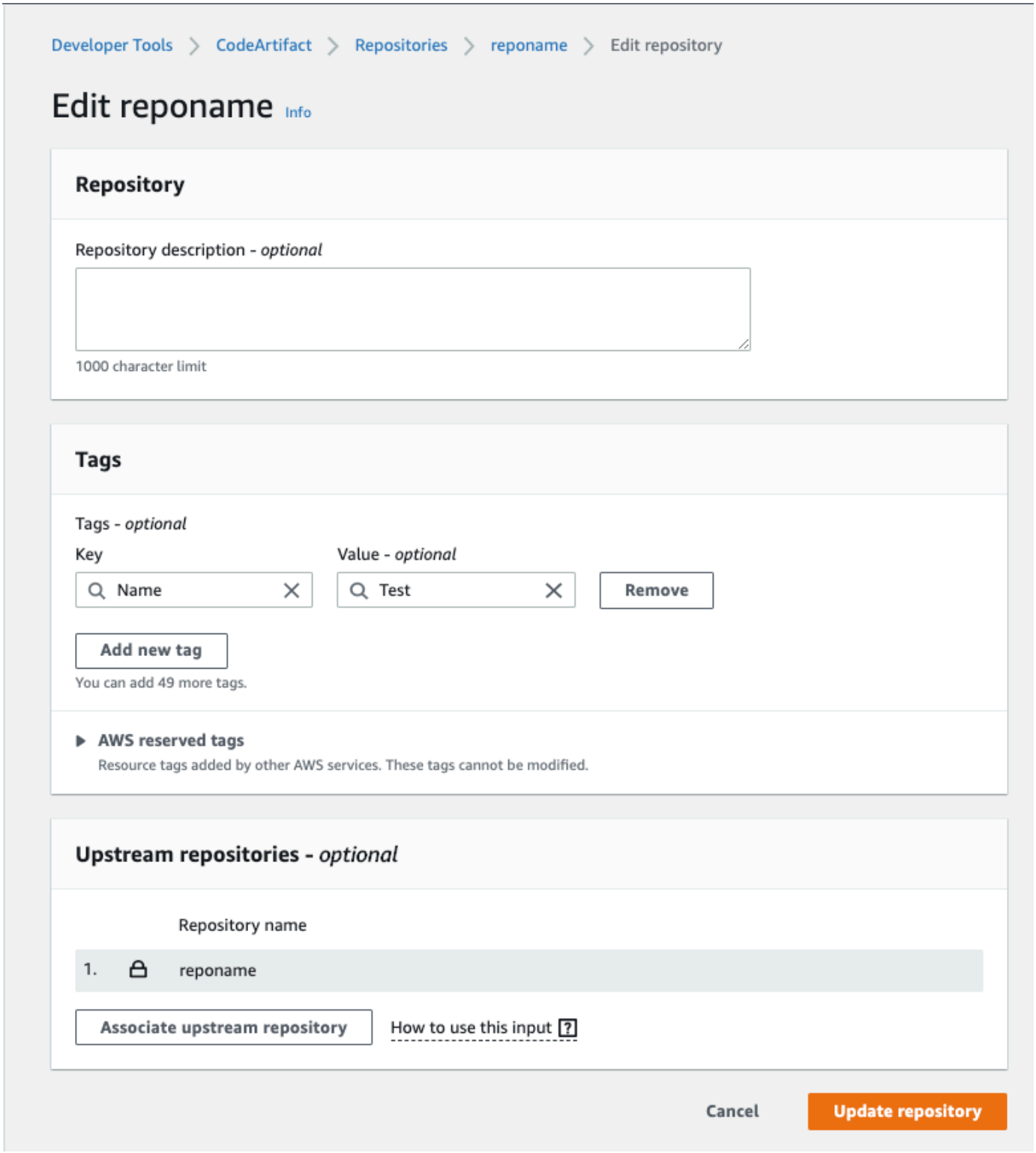
- 將標籤新增至儲存庫 (主控台)
- 檢視儲存庫的標籤 (主控台)
- 編輯儲存庫的標籤 (主控台)
- 從儲存庫移除標籤 (主控台)

將標籤新增至儲存庫 (主控台)

您可以使用控制台將標籤添加到現有存儲庫中。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「存放庫」頁面上，選擇要新增標記的存放庫。
3. 展開「詳細資料」區段。
4. 在 [儲存庫標籤] 底下，如果儲存庫中沒有標籤，請選擇 [新增儲存庫標籤]。如果儲存庫上有標籤，請選擇檢視並編輯儲存庫標籤。
5. 選擇 Add new tag (新增標籤)。

6. 在「關鍵字」和「值」欄位中，輸入您要加入的每個標籤的文字。(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)

您可以使用控制台列出現有存儲庫的標籤。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「存放庫」頁面上，選擇要檢視標籤的存放庫。
3. 展開「詳細資料」區段。
4. 在存放庫標籤下，選擇檢視並編輯儲存庫標籤。

> **Note**
>
> 如果沒有添加到此存儲庫的標籤，控制台將讀取添加存儲庫標籤。

編輯儲存庫的標籤 (主控台)

您可以使用主控台編輯已新增至儲存庫的標籤。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「存放庫」頁面上，選擇您要更新標籤的存放庫。
3. 展開「詳細資料」區段。
4. 在存放庫標籤下，選擇檢視並編輯儲存庫標籤。

> **Note**
>
> 如果沒有添加到此存儲庫的標籤，控制台將讀取添加存儲庫標籤。

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. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「存放庫」頁面上，選擇要移除標籤的存放庫。
3. 展開「詳細資料」區段。
4. 在存放庫標籤下，選擇檢視並編輯儲存庫標籤。

> **Note**
>
> 如果沒有添加到此存儲庫的標籤，控制台將讀取添加存儲庫標籤。

5. 在您要刪除之每個標籤的機碼和值旁邊，選擇「移除」。
6. 選擇更新儲存庫。

使用中的上游存儲庫 CodeArtifact

一個存儲庫可以有其他存 AWS CodeArtifact 儲庫作為上游存儲庫。這可讓套件管理員用戶端使用單一存放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 AWS CodeArtifact 存放庫新增至存放庫。若要將存放庫與上游存放庫相關聯，您必須擁有上游存放庫上`AssociateWithDownstreamRepository`動作的權限。如需詳細資訊，請參閱 使用上游存儲庫創建一個存儲庫 及 新增或移除上游儲存庫。

如果上游存放庫具有與公用存放庫的外部連線，則其下游的存放庫可以從該公用存放庫提取套件。例如，假設存放庫`my_repo`有一個名為的上游存放庫`upstream`，並且`upstream`具有與公共 npm 存放庫的外部連接。在這種情況下，連接到的軟件包管理器`my_repo`可以從 npm 公共存儲庫中提取軟件包。如需有關從上游存放庫或外部連線要求套件的詳細資訊，請參閱請求具有上游存儲庫的軟件包版本或從外部連線要求套件。

主題

- 上游存儲庫和外部連接有什麼區別？
- 新增或移除上游儲存庫
- 將 CodeArtifact 存儲庫 Connect 到公共存儲庫
- 請求具有上游存儲庫的軟件包版本
- 從外部連線要求套件
- 上游存放庫優先順序
- 上游存儲庫的 API 行為

上游存儲庫和外部連接有什麼區別？

在中 CodeArtifact，上游存儲庫和外部連接的行為大致相同，但存在一些重要的差異。

1. 您最多可以將 10 個上游存儲庫添加到一個 CodeArtifact 存儲庫。您只能新增一個外部連線。
2. 有單獨的 API 調用來添加上游存儲庫或外部連接。
3. 套件保留行為略有不同，因為上游儲存庫要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 Package 保留。

新增或移除上游儲存庫

請遵循以下各節中的步驟，在存放庫中新增或移除上游存 CodeArtifact 放庫。如需有關上游儲存庫的詳細資訊，請參閱使用中的上游存儲庫 CodeArtifact。

本指南包含有關將其他 CodeArtifact 儲存庫配置為上游儲存庫的資訊。有關配置外部連接到公共存儲庫（如 npmjs.com，Nuget 圖庫，Maven 中央或 PyPI）的信息，請參閱添加外部連接。

添加或刪除上游存儲庫（控制台）

執行下列程序中的步驟，使用 CodeArtifact 主控台將存放庫新增為上游存放庫。如需有關使用新增上游存放庫的資訊 AWS CLI，請參閱新增或移除上游儲存庫 (AWS CLI)。

若要使用 CodeArtifact 主控台新增上游存放庫

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [網域]，然後選擇包含存放庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游存放庫中，選擇關聯上游存放庫並新增您要新增為上游存放庫的存放庫。您只能在與上游存儲庫相同的域中添加存儲庫。
6. 選擇更新儲存庫。

若要使用 CodeArtifact 主控台移除上游存放庫

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [網域]，然後選擇包含存放庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游存放庫中，找到您要移除之上游存放庫的清單項目，然後選擇 [取消關聯]。

> **Important**
>
> 從儲存庫中移除上游儲存庫後，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。 CodeArtifact

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 (AWS CLI) 新增或移除儲 CodeArtifact 存庫的上游儲存庫。若要這麼做，請使用指`update-repository`令，並使用`--upstreams`參數指定上游儲存庫。

您只能在與上游存儲庫相同的域中添加存儲庫。

若要新增上游儲存庫 (AWS CLI)

1. 如果您還沒有，請按照中設定使用 AWS CodeArtifact的步驟進行設置和配置 CodeArtifact。 AWS CLI
2. 使用帶有`--upstreams`標誌的`aws codeartifact update-repository`命令來添加上游存儲庫。

> **Note**
>
> 呼叫此指`update-repository`令會以`--upstreams`旗標所提供的儲存庫清單取代現有已設定的上游儲存庫。如果要添加上游存儲庫並保留現有存儲庫，則必須在調用中包含現有的上游存儲庫。

以下示例命令將兩個上游存儲庫添加到名為`my_repo`的域中名為的存儲庫中`my_domain`。從存`my_repo`放庫 CodeArtifact 要求套件時，上游存放庫在`--upstreams`參數中的順序會決定其搜尋優先順序。如需詳細資訊，請參閱 上游存放庫優先順序。

如需有關連線至公用外部儲存庫 (例如 npmjs.com 或 Maven 中央) 的資訊，請參閱。將 CodeArtifact 存儲庫 Connect 到公共存儲庫

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

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

輸出包含上游存放庫，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": []
       }
   }
```

若要移除上游存放庫 (AWS CLI)

1. 如果您還沒有，請按照中設定使用 AWS CodeArtifact的步驟進行設置和配置 CodeArtifact。 AWS CLI
2. 要從存儲庫中刪除上游存儲 CodeArtifact 庫，請使用帶有--upstreams標誌的update-repository命令。提供給該命令的存儲庫列表將是 CodeArtifact 存儲庫的新上游存儲庫集。包括您要保留的現有上游存儲庫，並省略要刪除的上游存儲庫。

   要從存儲庫中刪除所有上游存儲庫，請使用該update-repository命令並包含--upstreams而不帶參數。以下內容會從名稱為的網域中名稱my_repo的儲存庫中移除上游儲存庫my_domain。

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

```
    --upstreams
```

輸出顯示的清單upstreams是空的。

```
{
       "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 存儲庫 Connect 到公共存儲庫

您可以在 CodeArtifact 存儲庫和外部公共存儲庫（如 https://npmjs.com 或 Maven 中央存儲庫）之間添加外部連接。然後，當您從 CodeArtifact 存放庫中尚未存在的存放庫要求套件時，可以從外部連線擷取套件。這樣就可以消耗應用程序使用的開源依賴關係。

在中 CodeArtifact，使用外部連線的預期方法是每個網域都有一個存放庫，且具有與指定公用存放庫的外部連線。例如，如果您要連接到 npmjs.com，請在域中配置一個與 npmjs.com 的外部連接的存儲庫，並使用其上游配置所有其他存儲庫。這樣，所有存儲庫都可以使用已經從 npmjs.com 獲取的軟件包，而不是再次獲取並存儲它們。

主題

- Connect 到外部存儲庫（控制台）
- Connect 至外部存放庫 (CLI)
- 支援的外部連線儲存
- 移除外部連線 (CLI)

Connect 到外部存儲庫（控制台）

當您使用控制台將連接添加到外部存儲庫時，將發生以下情況：

1. 如果外部存-store放庫不存在，則會在您的 CodeArtifact 網域中建立外部儲存庫的儲存庫。這些-store儲存庫在您的儲存庫與外部儲存庫之間的行為是中繼儲存庫，可讓您連線至多個外部存放庫。
2. 適當的-store存放庫會新增為您的儲存庫的上游。

下列清單包含中的每個-store存放庫以 CodeArtifact 及它們連線到的相應外部存放庫。

1. commonsware-store已連接到 CommonsWare 安卓版本庫。
2. google-android-store連接到谷歌安卓系統。
3. gradle-plugins-store連接到搖籃插件。
4. maven-central-store連接到 Maven 中央存儲庫。
5. clojars-store連接到 Clojar 存儲庫。
6. npm-store已連接至網站。
7. nuget-store已連線至核心組織。
8. pypi-store連接到 Python 包裝管理局。
9. rubygems-store已連接到 RubyGems .org。

連線至外部存放庫 (主控台)

1. 開啟主 AWS CodeArtifact 控台，網址為 https://console.aws.amazon.com/codesuite/codeartifact/home。
2. 在瀏覽窗格中，選擇 [網域]，然後選擇包含存放庫的網域名稱。
3. 選擇儲存庫的名稱。
4. 選擇編輯。
5. 在上游存放庫中，選擇「關聯上游存放庫」，然後新增作為上游連接的適當-store存放庫。
6. 選擇更新儲存庫。

將-store存放庫新增為上游儲存庫之後，連接至您儲存 CodeArtifact 庫的套件管理員可以從相應的外部儲存庫擷取套件。

Connect 至外部存放庫 (CLI)

您可以使用將 CodeArtifact 存放庫直接新增外部連線至存放庫，將存放庫連線至外部存放庫。 AWS CLI 這可讓連線至儲 CodeArtifact 存區域或其任何下游儲存區域的使用者從設定的外部儲存區域擷取套裝程式。每個 CodeArtifact 儲存庫只能有一個外部連線。

建議每個域有一個存儲庫，並與給定的公共存儲庫進行外部連接。要將其他存儲庫連接到公共存儲庫，請將具有外部連接的存儲庫作為其上游添加。如果您或網域中的其他人已經在主控台中設定了外部連線，則您的網域可能已經有一個`-store`存放庫，其中包含與您要連線的公用存放庫的外部連線。如需有關`-store`儲存庫和與主控台連線的詳細資訊，請參閱Connect 到外部存儲庫（控制台）。

將外部連線新增至存 CodeArtifact 放庫 (CLI)

- 用`associate-external-connection`於新增外部連線。下列範例會將儲存庫連線至 NPM 公用登錄 npmjs.com。如需受支援的外部儲存庫清單，請參閱支援的外部連線儲存。

```
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"
            }
        ]
    }
}
```

新增外部連線之後，請參閱以從外部連線要求套件取得有關透過外部連線從外部存放庫要求套件的資訊。

支援的外部連線儲存

CodeArtifact 支援與下列公用儲存庫的外部連線。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`參數的 Name 資料欄中的值。

儲存庫類型	描述	名稱
NPM	npm 公共註冊表	`public:npmjs`
Python	Python Package 索引	`public:pypi`
Maven	Maven 中央	`public:maven-central`
Maven	谷歌安卓儲存庫	`public:maven-googleandroid`
Maven	搖籃插件庫	`public:maven-gradleplugins`
Maven	CommonsWare 安卓儲存庫	`public:maven-commonsware`
Maven	罐子存儲庫	`public:maven-clojars`
NuGet	NuGet 畫廊	`public:nuget-org`
Ruby	RubyGems.org	`public:ruby-gems-org`

移除外部連線 (CLI)

若要移除使用中的`associate-external-connection`指令加入的外部連接 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": []
    }
}
```

請求具有上游存儲庫的軟件包版本

當客戶端（例如 npm）從具有多個上游 CodeArtifact 存儲庫的名my_repo為的存儲庫請求軟件包版本時，可能會發生以下情況：

- 如果my_repo包含請求的軟件包版本，則將其返回給客戶端。
- 如果my_repo不包含請求的軟件包版本，請在上游存儲庫中my_repo查 CodeArtifact找它。如果找到套件版本，則會將其參照複製到my_repo，並將套件版本傳回給用戶端。
- 如果其上游存放庫都my_repo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指令新增上游儲存庫時，在要求套件版本時，它們傳遞至--upstreams參數的順序會決定其優先順序。請求套件版本時，依您 CodeArtifact 要使用的順序指定上游儲存庫。--upstreams如需詳細資訊，請參閱 上游存放庫優先順序。

一個儲存庫允許的直接上游儲存庫數目上限為 10。由於直接上游存儲庫也可以擁有自己的直接上游存儲庫，因此 CodeArtifact 可以在 10 個以上的軟件庫中搜索軟件包版本。當要求套件版本時，CodeArtifact 查找的儲存庫數目上限為 25。

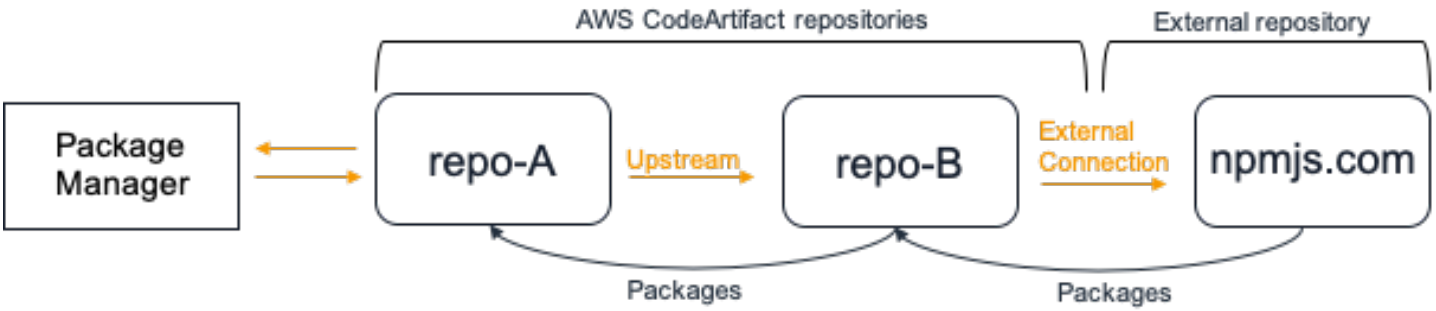
從上游儲存庫保留 Package

如果在上游存放庫中找到要求的套件版本，則會保留對該套件的參考，並且永遠可從下游存放庫中取得。保留的套件版本不受下列任何一項影響：

- 刪除上游存放庫。
- 中斷上游存放庫與下游存放庫的連線。
- 從上游存放庫刪除套件版本。
- 編輯上游存放庫中的套件版本 (例如，向其新增資產)。

透過上游關係擷取套件

如果 CodeArtifact 存放庫與具有外部連線的存放庫具有上游關係，則會從外部存放庫複製不在上游存放庫中的套件要求。例如，請考慮下列組態：名為的存放庫`repo-A`具有名為的上游存放庫`repo-B`。`repo-B`有一個外部連接到 https://npmjs.com。



如果設定npm為使用`repo-A`儲存庫，執行`npm install`會觸發將套件從 https://npmjs.com 複製到`repo-B`。安裝的版本也會被拉入`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
```

運行後`npm install`，只`repo-A`包含最新版本（`lodash 4.17.20`），因為這是npm從中`repo-A`獲取的版本。

```
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"
        }
    ]
}
```

由於與 https://npmjs.com repo-B 有外部連接，所有從 https://npmjs.com 導入的軟件包版本都存儲在repo-B. 這些套件版本可能已由任何具有上游關係的下游存放庫擷取。repo-B

的內容repo-B提供了一種查看隨著時間的推移從 https://npmjs.com 導入的所有軟件包和軟件包版本的方法。例如，若要查看隨時間匯入的lodash套件的所有版本，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"
```

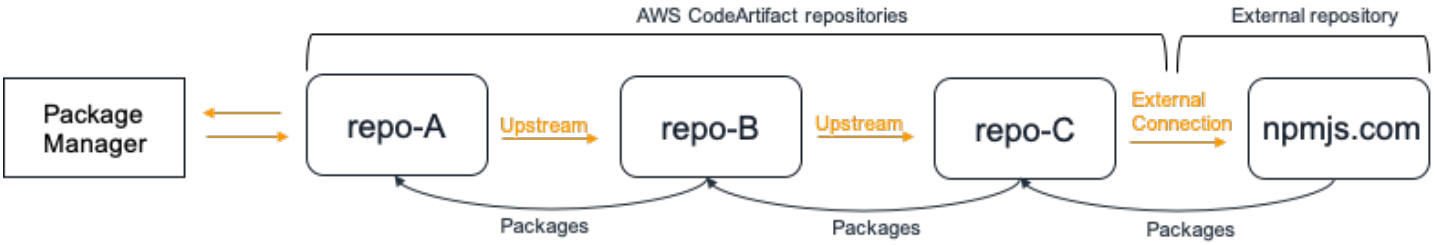
```
        },
        {
            "version": "0.2.0",
            "revision": "REVISION-3-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
        },
        {
            "version": "0.2.1",
            "revision": "REVISION-4-SAMPLE-6C81EFF7DA55CC",
            "status": "Published"
  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

中繼儲存庫中的 Package 保留

CodeArtifact 允許鏈接上游存儲庫。例如，repo-A可以有一repo-B個上游，並且repo-B可repo-C以有一個上游。此配置使軟件包版本在中repo-B並從中repo-C提供repo-A。

當套裝程式管理員連線至儲存庫repo-A並從儲存庫擷取套裝程式版本時repo-C，套件版本將不會保留在儲存庫repo-B中。套件版本只會保留在最下游的存放庫中，在此範例中為。repo-A它不會保留在任何中間存儲庫中。對於較長的鏈結也是如此；例如，如果有四個存放庫repo-A、repo-Brepo-C、repo-D和一個連接到從中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或中，repo-A但不會保留在repo-B或repo-C中。

從外部存放庫提取 Package 裝軟體版本時，封裝保留行為類似，只是套件版本一律會保留在已附加外部連線的存放庫中。例如，repo-A具有repo-B作為上游。 repo-B具有上游repo-C，並且repo-C還將 npmjs.com 配置為外部連接; 請參閱下圖。



如果連接到的軟件包管理器**repo-A**請求一個軟件包版本，例如 lodash 4.17.20，並且該軟件包版本不存在於三個存儲庫中的任何一個，則將從 npmjs.com 提取該軟件包版本。當 lodash 4.17.20 被提取時，它將被保留在中，**repo-A**因為它是最下游的存儲庫，並且**repo-C**它已附加到 npmjs.com 的外部連接。 lodash 4.17.20 將不會保留在中間存儲庫，`repo-B`因為它是一個中間存儲庫。

從外部連線要求套件

下列各節說明如何從外部連線要求套件，以及要求套件時的預期 CodeArtifact 行為。

主題

- 從外部連線擷取套件
- 外部連延延
- CodeArtifact 外部儲存庫無法使用時的行為
- 新套件版本的可用性
- 匯入包含多個資產的套件版本

從外部連線擷取套件

若要在將外部連線新增至 CodeArtifact儲存庫 (如中所述) 之後從外部連線擷取套件將 CodeArtifact 存儲庫 Connect 到公共存儲庫，請設定套件管理員使用您的存放庫並安裝套件。

> **Note**
>
> 下列指示使用npm, 來檢視其他套件類型的組態和使用指示 CodeArtifact 與 Maven 一起使用，請參閱使用CodeArtifact取代為NuGet、或使用CodeArtifact與蟒蛇。

若要從外部連線擷取套件

1. 使用 CodeArtifact 儲存庫配置和驗證您的套件管理員。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. 公有儲存庫使用套件。對於npm，請使用以下npm install命令，將 *lodash* 替換為您要安裝的軟件包。

```
npm install lodash
```

3. 將套件複製到 CodeArtifact 儲存庫之後，您可以使用list-packages和指list-package-versions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"
        }
    ]
}
```

此命list-package-versions令會列出複製到 CodeArtifact 儲存庫中的套件的所有版本。

```
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"
        }
    ]
}
```

外部連延延

使用外部連線從公用儲存庫擷取套件時，從公用存放庫擷取套件的時間和儲存在您的存放 CodeArtifact 庫中會有延遲。例如，假設您已經安裝了 npm 套件「lodash」的 1.2.5 版，如中所述從外部連線擷取套件。雖然`npm install lodash` lodash 命令已成功完成，但套件版本可能尚未出現在您的 CodeArtifact 儲存庫中。套件版本通常需要 3 分鐘左右的時間才會出現在您的儲存庫中，但偶爾可能需要更長的時間。

由於這種延遲，您可能已成功擷取套件版本，但可能還無法在 CodeArtifact 主控台中的存放庫中或呼叫 ListPackages 和 ListPackageVersions API 作業時看到該版本。 CodeArtifact 一旦以非同步方式保存套件版本，它就會在主控台和透過 API 要求顯示。

CodeArtifact 外部儲存庫無法使用時的行為

有時候，外部儲存庫會發生中斷情況，表示 CodeArtifact 無法從其擷取套件，或者擷取套件的速度會比正常情況慢得多。發生這種情況時，已經從外部存儲庫（例如 npmjs.com）提取並存儲在存儲 CodeArtifact 庫中的軟件包版本將繼續可以從中下載 CodeArtifact。但是，即使已設定與該儲存庫的外部連線，尚未儲存於中的套件也 CodeArtifact 可能無法使用。例如，您的 CodeArtifact 存儲庫可能包含 npm 軟件包版本，`lodash 4.17.19` 因為這是您到目前為止在應用程序中使用的版本。當您要升級到時`4.17.20`，通常 CodeArtifact 會從 npmjs.com 獲取該新版本並將其存儲在您的 CodeArtifact 存儲庫中。但是，如果 npmjs.com 發生中斷，則此新版本將無法使用。唯一的解決方法是在 npmjs.com 恢復後再試一次。

外部儲存庫中斷也會影響將新套件版本發佈至 CodeArtifact。在已設定外部連線的存放庫中，CodeArtifact 將不允許發佈已存在於外部存放庫中的套件版本。如需詳細資訊，請參閱套件概觀。但是，在極少數情況下，外部存放庫中斷可能表示 CodeArtifact 沒有外部存放庫中存在哪些套件和套件版本的 up-to-date 資訊。在這種情況下， CodeArtifact 可能會允許發布一個軟件包版本，它通常會拒絕。

新套件版本的可用性

若要讓公用儲存庫 (例如 npmjs.com) 中的套件版本可以透過 CodeArtifact 儲存庫取得，必須先將其新增至區域套件中繼資料快取。此快取會由每個AWS區域維護，並包含描述受支援公用儲存庫內容的 CodeArtifact 中繼資料。由於此快取的原因，在將新套件版本發佈至公用存放庫與可用時間之間會有延遲 CodeArtifact。此延遲因套件類型而異。

對於 npm、Python 和 Nuget 套件，從新的套件版本發佈到 npmjs.com、pypi.org 或 nuget.org 之後，可能會有最多 30 分鐘的延遲時間，以及可從 CodeArtifact 存放庫安裝的時間。 CodeArtifact 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 軟件包，從新軟件包版本發布到公共存儲庫以及可從存儲庫安裝之後，最多可能會延遲 3 小時。 CodeArtifact CodeArtifact 最多每 3 小時檢查一次軟件包的新版本。在 3 小時的快取存留期過期之後，對指定套件名稱的第一個要求，將導致該套件的所有新版本匯入區域快取。

對於常用的 Maven 軟件包，通常會每 3 小時導入一次新版本，因為請求的高速率意味著一旦緩存存留期過期，通常會更新緩存。對於不常使用的套裝程式，直到從 CodeArtifact儲存庫要求套件的版本之前，快取將不會有最新版本。在第一個要求中，只有先前匯入的版本可從中使用 CodeArtifact，但此要求會導致快取更新。在後續的要求中，套件的新版本將會新增至快取中，並可供下載。

匯入包含多個資產的套件版本

Maven 和 Python 軟件包可以在每個軟件包版本中具有多個資產。這使得導入這些格式的軟件包比 npm 和 NuGet packages 更複雜，每個軟件包版本只有一個資產。如需針對這些套件類型匯入哪些資產的說明，以及新增資產的可用方式的說明，請參閱從上流和外部連接請求 Python 包和從上流和外部連接請求 Maven 軟件包。

上游存放庫優先順序

當您從具有一或多個上游存放庫的存放庫要求套件版本時，其優先順序會與呼叫`create-repository`或`update-repository`指令時列出的順序相對應。找到請求的軟件包版本時，即使未搜索所有上游存儲庫，搜索也會停止。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`指令查看優先順序。

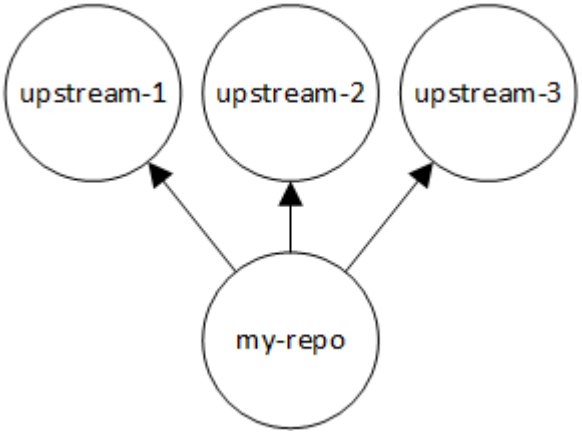
```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-
owner 111122223333
```

結果可能如下。它表明上游存儲庫優先級是`upstream-1`第一，第`upstream-2`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": []
    }
}
```

簡單的優先順序示例

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



中的套件版本要求會依下列順序my_repo搜尋儲存庫，直到找到儲存庫為止，或將 HTTP 404 回Not Found應傳回給用戶端為止：

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

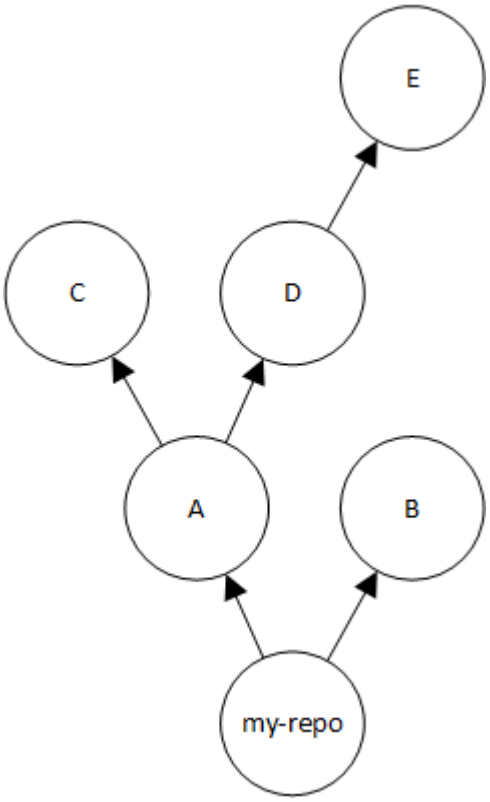
如果找到套件版本，即使未在所有上游儲存庫中查看，搜尋也會停止。例如，如果在中找到套件版本upstream-1，搜尋就會停止，而 CodeArtifact 不會查看upstream-2或upstream-3。

當您使用 AWS CLI 指令列出中list-package-versions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

如果上游存放庫有自己的上游存放庫，則在移至下一個上游存放庫之前，會使用相同的邏輯來尋找套件版本。例如，假設您的my_repo存放庫有兩個上游儲存庫，以A及B. 如果存放庫A具有上游存放庫，則my_repo首先會搜尋套件版本的要求A，然後在的上游儲存庫中尋找A，依此類推。my_repo

在下圖中，my_repo存放庫包含上游存放庫。上游存儲庫A有兩個上游存儲庫，並D有一個上游存儲庫。圖中相同級別的上游存儲庫以其優先順序從左到右顯示（存A儲庫的優先級順序高於存儲庫B，並C且存儲庫的優先級順序高於存儲庫D）。



在此範例中，中的套件版本要求會依下列順序在儲存庫中my_repo尋找，直到找到它為止，或直到套件管理員傳回 HTTP 404 回Not Found應給用戶端為止：

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

上游存儲庫的 API 行為

當您在連接到上游存放庫的存放庫上呼叫特定 CodeArtifact API 時，行為可能會因套裝軟體或套件版本是儲存在目標存放庫還是上游存放庫中而有所不同。這些 API 的行為記錄在這裡。

如需 API 的詳細資 CodeArtifact 訊，請參閱 CodeArtifact API 參考資料。

如果目標存放庫中沒有指定的套件版本，則參照套件或套件版本的大多數 API 都會傳回ResourceNotFound錯誤。即使套件或套件版本存在於上游存放庫中也是如此。實際上，調用這些 API 時會忽略上游存儲庫。這些 API 包括：

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

為了演示這種行為，我們有兩個存儲庫：target-repo和upstream-repo。 target-repo為空且已upstream-repo配置為上游儲存庫。 upstream-repo包含 npm 套件lodash。

當調用包含包的 DescribePackageVersion upstream-repo API 時，我們得到以下輸出：lodash

```
{
    "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"
    }
```

當調用相同的 API（空的target-repo，但已upstream-repo配置為上游）時，我們得到以下輸出：

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

CopyPackageVersionsAPI 的行為方式不同。依預設，CopyPackageVersionsAPI 只會複製儲存在目標儲存庫中的套件版本。如果套件版本儲存在上游存放庫中，但不儲存在目標存放庫中，則不會複製該套件版本。若要包含僅儲存在上游存放庫中的套件的套件版本，請在 API 請求true中includeFromUpstream將的值設定為。

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

使用中的套件 CodeArtifact

下列主題說明如何使用 CodeArtifact CLI 和 API 對封裝執行動作。

主題

套件概觀

套件是解決相依性和安裝軟體所需的軟體套件和中繼資料。在中 CodeArtifact，套件包含套件名稱、選擇性命名空間 (例如 `@types` in) `@types/node`、一組套件版本，以及封裝層級中繼資料 (例如 npm 標籤)。

內容

- Package 版本狀態
- Package 名稱、套件版本和資產名稱標準化

支援的套件格式

AWS CodeArtifact 支持 NPM，PyPI，Maven，斯威夫特 NuGet，紅寶石，通用包格式。

Package 發佈

您可以使用npm、twine、、、和等工具，將任何受支援套件格式的新版本發佈至 CodeArtifact 儲存庫dotnet。Maven Gradle nuget

發佈權限

您的 AWS Identity and Access Management (IAM) 使用者或角色必須具有發佈至目標儲存庫的權限。發佈套件需要下列權限：

- 釋界：codeartifact:PublishPackageVersion和 codeartifact:PutPackageMetadata
- 故宮：codeartifact:PublishPackageVersion
- NuGet：codeartifact:PublishPackageVersion和 codeartifact:ReadFromRepository
- Python：codeartifact:PublishPackageVersion
- 通用：codeartifact:PublishPackageVersion
- 迅速：codeartifact:PublishPackageVersion
- 紅寶石：codeartifact:PublishPackageVersion

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

如需中的權限的詳細資訊 CodeArtifact，請參閱AWS CodeArtifact 權限參考。

覆寫套件資產

您無法重新發佈已存在於不同內容的封裝資產。例如，假設您已經發佈了一個包含 JAR 資產的 Maven 套件mypackage-1.0.jar。只有在新舊資產的總和檢查碼相同時，您才能再次發佈該資產。若要重

新發佈具有新內容的相同資產，請先使用delete-package-versions指令刪除套件版本。嘗試重新發佈具有不同內容的相同資產名稱會導致 HTTP 409 衝突錯誤。

對於支持多種資產（泛型，PyPI 和 Maven）的包格式，您可以將具有不同名稱的新資產添加到現有的軟件包版本中，假設您具有所需的權限。對於通用套件，只要套件版本處於Unfinished狀態，您就可以新增資產。由於 npm 僅支持每個軟件包版本的單個資產，因此要以任何方式修改已發布的軟件包版本，必須首先使用delete-package-versions.

如果您嘗試重新發佈已存在的資產 (例如，mypackage-1.0.jar)，且已發佈資產的內容與新資產相同，則作業將會成功，因為作業是冪等的。

私有套件和公開儲存庫

CodeArtifact 不會將儲存在儲存 CodeArtifact 庫中的套件發佈到公共儲存庫，例如 npmjs.com 或 Maven 中央。 CodeArtifact 將套件從公用儲存庫匯入至儲 CodeArtifact 存庫，但它永遠不會向其他方向移動套件。您發佈至 CodeArtifact 儲存庫的套件會保持非公開狀態，而且只有您已授與存取權的 AWS 帳戶、角色和使用者才能使用。

發佈修補的套件版本

有時候，您可能會想要發佈修改過的套件版本，可能會在公用存放庫中提供。例如，您可能在一個名為的關鍵應用程式相依性中發現了一個錯誤mydep 1.1，而且您需要比套件廠商檢閱並接受變更的時間更快地修正它。如前所述，如果可透過上游儲 CodeArtifact存庫和外部連線從儲存庫存取公用 CodeArtifact 存放庫，則會 CodeArtifact 防止您在儲存庫mydep 1.1中發佈。

若要解決此問題，請將套件版本發佈到無法 CodeArtifact 存取公用存放庫的其他存放庫。然後使用 copy-package-versions API 將修補的版本複製到您mydep 1.1要從中使用它的 CodeArtifact 存儲庫。

發佈的資產大小限制

可以發佈的封裝資產大小上限會受到中顯示的資產檔案大小上限限制配額 AWS CodeArtifact。例如，您無法發布大於當前資產文件大小最大配額的 Maven JAR 或 Python 輪子。如果您需要在中儲存較大的資產 CodeArtifact，請要求提高配額。

除了資產檔案大小上限配額之外，npm 套件的發佈要求大小上限為 2 GB。此限制獨立於資產文件大小上限配額，並且不能隨著配額增加而提高。在 npm 發佈要求 (HTTP PUT) 中，套件中繼資料和 npm 套件 tar 封存檔的內容會捆綁在一起。因此，可以發行的 npm 套件的實際大小上限會有所不同，而且取決於包含的中繼資料的大小。

> **Note**
>
> 已發佈的 npm 套件限制為小於 2 GB 的大小上限。

發佈延遲

發佈至 CodeArtifact 儲存庫的 Package 版本通常可在不到一秒的時間內下載。例如，如果您將 npm 套件版本發佈到 CodeArtifact withnpm publish，則該版本應該可在不到一秒的時間內供npm install命令使用。但是，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，請使用重試來確保下載可靠。例如，在發佈套件版本之後，如果剛發佈的套件版本在第一次下載嘗試中最初不可用，請重複下載最多三次。

> **Note**
>
> 從公用存放庫匯入套件版本的時間通常會比發行還要長。如需詳細資訊，請參閱 外部連延延。

Package 版本狀態

中的每個套件版本都 CodeArtifact 具有描述套件版本目前狀態和可用性的狀態。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來要求。套件版本將包含在傳回給套件管理員的套件版本清單中，例如，在的輸出中npm view <package-name> versions。套件版本的所有資產都可從儲存庫取得。
- 未完成 — 用戶端已為套件版本上傳一或多個資產，但尚未透過將其移至狀態來完成Published。目前只有泛型和 Maven 軟件包版本可以具有Unfinished. 對於 Maven 軟件包，當客戶端上傳一個或多個軟件包版本的資產，但沒有為包含該版本的軟件包發布maven-metadata.xml文件時，可能會發生這種情況。當 Maven 軟件包版本未完成時，它將不會包含在返回給客戶端的版本列表中gradle，因此不能將其用作構建的一部分。mvn通用包可以通過調用 PublishPackageVersionAPI 時提供unfinished標誌來故意保持在Unfinished狀態。通用包可以通過Published省略unfinished標誌或調用 UpdatePackageVersionsStatusAPI 來更改為狀態。
- 未列出 — 套件版本的資產可從存放庫下載，但套件版本不包含在傳回至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。這意味著 npm 的依賴關係解析邏輯不會選擇軟件包版本，因為該版本不會出現在可用版本列表中。但

是，如果檔案中已參考「未列出」套npm package-lock.json件版本，則仍可下載並安裝該套件版本，例如在執行npm ci時。

- 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至封裝管理員的版本清單中。由於資產無法使用，因此會封鎖用戶端使用套件版本。如果您的應用程式組建取決於已更新為封存的版本，則組建將會中斷，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已封存的套件版本，因為它仍然存在於儲存庫中，但是您可以將套件版本的狀態變更回「未公開」或「UpdatePackageVersionsStatus 已使用 API 發佈」。
- 已處置 — 套件版本不會顯示在清單中，且無法從存放庫下載資產。它們之間的主要區別處置和已歸檔在於，狀態為「已處置」，軟件包版本的資產將被永久刪除。 CodeArtifact因此，您無法將封裝版本從「已處置」移至「已封存」、「未公開」或「已發佈」。無法再使用封裝版本，因為資產已刪除。套件版本標示為「已處置」之後，您將不再需要針對封裝資產的儲存計費。

在沒有--status參數的情況下調 list-package-versions 用時，默認情況下將返回所有狀態的 Package 版本。

除了先前列出的狀態之外，還可以使用 DeletePackageVersionsAPI 刪除套件版本。刪除之後，軟件包版本不再存在於存儲庫中，您可以使用軟件包管理器或構建工具自由地重新發布該軟件包版本。刪除套件版本後，您將不再需要支付該套件版本資產的儲存費用。

Package 名稱、套件版本和資產名稱標準化

CodeArtifact 在儲存套件名稱、套件版本和資產名稱之前將套件名稱標準化，這表示中的名稱或版本 CodeArtifact 可能與套件發佈時提供的名稱或版本不同。如需有關如何在每個套件類型中標準化名稱和版本的 CodeArtifact 詳細資訊，請參閱下列文件：

- 套件名稱規範化
- NuGet軟件包名稱、版本和資產名稱規範化

CodeArtifact 不會對其他套件格式執行標準化。

列出套件名稱

使用中的list-packages指令可 CodeArtifact 取得儲存庫中所有套件名稱的清單。這個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 軟件包名稱

若只要列出 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 (npm 範圍) 中的 npm 套件，請使用`--namespace`和`--format`選項。

Important

`--namespace`選項的值不應包含行距`@`。若要搜尋命名空間`@types`，請將值設定為`##`。

Note

該`--namespace`選項按命名空間前綴進行過濾。任何以傳遞給`--namespace`選項的值開頭的範圍的 npm 套件都會在回`list-packages`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 軟件包名

若只要列出 Maven 套件的名稱，請將`--format`選項的值設定為`maven`。您還必須在`--namespace`選項中指定 Maven 組 ID。

> **Note**
>
> 該`--namespace`選項按命名空間前綴進行過濾。任何以傳遞給`--namespace`選項的值開頭的範圍的 npm 套件都會在回`list-packages`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 件名稱

若只要列出 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
```

按軟件包名稱前綴過濾

若要傳回以指定字串開頭的套件，您可以使用此選`--package-prefix`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`--format`、`--namespace`和`--package-prefix`選項，但本身`--namespace`無法使用。搜尋範圍開頭為`@types`的所有 npm 套件需要指定`--format`選項。`--namespace`單獨使用會導致錯誤。

使用這三個選項中的任何一個也不受支援，`list-packages`且會傳回存放庫中所有格式的所有套件。

格式輸出

您可以使用適用於所有 AWS CLI 指令的參數，使`list-packages`回應變得更緊湊且更具可讀性。使用`--query`參數可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

依預設，傳回的結果數目上限`list-packages`為 100。您可以使用`--max-results`選項來變更此結果限制。

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

允許的最大值`--max-results`為 1,000。若要允許在包含 1,000 個以上套件的儲存庫中列出套件，請使用回應中的`nextToken`欄位`list-packages`支援分頁。如果儲存區域中的套裝程式數目大於的值`--max-results`，您可以`nextToken`將的值傳遞給另一次呼叫，`list-packages`以取得下一頁結果。

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

列出軟件包版本

使用中的`list-package-versions`指令可 AWS CodeArtifact 取得儲存庫中套件名稱所有版本的清單。

```
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。如需有關套件版本狀態的詳細資訊，請參閱Package 版本狀態。

您可以使用`--max-results`和`--next-token`參數`list-package-versions`來分頁回應。對於`--max-results`，請指定介於 1 到 1000 之間的整數，以指定單一頁面中傳回的結果數目。它的默認值是 50。若要傳回後續頁面，請`list-package-versions`再次執行，並將先前指令輸出中接收到的`nextToken`值傳遞給`--next-token`。如果不使用該`--next-token`選項，則始終返回結果的第一頁。

該`list-package-versions`命令不會列出上游存儲庫中的軟件包版本。但是，會列出在套件版本請求期間複製到儲存庫的上游存放庫中封裝版本的參照。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。

列出 npm 軟件包版本

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選項的值不應包含行距@。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 軟件包版本

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。您還必須在--namespace選項中指定 Maven 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可以根據發佈時間以遞減順序排序輸出版本 (最近發佈的版本會先列出)。使用具有值的--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

的傳回值取defaultDisplayVersion決於套件格式：

- 對於通用，Maven 和 PyPI 軟件包，它是最近發布的軟件包版本。
- 對於 npm 軟件包，它是由latest標籤引用的版本。如果未設定latest標籤，則會是最近發佈的套件版本。

格式輸出

您可以使用適用於所有 AWS CLI 指令的參數，使list-package-versions回應變得更緊湊且更具可讀性。使用--query參數可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
```

若要取得更多資訊，請參閱《AWS Command Line Interface 使用指南》AWS CLI中的〈控制指令輸出〉。

列出套件版本資產

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

執行命list-package-version-assets令以傳回有關您 AWS 帳戶中每個資產和目前 AWS 區域的下列資訊：

- 它的名字。
- 它的大小，以字節為單位。
- 一組用於校驗和驗證的哈希值。

例如，使用以下命令列出 Python 包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 軟件包的資產

npm 包始終具有名稱為的單個資產`package.tgz`。要列出範圍 npm 包的資產，請在選項中包含範圍。`--namespace`

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

列出一個 Maven 包的資產

要列出 Maven 包的資產，請在`--namespace`選項中包含包命名空間。要列出 Maven 包的資產`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 與套件版本相關聯的個別.tgz檔案 (例如，npm 檔案或 Maven POM 或 JAR 檔案)。您可以使用下載套件資產get-package-version-assets command。這可讓您在不使用套件管理員用戶端 (例如npm或) 的情況下擷取資產pip。若要下載資產，您必須提供可使用list-package-version-assets指令取得的資產名稱，如需詳細資訊，請參閱列出套件版本資產。資產將使用您指定的檔案名稱下載到本機儲存區。

########## 27.1-JRE.jar ### Maven # COM ######### 27.1-J RE#

```
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
```

############################ guava-27.1-jre.jar############### -27.1-jre.jar#

命令的輸出將是：

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

> **Note**
>
> 若要從範圍的 npm 套件下載資源，請在選項中包含範圍。--namespace使用時必須省略@符號--namespace。例如，如果範圍是@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權限。如需有關以資源為基礎的權限原則的詳細資訊，請參閱AWS Identity and Access Management 使用指南中的以資源為基礎的

在儲存庫間複製套件

您可以將套件版本從一個儲存庫複製到中 CodeArtifact的另一個儲存庫 這對於諸如封裝推進工作流程或在專案團隊或專案之間共用封裝版本等案例非常有用。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

若要在中複製套件版本 CodeArtifact，呼叫使用者必須具有必要的 IAM 許可，且附加至來源和目標儲存庫的資源型政策必須具有必要的權限。如需以資源為基礎的權限原則和 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`必須具有來源儲存庫的`CopyPackageVersions`權限，以及目標儲存庫的權限。`ReadFromRepository`

來源儲存庫必須具有`ReadFromRepository`權限，且目標儲存庫必須具有指派給 IAM 帳戶或使用者複製套件的`CopyPackageVersions`權限。下列原則是使用`put-repository-permissions-policy`指令新增至來源儲存區域或目的地儲存區域的範例儲存區域原則。將 *111122223333* *#####*##的識別碼。`copy-package-versions`

> **Note**
>
> 調用`put-repository-permissions-policy`將替換當前存儲庫策略（如果存在）。您可以使用命`get-repository-permissions-policy`令來查看策略是否存在，如需詳細資訊，請參閱閱讀政策。如果原則確實存在，您可能想要將這些權限新增至其中，而不是取代它。

來源儲存庫權限原則範例

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

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

目標儲存庫權限原則範例

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

複製套件版本

使用中的`copy-package-versions`指令可 CodeArtifact 將一或多個套裝程式版本從來源儲存庫複製到相同網域中的目的地儲存區域。下列範例會將名為的 npm 套件 6.0.2 和 4.0.0 版複製`my-package`到`my_repo`儲存庫。`repo-2`

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

您可以在單一作業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，您必須呼叫每`copy-package-versions`個套件名稱。

前面的命令將產生以下輸出，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

通常，`copy-package-versions`只會在`--source-repository`選項指定的存放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，請在將`includeFromUpstream`參數設定為 true 的情況下呼叫 `CopyPackageVersions` API。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。

複製範圍內的 npm 套件

若要複製範圍中的 npm 套件版本，請使用`--namespace`選項來指定範圍。例如，若要複製封裝`@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 軟件包版本

若要在儲存庫之間複製 Maven 套件版本，請指定要複製的套件，方法是使用選項傳遞 Maven 群組識別碼，並使用`--namespace`選項傳遞 Maven 文件 ID。`--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
```

如果套件版本複製成功，輸出將類似下列內容。

```
{
    "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，則複製將會失敗。如果某些版本存在於來源儲存庫中，而某些版本不存在，則所有版本都將無法複製。在下列範例中，array-uniquenpm 套件的 0.2.0 版本存在於來源儲存庫中，但版本 5.6.7 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
```

在這種情況下，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錯誤碼用於指示版本未複製到目的地儲存庫，因為無法複製其他版本。

目的地儲存庫中已存在的版本

將套件版本複製到已存在的儲存庫時，會 CodeArtifact比較其套件資產和兩個儲存庫中的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，則不會執行副本，但作業會視為成功。這意味著這`copy-package-versions`是冪等的。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`有兩個版本的 npm 套件`repo-1`。版本 0.2.1 也存在於目標存儲庫`dest-repo`和版本 0.2.0 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 0.2.1 0.2.0
```

在這種情況下，輸出將類似於以下內容。

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

版本 0.2.0 列在中，`successfulVersions`因為它已成功從來源複製到目標儲存庫。版本 0.2.1 不會顯示在輸出中，因為它已經存在於目標存儲庫中。

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，複製作業將會失敗。您可以使用`--allow-overwrite`參數強制覆寫。

如果某些版本存在於目標儲存庫中，而某些版本不存在，則所有版本都將無法複製。在下列範例中，`array-uniquenpm` 套件的 0.3.2 版本同時存在於來源和目的地儲存庫中，但套件版本的內容不同。版本 0.2.1 存在於源存儲庫中，但不存在於目標中。

```
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
```

在這種情況下，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 被標記為，SKIPPED因為它沒有複製到目標存儲庫。未複製，因為版本 0.3.2 的複本失敗，因為它已存在於目標儲存庫中，但在來源和目標儲存庫中並不相同。

指定套件版本修訂

套件版本修訂是指定套件版本的一組特定資產和中繼資料的字串。您可以指定封裝版本修訂，以複製處於特定狀態的封裝版本。若要指定封裝版本修訂，請使用--version-revisions參數將一或多個以逗號分隔的封裝版本和封裝版本修訂配對傳遞給命copy-package-versions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my-package版本修訂版本的來源儲存庫中複製 0.3.2 版REVISION-1-SAMPLE-6C81EFF7DA55CC的套件。

```
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
```

下列範例會複製兩個版本的套件my-package，0.3.2 和 0.3.13。只有在 0.3.2 的來源儲存庫版本具有修訂版REVISION-1-SAMPLE-6C81EFF7DA55CC且版本 0.3.13 my-package 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
```

若要尋找封裝版本的修訂版本，請使用describe-package-version或指list-package-versions令。

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

複製 npm 套件

有關 npm 軟件包copy-package-versions行為的更多信息，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指令一次刪除一或多個套件版本。若要從儲存庫中完全移除套件，包括所有相關聯的版本和組態，請使用delete-package指令。套件可以存在於沒有任何套件版本的儲存庫中。當使用delete-package-versions命令刪除所有版本時，或者如果使用 put-package-origin-configuration API 操作未建立任何版本的套件，則可能會發生這種情況 (請參閱編輯套件原點控制項)。

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 軟件包或軟件包版本

- 刪除 Maven 軟件包或軟件包版本

刪除套件 (AWS CLI)

您可以使用delete-package指令刪除套件，包括其所有套件版本和組態。下列範例會刪除網域中儲存庫my-packagemy_repo中指定的 PyPI 套件：my_domain

```
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"
   }
}
```

您可以透過執行describe-package相同的套件名稱來確認套件已刪除：

```
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. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在導覽窗格中，選擇 Repositories (儲存庫)。
3. 選擇您要刪除套裝程式的「儲存區域」。

4. 選擇要刪除的 Package。
5. 選擇「刪除 Package」。

刪除套件版本 (AWS CLI)

您可以使用delete-package-versions指令一次刪除一或多個套件版本。下列範例會刪除網my-package域中名為5.0.0的 PyPI 套件的版本4.0.04.0.1、和：my_repomy_domain

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

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": {}
}
```

您可以通過運行list-package-versions相同的軟件包名稱來確認版本已刪除：

```
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. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在導覽窗格中，選擇 Repositories (儲存庫)。
3. 選擇您要刪除套裝程式版本的「儲存區域」。
4. 選擇您要刪除版本的 Package。
5. 選取您要刪除的 P ackage 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 軟件包或軟件包版本

若要刪除 npm 套件或個別套件版本，請將選`--format`項設定為`npm`。若要刪除範圍 npm 套件中的套件版本，請使用`--namespace`此選項來指定範圍。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 軟件包或軟件包版本

若要刪除 Maven 套件或個別套件版本，請將`--format`選項設定為，`maven`並指定要刪除的套件，方法是將 Maven 群組識別碼與`--namespace`選項和 Maven ArtifacId 傳遞。`--name`例如，下列內容顯示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
```

下列範例會示範如何刪除套件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
```

檢視和更新套件版本詳細資料和相依性

您可以在中檢視有關套件版本的資訊，包括相依性 CodeArtifact。您也可以更新套件版本的狀態。如需有關套件版本狀態的詳細資訊，請參閱Package 版本狀態。

查看軟件包版本詳情

使用命describe-package-version令檢視有關套件版本的詳細資料。Package 版本詳細資訊會在套件發佈至時從封裝中擷取 CodeArtifact。不同軟件包中的詳細信息各不相同，取決於它們的格式以及他們的作者向他們添加了多少信息。

describe-package-version命令輸出中的大多數資訊取決於套件格式。例如，從檔案describe-package-version擷取 npm 套package.json件的資訊。修訂版由建立 CodeArtifact。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，則可以位於同一個儲存庫中。使用選擇性--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 查看 Maven 包版本詳細信息 。

下列範例會傳回my_repo儲存庫中名為pyhamcrest之 Python 套件版本1.9.0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"
}
```

檢視 npm 套件版本詳細資訊

若要檢視關於 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 scope)。`--namespace`選項的值不應包含行距`@`。若要搜尋命名空間`@types`，請將值設定為`##`。

以下返回有關`@types`範圍`webpack`中命名`4.41.5`的 npm 包的版本的詳細信息。

```
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"
  }
}
```

查看 Maven 包版本詳細信息

若要檢視有關 Maven 套件版本的詳細資訊，請將`--format`選項的值設定為，`maven`並在選`--namespace`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`儲存庫中名為`commons-rng-client-api`的 Maven 套件版本`1.2`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 檔案中解壓縮套件版本詳細資訊。給定軟件包版本的元數據將僅包含該確切軟件包版本的 POM 中的信息，而不包括父 POM 或使用 POM 標籤傳遞引用的任何其他 POM 的信息。`parent`這表示的輸出`describe-package-version`會忽略依賴`parent`參考來包含此中繼資料的 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`指令取得套件版本相依性的清單。下列命令會列出`my_repo`儲存庫中`my_domain`網域`my-package`中名為 version `4.41.5` 的 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「相依類型」欄位支援的值範圍，請參閱 API 中的PackageDependency資料類型。CodeArtifact

檢視套件版本讀我檔

某些套件格式 (例如 npm) 會包含README檔案。使用get-package-version-readme取得套README件版本的檔案。下列命令會傳回my_repo儲存庫中my_domain網域中名為 my-package version 4.41.5 的 npm 套README件檔案。

> **Note**
>
> CodeArtifact 不支持顯示來自通用或 Maven 軟件包的自述文件。

```
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

中的每個套件版本都 CodeArtifact 具有描述套件版本目前狀態和可用性的狀態。您可以同時使用和控制台變更套件版本狀態。 AWS CLI

> **Note**
>
> 如需有關封裝版本狀態的詳細資訊，包括可用狀態的清單，請參閱Package 版本狀態。

更新套件版本狀態

設定套件版本的狀態可讓您控制套裝軟體版本的使用方式，而不必從存放庫中完全刪除套件版本。例如，當套件版本的狀態為時Unlisted，它仍然可以正常下載，但它不會出現在傳回指令的套件版本清單中，例如npm view。該 UpdatePackageVersionsStatus API 允許在單個 API 調用中設置同一包的多個版本的軟件包版本狀態。有關不同狀態的描述，敬請參閱套件概觀。

使用update-package-versions-status指令將套件版本的狀態變更為PublishedUnlisted、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 版的狀態設定chalk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，但該命令對其他格式的運作方式完全相同。使用單一指令可以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，封裝版本就無法移回Unfinished狀態，因此不允許此狀態作為--target-status參數的值。若要將套件版本移至狀Disposed態，請改用dispose-package-versions指令，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，您必須擁有封裝資源的codeartifact:UpdatePackageVersionsStatus權限。這表示您可以授予每個套update-package-versions-status件呼叫的權限。例如，授予在 npm 套件 *chalk* *update-package-versions-status* 上呼叫權限的 IAM 政策將包含類似下列的陳述式。

```
{
  "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 軟件包的狀態

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，若要取消刊登的 8.0.0 版@nestjs/core，請使用下列指令。

```
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 包的狀態

Maven 包始終具有一個組 ID，在中稱為命名空間 CodeArtifact。呼叫時，請使用`--namespace`參數來指定 Maven 群組識別碼`update-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

套件版本修訂是指定套件版本的一組特定資產和中繼資料的字串。您可以指定封裝版本修訂，以更新處於特定狀態的套件版本狀態。若要指定封裝版本修訂，請使用`--version-revisions`參數來傳遞一或多個以逗號分隔的封裝版本和封裝版本修訂配對。只有在套件版本的目前修訂版本符合指定的值時，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
```

若要使用單一指令更新多個版本，請將以逗號分隔的版本和版本修訂配對清單傳遞至選`--version-revisions`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配對。

```
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`必須與傳遞給的版本相同`--versions`。如果修訂指定不正確，則該版本的狀態將不會更新。

使用預期的狀態參數

該`update-package-versions-status`命令提供支持指定軟件包版本的預期當前狀態的`--expected-status`參數。如果目前狀態與傳遞給的值不符`--expected-status`，則不會更新該套件版本的狀態。

例如，在 *my_repo* 中，npm 套件的 4.0.0 版和 4.1.0 版`chalk`目前的狀態為. Published 呼叫指定`update-package-versions-status`的預期狀態`Unlisted`將無法更新這兩個套件版本，因為狀態不相符。

```
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"
        }
    }
}
```

個別套件版本發生錯誤

調用時軟件包版本的狀態不會更新有多種原因update-package-versions-status。例如，套件版本修訂可能指定不正確，或者預期的狀態與目前狀態不符。在這些情況下，版本將包含在 API 回應中的failedVersions地圖中。如果某個版本失敗，則在同一個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。這樣的版本也將包含在failedVersions地圖errorCode中SKIPPED。

在的目前實作中update-package-versions-status，如果一或多個版本的狀態無法變更，則會略過所有其他版本。也就是說，所有版本都已成功更新或未更新任何版本。API 合約無法保證此行為；future，某些版本可能會成功，而其他版本在單次呼叫時失敗update-package-versions-status。

下列範例指令包含封裝版本修訂不符所造成的版本狀態更新失敗。該更新失敗會導致另一個版本狀態更新呼叫被跳過。

```
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套件狀態的行為與類似Archived，不同之處在於套件資產將被永久刪除， CodeArtifact 如此一來，網域擁有者的帳戶就不會再為資產儲存計費。如需有關每個套件版本狀態的詳細資訊，請參閱Package 版本狀態。若要將套件版本的狀態變更為Disposed，請使用dispose-package-versions指令。這項功能與處置套件版本是不同的，update-package-versions-status因為處置套件版本是無法復原的。由於封裝資產將會遭到刪除，因此版本的狀態無法變更回ArchivedUnlisted、或Published。可對已處置的封裝版本執行的唯一動作是使用delete-package-versions指令將其刪除。

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

dispose-package-versions指令的行為與類似update-package-versions-status，包括版本修訂版本--version-revisions 和預期狀態區段中所描述的和--expected-status選項的行為。例如，下列命令嘗試處置套件版本，但因為預期狀態不符而失敗。

```
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"
        }
    }
}
```

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

```
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": {}
}
```

編輯套件原點控制項

在中 AWS CodeArtifact，您可以直接發佈套件版本、從上游存放庫下拉，或從外部公用存放庫擷取套件版本，將套件版本新增至存放庫。允許透過直接發佈和從公用儲存庫擷取來新增套件的套件版本，使您容易遭受相依性替換攻擊。如需詳細資訊，請參閱 依賴替換攻擊。若要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來源控制項，以限制如何將該套件的版本新增至儲存庫。

任何想要允許不同套件的新版本同時來自內部來源 (例如直接發佈) 和外部來源 (例如公用儲存庫) 的團隊，都應該考慮設定套件來源控制項。根據預設，套件原始控制項會根據第一個套件版本新增至儲存庫的方式來設定。如需套件原始控制項設定及其預設值的相關資訊，請參閱〈〉Package 原點控制設定。

若要在使用 `put-package-origin-configuration` API 作業之後移除套件記錄，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

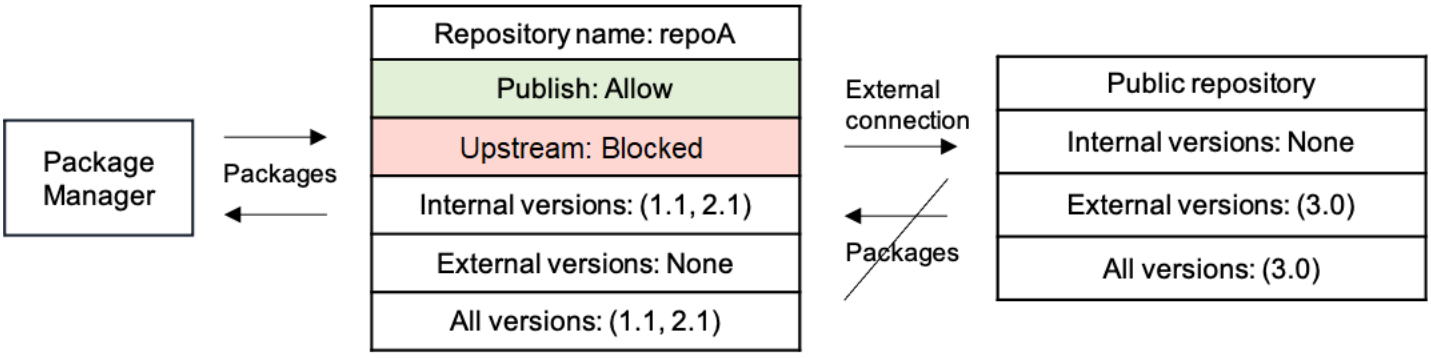
本節包括將套件版本新增至 CodeArtifact 儲存庫時的一些常見案例。將根據第一個 Package 件版本的新增方式，為新套件設定套件來源控制設定。

在下列案例中，內部套件是直接從套件管理員發行至存放庫的封裝，例如您或您的小組撰寫並維護的套件。外部套件是存在於公用存放庫中的套件，可透過外部連線擷取到儲存庫中。

針對現有內部套件發行外部套件版本

在這個案例中，請考慮一個內部套件，PackaGa。您的團隊會將 PackaGea 的第一個 CodeArtifact 套件版本發佈到儲存庫中。因為這是該套件的第一個套件版本，所以套件原始控制項設定會自動設定為「發佈:允許」和「上游:區塊」。套裝程式存在於您的儲存庫中之後，會將具有相同名稱的套件發行至連線至存放庫的公用儲存 CodeArtifact 庫。這可能是針對內部軟件包的嘗試依賴替換攻擊，也可能只是巧合。無論如何，套件來源控制項都設定為封鎖新外部版本的擷取，以保護自己免受潛在攻擊。

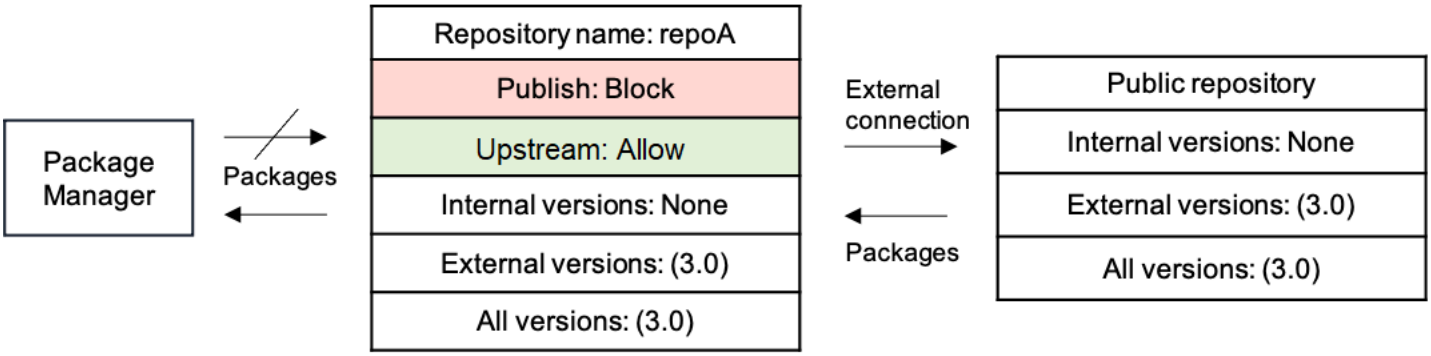
在下圖中，RePoA 是具有與公共存 CodeArtifact 儲庫的外部連接的存儲庫。您的存儲庫包含的版本 1.1 和 2.1 的 PackaGa，但 3.0 版本已發佈到公共存儲庫。一般而言，RePoA 會在套件管理員要求套件之後擷取 3.0 版。由於套件擷取設定為「封鎖」，因此 3.0 版本不會擷取到您的 CodeArtifact 儲存庫中，而且無法提供給與其連線的套件管理員。



針對現有外部套件發行內部套件版本

在這個案例中，PackageB 外部存在於您已連線至存放庫的公用存放庫中的套件。當連接到儲存庫的套件管理員要求 PackageB 時，套件版本會從公用存放庫擷取到您的存放庫中。因為這是第一個套件版本的 PackageB 新增至您的儲存庫，因此套件原始設定會設定為「發佈:區塊」和「上游:允許」。稍後，您嘗試將具有相同套件名稱的版本發佈到存放庫。您可能不知道公開套件，並嘗試以相同名稱發佈不相關的套件，或者您嘗試發佈修補的版本，或者您嘗試直接發佈外部已存在的套件版本。CodeArtifact 將拒絕您嘗試發布的版本，但允許您明確覆蓋拒絕並在必要時發布版本。

在下圖中，RePoA 是具有與公共存 CodeArtifact 儲庫的外部連接的存儲庫。您的存放庫包含從公用存放庫擷取的 3.0 版本。您想要將 1.1 版發佈至您的儲存庫。通常，您可以將 1.2 版發佈至 RePoA，但由於發佈設定為「封鎖」，因此無法發佈 1.2 版。



發佈現有外部套件的修補套件版本

在這個案例中，PackageB 外部存在於您已連線至存放庫的公用存放庫中的套件。當連接到儲存庫的套件管理員要求 PackageB 時，套件版本會從公用存放庫擷取到您的存放庫中。因為這是第一個套件版本的 PackageB 新增至您的儲存庫，因此套件原始設定會設定為「發佈:區塊」和「上游:允許」。您的團隊決定需要將此套件的已修補套件版本發佈到存放庫。為了能夠直接發佈套件版本，您的團隊將套件原始控制項設定變更為「發佈:允許」和「上游:BLOCK」。此套件的版本現在可以直接發佈至您的儲存庫，並從公用存放庫擷取。在您的團隊發佈已修補的套件版本之後，您的團隊會將套件來源設定還原為「發佈:BLOCK」和「上游:允許」。

Package 原點控制設定

使用套件來源控制項，您可以設定如何將套件版本新增至儲存庫。下列清單包括可用的套件原點控制設定和值。

> **Note**
>
> 在套件群組上設定原始控制項時，可用的設定和值會有所不同。如需詳細資訊，請參閱 Package 群組原點控制項。

發布

此設定可設定是否可以使用套裝程式管理員或類似工具將套件版本直接發佈至儲存庫。

- 允許：Package 版本可以直接發布。
- BLOCK：Package 版本不能直接發布。

上游

此設定可設定套件版本是否可以從外部公用儲存庫擷取，或是在套件管理員要求時從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 CodeArtifact 儲存庫的其他儲存庫中保留，或從具有外部連線的公用來源擷取的套件版本。
- BLOCK：無法從設定為上游儲存庫的其他 CodeArtifact 儲存庫保留 Package 版本，或從具有外部連線的公用來源擷取套件版本。

預設套件原始碼控制設定

預設的套件原始控制設定是根據套件的關聯套件群組的原始控制設定來設定。如需有關套件群組和套件群組原始控制項的詳細資訊，請參閱使用套件群組 CodeArtifact和Package 群組原點控制項。

如果套裝程式與某個套裝程式群組產生關聯，且其限制設定值ALLOW為每種限制類型，則套裝程式的預設套件原始控制項會以該套裝程式的第一個版本新增至儲存庫的方式為基礎。

- 如果第一個軟件包版本由軟件包管理器直接發布，則設置將是「發布：允許」和「上游：塊」。
- 如果第一個套件版本是從公開來源擷取，則設定會是「發佈:區塊」和「上游:允許」。

> **Note**
>
> 在 2022 年 5 月左右之前存在於 CodeArtifact 儲存庫中的套件，將具有「發佈：允許」和「上游：允許」的預設套件原始控制項。此類 Package 件必須手動設定套件原始碼控制項。目前的預設值已在新套件上設定，並於 2022 年 7 月 14 日啟動此功能時開始強制執行。如需有關設定套件原始控制項的詳細資訊，請參閱編輯套件原點控制項。

否則，如果某個套件與至少具有一個限制設定的套件群組相關聯BLOCK，則該套件的預設原始控制設定將設定為「發佈:允許」和「上游:LOW」。

套件原始控制項如何與套件群組原始控制項互動

由於套件具有原始控制設定，而且它們的關聯套件群組有原始控制設定，所以瞭解這兩個不同的設定如何互動是非常重要的。

這兩個設定之間的互動是設定BLOCK永遠勝過的設定ALLOW。下表列出了一些範例組態及其有效的原點控制設定。

Package 原點控制設定	Package 群組原點控制設定	有效的原點控制設定
發佈:允許 上游：允許	發佈:允許 上游：允許	發佈:允許 上游：允許
發佈:圖塊 上游：允許	發佈:允許 上游：允許	發佈:圖塊 上游：允許
發佈:允許 上游：允許	發佈:允許 上游：塊	發佈:允許 上游：塊

這意味著一個包的原始設置為 Publish：ALLO W 和上游：LO W，然後它有效地延遲到關聯的軟件包組的原始控制設置。

編輯套件原點控制項

Package 原始控制項會根據套件的第一個套件版本新增至存放庫的方式自動設定，如需詳細資訊，請參閱預設套件原始碼控制設定。若要新增或編輯 CodeArtifact 儲存區域中套裝程式的套裝程式原始控制項，請執行下列程序中的步驟。

若要新增或編輯套件原始控制項 (主控台)

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇「存放庫」，然後選擇包含您要編輯之套件的存放庫。
3. 在「封裝」表格中，搜尋並選取您要編輯的封裝。
4. 在套件摘要頁面的 Origin 控制項中，選擇編輯。
5. 在 「編輯原點控制項」中，選擇您要為此套件設定的套件原始控制項。必須同時設定套件原始控制項設定 (「發佈」和「上游」)。
   - 若要允許直接發佈封裝版本，請在 [發佈] 中選擇 [允許]。若要封鎖套件版本的發佈，請選擇 [封鎖]。
   - 若要允許從外部儲存庫擷取套裝程式並從上游儲存庫提取套裝程式，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套件版本，請選擇「封鎖」。

若要新增或編輯套件原始控制項 (AWS CLI)

1. 如果您尚未設定，請 AWS CLI 依照中的步驟進行設定設定使用 AWS CodeArtifact。
2. 使用指put-package-origin-configuration令新增或編輯套件原始控制項。取代下列欄位：
   - 將 *my_domain* 取代為包含您要更新之套件的 CodeArtifact 網域。
   - 將 *my_repo* 取代為包含您要更新之套件的 CodeArtifact 儲存庫。
   - 將 *npm* 取代為您要更新的套件的套件格式。
   - 將 *my_package* 取代為您要更新的套件名稱。
   - 將 *LOW* 和 *BLOCK* 替換為您想要的軟件包源控制設置。

```
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 不允許發布可訪問的上游存儲庫或公共存儲庫中存在的軟件包版本。例如，假設您想要將 Maven 套件發佈`com.mycompany.mypackage:1.0`到儲存庫`myrepo`，並且`myrepo`具有與 Maven Central 外部連線的上游儲存庫。請考慮下列案例。

1. 上的套件原始控制項設定`com.mycompany.mypackage`為「發佈：允許」和「上游：允許」。如果`com.mycompany.mypackage:1.0`存在於上游存儲庫或 Maven Central 中，則 CodeArtifact 拒絕任何發布到它的嘗試，並出現 409 衝突錯誤。`myrepo`您仍然可以發佈不同的版本，例如`com.mycompany.mypackage:1.1`.
2. 上的套件原始控制項設定`com.mycompany.mypackage`為「發佈:允許」和「上游:區塊」。您可以將任何版本的發佈`com.mycompany.mypackage`到尚未存在的存放庫，因為無法存取套件版本。
3. 上的套件原始控制項設定`com.mycompany.mypackage`為「發佈:區塊」和「上游:允許」。您無法將任何套件版本直接發佈到存放庫。

使用套件群組 CodeArtifact

P@@ ackage 件群組可用來將組態套用至符合使用套件格式、套件命名空間和套件名稱的已定義模式的多個套件。您可以使用套件群組，更方便地為多個套件設定套件原始碼控制項。Package 來源控制項可用來封鎖或允許擷取或發佈新套件版本，以保護使用者免於遭受稱為相依性替代攻擊的惡意動作。

中的每個網域都 CodeArtifact 會自動包含一個根套件群組。此根套件群組包含所有套裝程式，並允許套件版本依預設從所有原始類型輸入網域中的儲存庫。/*您可以修改根封裝群組，但無法刪除。

這些主題包含中封裝群組的相關資訊 AWS CodeArtifact。

主題

建立套件群組

您可以使用 CodeArtifact 控制台 AWS Command Line Interface (AWS CLI) 或建立套件群組 AWS CloudFormation。如需有關使用管理 CodeArtifact 封裝群組的詳細資訊 CloudFormation，請參閱建立 CodeArtifact 資源 AWS CloudFormation。

建立套件群組（主控台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [網域]，然後選擇要在其中建立封裝群組的網域。
3. 選擇「Package 群組」，並選擇「建立套件群組」。
4. 在 P ackage 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封裝群組定義，也可以使用視覺化模式進行選取，並自動建立封裝群組定義。

5. 若要使用視覺化模式來建立套件群組定義：

   a. 選擇「視覺」以切換至視覺模式。

   b. 在「P ackage」格式中，選擇要與此群組關聯的套件格式。

   c. 在命名空間（範圍）中，選擇要匹配的命名空間條件。

      - 等於：完全匹配指定的命名空間。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 word 開頭：匹配以指定單詞開頭的命名空間。如果選擇，請輸入要符合的前置字。若要取得有關單字和單字邊界的更多資訊，請參閱單詞，單詞邊界和前綴匹配。
      - 全部：比對所有命名空間中的套件。

   d. 如果選取了「等於」、「空白」或「以文字開頭」，請在 P ackage 件名稱中選擇要符合的套件名稱條件。

      - 完全等於：完全匹配指定的軟件包名稱。如果選擇，請輸入要符合的套件名稱。
      - 以前綴開頭：匹配以指定前綴開頭的軟件包。
      - 以 word 開頭：比對以指定字詞開頭的套件。如果選擇，請輸入要符合的前置字。若要取得有關單字和單字邊界的更多資訊，請參閱單詞，單詞邊界和前綴匹配。
      - 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 P ackage 群組定義中，輸入套件群組定義。如需封裝群組定義語法的詳細資訊，請參閱Package 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。檢閱之後，選擇 [下一步]。

8. 在「P ackage」群組資訊中，選擇性地新增封裝群組的說明和連絡人電子郵件。選擇下一步。

9. 在 P ackage 件原始控制項中，設定要套用至群組中套件的原始控制項。如需套件群組原始控制項的詳細資訊，請參閱Package 群組原點控制項。

10. 選擇「建立套件群組」。

建立套件群組 (AWS CLI)

使用指`create-package-group`令在您的網域中建立套件群組。針對此`--package-group`選項，請輸入決定哪些套件與群組相關聯的套件群組定義。如需封裝群組定義語法的詳細資訊，請參閱Package 群組定義語法和相符行為。

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

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

檢視或編輯套件群組

您可以使用 CodeArtifact 主控台或 AWS Command Line Interface (AWS CLI) 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料或組態。

檢視或編輯套件群組（主控台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [網域]，然後選擇包含您要檢視或編輯之封裝群組的網域。
3. 選擇「P ackage 群組」，然後選擇要檢視或編輯的套件群組。
4. 在「詳細資料」中，檢視套件群組的相關資訊，包括其上層群組、說明、ARN、聯絡人電子郵件和套件原始控制項。
5. 在子群組中，檢視將此群組作為上層群組的封裝群組清單。此清單中的套件群組可以繼承此套件群組的設定。如需詳細資訊，請參閱 Package 群組階層與模式特異性。
6. 在封裝中，根據套裝程式群組定義檢視屬於此套裝程式群組的套件。在「強度」欄中，您可以看到封裝關聯的強度。如需詳細資訊，請參閱 Package 群組階層與模式特異性。
7. 若要編輯封裝群組資訊，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。您無法編輯封裝群組的定義。

b. 在 P ackage 件群組原始控制項中，更新套件群組的原始控制設定，以決定關聯的套件如何進入網域中的儲存庫。如需詳細資訊，請參閱 Package 群組原點控制項。

檢視或編輯套件群組 (AWS CLI)

使用下列指令來檢視或編輯套件群組 AWS CLI。如果您尚未設定，請 AWS CLI 依照中的步驟進行設定設定使用 AWS CodeArtifact。

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

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

若要檢視有關封裝群組的詳細資訊，請使用`describe-package-group`指令。如需封裝群組定義的詳細資訊，請參閱Package 群組定義語法與範例。

```
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`指令。

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

若要檢視與封裝相關聯的封裝群組，請使用`get-associated-package-group`指令。您必須針對 Python 和 Swift 套件格式使用標準化套件名稱和命名空間。 NuGet有關軟件包名稱和命名空間如何標準化的更多信息，請參閱 NuGet，Python 和 Swift 名稱標準化文檔。

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

```
        --namespace scope
```

若要編輯封裝群組，請使用update-package-group指令。此指令用於更新套件群組的聯絡資訊或描述。如需有關套件群組原始控制設定以及新增或編輯這些設定的資訊，請參閱Package 群組原點控制項。如需封裝群組定義的詳細資訊，請參閱 Package 群組定義語法與範例

```
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 控制台或 AWS Command Line Interface (AWS CLI) 刪除套件群組。

刪除套件群組時，請注意下列行為：

- 無法刪除根封裝群組。/*
- 與該套件群組相關聯的套件和套件版本不會被刪除。
- 刪除封裝群組時，直接子套件群組將成為封裝群組之直接父封裝群組的子系。因此，如果有任何子群組繼承父系的任何設定，則這些設定可能會變更。

刪除套件群組（主控台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [網域]，然後選擇包含您要檢視或編輯之封裝群組的網域。
3. 選擇「Package 群組」。
4. 選擇您要刪除的套件群組，然後選擇「刪除」。
5. 在欄位中輸入刪除，然後選擇「刪除」。

刪除套件群組 (AWS CLI)

若要刪除套件群組，請使用delete-package-group指令。

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

Package 群組原點控制項

Package 原始控制項是用來設定套件版本如何進入網域。您可以在套裝程式群組上設定原始控制項，以設定與套裝程式群組關聯的每個套件版本如何在網域中輸入指定的儲存庫。

Package 群組原始控制設定包含下列項目：

- 限制設定：這些設定定義套裝程式是否可以 CodeArtifact 從發佈、內部上行或外部公用儲存庫中輸入儲存區域。
- 允許存放庫清單：每個限制設置可以設置為允許特定的存儲庫。如果將限制設定設定為允許特定存放庫，則該限制將具有對應的允許存放庫清單。

> **Note**
>
> 套件群組的原始控制設定與個別套件的原始控制設定略有不同。如需套件的原始控制設定的詳細資訊，請參閱Package 原點控制設定。

限制設定

套裝程式群組的原始控制設定值會決定與該群組關聯的套裝程式如何進入網域中的儲存區域。

發佈

此設定可PUBLISH設定是否可以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存放庫。

- 允許：Package 版本可以直接發佈到所有儲存庫。
- BLOCK：Package 版本不能直接發佈到任何存儲庫。
- ALLOW_SPECION_LATIONS：Package 版本只能直接發佈至允許的儲存庫清單中指定的儲存庫以進行發佈。

- 繼承：此PUBLISH設定會繼承自第一個父套件群組，其設定不是INHERIT。

外部上游

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

- 允許：任何套件版本都可以從具有外部連線的公開來源擷取到所有儲存庫中。
- BLOCK：Package 版本無法從具有外部連線的公開來源擷取到任何儲存庫中。
- ALLOW_SPECION_LAT EGORS：Package 版本只能從公用來源擷取至外部上行串流允許的儲存庫清單中指定的儲存庫。
- 繼承：此EXTERNAL_UPSTREAM設定會繼承自第一個父套件群組，其設定不是INHERIT。

內部 _ 上游

此INTERNAL_UPSTREAM設定會設定套件管理員要求時，是否可以從相同 CodeArtifact 網域中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 CodeArtifact 儲存庫的其他儲存庫中保留。
- BLOCK：無法從配置為上游儲存庫的其他 CodeArtifact 儲存庫中保留 Package 版本。
- ALLOW_ CodeArtifact SPECION_LATEGORS：Package 版本只能從設定為上游儲存庫的其他存放庫保留至內部上游允許的儲存庫清單中指定的儲存庫中。
- 繼承：此INTERNAL_UPSTREAM設定會繼承自第一個父套件群組，其設定不是INHERIT。

允許存放庫清單

將限制設定設定為時ALLOW_SPECIFIC_REPOSITORIES，套裝程式群組會隨附一份允許的儲存庫清單，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 個允許存放庫清單的任何位置，每個配置為的設定都有一個ALLOW_SPECIFIC_REPOSITORIES。

當您將儲存區域新增至套裝程式群組的允許存放庫清單時，您必須指定要將其新增至哪個允許的儲存庫清單。

可能的允許存放庫清單如下：

- EXTERNAL_UPSTREAM：允許或攔截從新增儲存庫中的外部儲存庫擷取套件版本。

- INTERNAL_UPSTREAM：允許或阻止從添加的存儲庫中的另一個 CodeArtifact 存儲庫中提取軟件包版本。
- PUBLISH：允許或攔截套件版本從套件管理員直接發佈到新增的儲存庫。

編輯套件群組原點控制設定

若要新增或編輯套件群組的原始控制項，請執行下列程序中的步驟。如需套件群組原始控制項設定的詳細資訊，請參閱限制設定和允許存放庫清單。

新增或編輯套件群組原始控制項 (CLI)

1. 如果您尚未設定，請 AWS CLI 依照中的步驟進行設定設定使用 AWS CodeArtifact。
2. 使用指update-package-group-origin-configuration令新增或編輯套件原始控制項。

   - 對於--domain，輸入包含您要更新之套件群組的 CodeArtifact 網域。
   - 針對--domain-owner，輸入網域擁有者的帳號。
   - 對於--package-group，輸入您要更新的套件群組。
   - 在中--restrictions，輸入代表原點控制限制的鍵值對。
   - 在中--add-allowed-repositories，輸入 JSON 物件，其中包含限制類型和存放庫名稱，以新增至對應的允許存放庫清單以供限制使用。
   - 在中--remove-allowed-repositories，輸入 JSON 物件，其中包含限制類型和存放庫名稱，以便從對應的允許存放庫清單中移除以進行限制。

```
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
```

下面的例子在一個命令中添加了多個限制和多個存儲庫。

```
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
```

Package 群組原始控制項組態範例

下列範例顯示常見套件管理案例的套件原始控制組態。

允許發行具有私人名稱的套件，但不能擷取

這種情況很可能是套件管理中的常見案例：

- 允許將包含私人名稱的套件從套件管理程式發佈到您網域中的儲存庫，並阻止它們從外部公用儲存庫擷取到您網域中的儲存庫。
- 允許所有其他套件從外部公用儲存庫擷取到您網域中的儲存庫，並阻止套件管理員將它們發佈到您網域中的儲存庫。

要實現這一目標，您應該配置一個包含私有名稱和原始設置的模式的軟件包組：發布：允許，外部 _ 上游：塊和內部 _ 上游：允許。這樣可以確保具有私人名稱的套件可以直接發佈，但無法從外部儲存庫中擷取。

下列 AWS CLI 指令會建立並設定包含符合所需行為的來源限制設定的套件群組：

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

在這個案例中，您的網域有多個儲存庫。在這些存儲庫中，repoA具有與公共存儲庫的上游連接repoB，該連接到公共存儲庫具有外部連接npmjs.com，如圖所示：

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。您也想要封鎖擷取與套裝程式群組關聯至網域中任何其他儲存庫的套件，並封鎖套件管理員直接發佈套件。若要達成此目的，您可以建立並設定封裝群組，如下所示：

「發布：塊」和「外部 _ 上游：允許 _ 特定存儲庫」和「內部 _ 上游：允許特定存儲庫」的原始限制設置。

repoA並repoB新增至適當的允許存放庫清單：

- repoA應該被添加到INTERNAL_UPSTREAM列表中，因為它會從其內部上游獲取軟件包repoB。
- repoB應該被添加到EXTERNAL_UPSTREAM列表中，因為它會從外部存儲庫中獲取軟件包npmjs.com。

下列 AWS CLI 指令會建立並設定包含符合所需行為的來源限制設定的套件群組：

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
```

套件群組原始控制項設定如何與套件原始控制項設定互動

由於套件具有原始控制設定，而且它們的關聯套件群組有原始控制設定，所以瞭解這兩個不同的設定如何互動是非常重要的。若要取得有關設定之間互動的資訊，請參閱套件原始控制項如何與套件群組原始控制項互動。

Package 群組定義語法和相符行為

本主題包含有關定義封裝群組、模式比對行為、封裝關聯強度和套件群組階層的資訊。

內容

Package 群組定義語法與範例

定義套裝軟體群組的模式語法會緊跟套件路徑的格式。套件路徑是從套件的座標元件 (格式、命名空間和名稱) 建立的，方法是在開頭加上正斜線，並以正斜線分隔每個元件。例如，在命名空間anycompany-ui-components中命名的 npm 套件的套件路徑是 /npm/ space/anycompany-ui-components。

套裝軟體群組模式遵循與封裝路徑相同的結構，除了省略未指定為群組定義一部分的元件，且樣式會以字尾結束。包含的後綴決定了模式的匹配行為，如下所示：

- 字$尾將符合完整的封裝座標。
- ~後綴將匹配前綴。
- *後綴將匹配先前定義的組件的所有值。

以下是每個允許組合的示例模式：

1. 所有套件格式：/*
2. 一個特定的包格式：/npm/*
3. Package 格式和命名空間前綴：/maven/com.anycompany~
4. Package 格式和命名空間：/npm/space/*
5. Package 格式、命名空間和名稱前置詞：/npm/space/anycompany-ui~
6. Package 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前置符合項目，並在用*來比對路徑中下一個元件的所有值 (無論是所有格式、所有命名空間或所有名稱) 時，會在正斜線之後加上。

Package 群組定義與規範化

CodeArtifact 標準化 NuGet Python 和 Swift 包名稱，並在存儲它們之前標準化 Swift 包命名空間。CodeArtifact 將套件與套件群組定義相符時，會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的命名空間和名稱。有關軟件包名稱和命名空間如何標準化的更多信息，請參閱 NuGet，Python 和 Swift 名稱標準化文檔。

套件群組定義中的命名空間

對於沒有命名空間 (Python 和 NuGet) 的套件或套件格式，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的命名空間區段。例如，名為請求的 Python 包的路徑是 /python// 請求。

對於具有命名空間（Maven，泛型和 Swift）的包或包格式，如果包含包名稱，則必須包含命名空間。對於斯威夫特包格式，規範化的包命名空間將被使用。如需 Swift 套件命名空間如何標準化的詳細資訊，請參閱。Swift 包名稱和命名空間規範化

Package 群組階層與模式特異性

「in」或「關聯」套件群組的套件是包含與群組模式相符的套件路徑，但不符合更具體群組的模式。例如，給定了包組*/npm/space/**，*/npm/**並且包路徑 /npm//反應與第一組（ ）相關聯，而 /npm/ 空間/ aui.組件和/npm/空間/ 與第二個組（*/npm/**）相關聯。amplify-ui-core /npm/space/*即使一個套件可能符合多個群組，但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設定適用於套件。

當一個包路徑匹配多個模式時，「更具體」的模式可以被認為是最長的匹配模式。或者，更具體的模式是符合較不特定模式的套件適當子集的模式。從我們前面的例子中，每個匹配的軟件包/npm/space/*也匹配/npm/*，但反過來不是真的，這使得/npm/space/*更具體的模式，因為它是一個適當的子集/npm/*。由於某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裝軟體群組組態適用於套件，但該群組可能會設定為繼承其上層群組的組態。

單詞，單詞邊界和前綴匹配

在討論前綴匹配之前，讓我們定義一些關鍵術語：

- 一個字母或數字後跟零個或多個字母，數字或標記字符（例如重音符號，變音符號等）。
- 當到達非單詞字符時，單詞邊界位於單詞的末尾。非單字字元是標點符號字元，例如.-、和。_

具體來說，一個單詞的正則表達式模式是[\p{L}\p{N}][\p{L}\p{N}\p{M}]*，可以按如下方式細分：

- \p{L}代表任何字母。
- \p{N}代表任何數字。
- \p{M}代表任何標記字符，例如重音符號，變音符號等

因此，[\p{L}\p{N}]表示一個數字或字母，並[\p{L}\p{N}\p{M}]*表示零個或多個字母，數字或標記字符和一個單詞邊界是在這個正則表達式模式的每個匹配的末尾。

> **Note**
>
> 文字邊界比對是以「字」的這個定義為基礎。它不是基於字典中定義的單詞，或 CameCase。例如，oneword或中沒有文字邊界OneWord。

現在定義了單詞和單詞邊界，我們可以使用它們來描述中的前綴匹配 CodeArtifact。為了指示字邊界上的前綴匹配，在單詞字符後面使用匹配字符（~）。例如，該模式/npm/space/foo~匹配包路徑/npm/space/foo和/npm/space/foo-bar，但不匹配/npm/space/food或/npm/space/foot。

必須使用萬用字元 (*)，而不是跟隨非單字字元 (例如在樣式/npm/*中)。~

區分大小寫

Package 群組定義區分大小寫，也就是說，只有大小寫不同的模式可以作為個別的套件群組存在。例如，用戶可以使用模式創建單獨的軟件包組 /npm//AsyncStorage$/npm//asyncStorage$，以及/npm//asyncstorage$對於 npm 公共註冊表中存在的三個單獨的軟件包：AsyncStor age AsyncStorage，僅因大小寫而異的異步存儲。

儘管案例很重要，但如果套件的模式有不同的變化，則 CodeArtifact 仍會將套件與套件群組產生關聯。如果用戶在沒有創建上面顯示的其他兩個組的情況下創建了/npm//AsyncStorage$軟件包組，那麼名稱的所有大小寫變化 AsyncStorage，包括 AsyncStorage 和 asyncstorage 都將與軟件包組相關聯。但是，如下一節所述強弱匹配，這些變化的處理方式與完全符合模式的方式不同。AsyncStorage

強弱匹配

上一節中的資訊指出封裝群組區分大小寫，然後繼續說明它們不區分大小寫。區分大小寫這是因為中的包組定義 CodeArtifact 具有強匹配（或完全匹配）和弱匹配（或變體匹配）的概念。一個強大的匹配是當包裝完全匹配的模式，沒有任何變化。一個弱的匹配是當包匹配的模式的變化，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整為更一般的套件群組。當套件是最特定相符群組模式的變體 (弱比對) 時，套件會與群組產生關聯，但套件會遭到封鎖，而不是套用群組的原始控制組態，以防止套件的任何新版本從串流上移或發佈。這種行為可降低因名稱幾乎相同的套件相依性混淆而造成供應鏈攻擊的風險。

為了說明弱匹配行為，假設軟件包組/npm/*允許獲取和阻止發布。更具體的套件群組會設定為封鎖擷取並允許發佈。/npm//anycompany-spicy-client$名anycompany-spicy-client為的套件與套件群組相符，它允許發佈套件版本並封鎖擷取套件版本。允許發布的軟件包名稱的唯一大小寫是 anycompany-spicy-client，因為它與軟件包定義模式具有很強的匹配。不同的情況變化，例如 AnyCompany-辣-client 被阻止發布，因為它是一個弱匹配。更重要的是，套件群組會封鎖擷取所有大小寫變體，而不僅僅是模式中使用的小寫名稱，進而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，弱匹配還忽略了破折號-，點 ._，下劃線和可混淆字符（例如來自單獨字母的類似字符）序列中的差異。在用於弱匹配的正規化期間， CodeArtifact 執行 case 折疊（類似於轉換為小寫），用單個點替換破折號，點和底線字符的序列，並將可混淆的字符標準化。

弱匹配將破折號，點和下劃線視為等效，但不會完全忽略它們。這意味著 FO O 酒吧，foo.bar，foo.. bar 和 foo_bar 都是弱匹配等價物，但不是。雖然有幾個公用儲存庫實作了防止這些變異類型的步驟，但是公用儲存庫所提供的保護並不會讓套件群組的這項功能變得不必要。例如，公共存儲庫（如 npm 公共註冊表註冊表）只會阻止名為 my-package 的軟件包的新變體，如果我的包已經發布到它。如果 my-package 是一個內部軟件包，並且創建了允許發布和塊獲取的包組/npm//my-package$，那麼您可能不希望將 my-package 發佈到 npm 公共註冊表以防止允許諸如 my. package 之類的變體。

儘管某些包格式（例如 Maven）對這些字符的方式對待這些字符（Maven 將其視.為命名空間層次結構分隔符，但不是-或_），但類似 com.act-on 的內容仍然可能與 com.act.on 混淆。

> **Note**
>
> 請注意，無論何時有多個變體與封裝群組相關聯，管理員都可以針對特定變體建立新的封裝群組，以針對該變體配置不同的行為。

標記套件群組 CodeArtifact

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在中將標記套用至套件群組 CodeArtifact。如需有關資 CodeArtifact 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資訊，請參閱標記資源。

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

標記套件群組 (CLI)

您可以使用 CLI 來管理封裝群組標籤。

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

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

主題

將標記新增至套件群組 (CLI)

您可以在建立封裝群組時將標記新增至封裝群組，或新增至現有的封裝群組。如需有關在建立封裝群組時將標記新增至封裝群組的資訊，請參閱建立套件群組。

若要將標記新增至現有套件群組 AWS CLI，請在終端機或命令列上執行該tag-resource命令，並指定要新增標籤的套件群組的 Amazon 資源名稱 (ARN)，以及要新增之標籤的金鑰和值。如需封裝群組 ARN 的詳細資訊，請參閱Package 群組 ARN。

您可以將多個標記新增至套件群組。*############/npm/scope/ #### ~ ########## key1 ########## value 1####### key 2 ########## value 2#*

```
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，則此命令沒有輸出。

檢視套件群組的標記 (CLI)

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

在終端機或命令列上，使用套件群組的 Amazon 資源名稱 (ARN) 執行list-tags-for-resource命令。如需封裝群組 ARN 的詳細資訊，請參閱Package 群組 ARN。

例如，若要檢視套件群組的標籤鍵和標籤值清單，請以 ARN 值命名為 */npm/scope/ ##*公司 ~ 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"
    }
}
```

編輯封裝群組的標記 (CLI)

請依照下列步驟使 AWS CLI 用編輯封裝群組的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。您也可以從封裝群組中移除標籤，如下一節所示。

在終端機或命令列上，執行指tag-resource令，指定要更新標籤之套件群組的 ARN，並指定標籤鍵和標籤值。如需封裝群組 ARN 的詳細資訊，請參閱Package 群組 ARN。

```
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，則此命令沒有輸出。

從套件群組移除標記 (CLI)

請依照下列步驟使 AWS CLI 用從封裝群組中移除標籤。

> **Note**
>
> 如果您刪除封裝群組，所有標記關聯都會從刪除的封裝群組中移除。在刪除套件群組之前，您不需要移除標籤。

在終端機或命令列上，執行指untag-resource令，指定要移除其中標籤的套件群組的 ARN，以及要移除之標籤的標籤索引鍵。如需封裝群組 ARN 的詳細資訊，請參閱Package 群組 ARN。

###############/npm/scope/ ########### 1 # key2#

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

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

如果成功，則此命令沒有輸出。移除標籤之後，您可以使用`list-tags-for-resource`指令檢視封裝群組上剩餘的標籤。

使用中的網域 CodeArtifact

CodeArtifact 網域可讓您更輕鬆地跨組織管理多個儲存庫。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。即使資產可從多個儲存庫取得，資產也只會儲存在網域中一次。

雖然您可以擁有多個網域，但我們建議您使用單一生產網域，其中包含所有已發佈的成品，以便您的開發團隊可以尋找並共用封裝。您可以使用第二個預先生產網域來測試生產網域組態的變更。

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

主題

網域概觀

當您使用時 CodeArtifact，網域適用於下列項目：

- 重複資料刪除儲存：即使資產在 1 或 1,000 個儲存庫中可用，也只需在網域中儲存一次。這意味著您只需支付一次存儲費用。
- 快速複製：將套件從上游 CodeArtifact 存放庫提取至下游或使用 CopyPackageVersions API 時，只需更新中繼資料記錄。不會複製任何資產。這使得設置新的存儲庫進行測試或測試變得快速。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。
- 在儲存庫和團隊之間輕鬆共用：網域中的所有資產和中繼資料都使用單一 AWS KMS key (KMS 金鑰) 加密。您不需要為每個儲存庫管理金鑰，也不需要為單一金鑰授與多個帳戶存取權。
- 跨多個儲存庫套用原則：網域管理員可以在整個網域中套用原則。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用存放庫的連線，做為套件來源使用。如需詳細資訊，請參閱網域原則。
- 唯一的存儲庫名稱：該域為存儲庫提供了命名空間。存放庫名稱只需要在網域中是唯一的。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。

您無法建立沒有網域的存放庫。當您使用 CreateRepositoryAPI 建立存放庫時，您必須指定網域名稱。您無法將存放庫從一個網域移至另一個網域。

存放庫可以由擁有該網域的相同 AWS 帳戶或不同帳戶所擁有。如果擁有帳戶不同，則必須將存放庫擁有的帳戶授與網域資源的`CreateRepository`權限。您可以使用PutDomainPermissionsPolicy命令將資源策略新增至網域來執行此操作。

雖然組織可以有多個網域，但建議您擁有一個包含所有已發行成品的單一生產網域，以便開發團隊可以在其組織中尋找並共用封裝。第二個生產前網域對於測試生產網域組態的變更很有用。

跨帳戶網域

網域名稱只需要在帳戶中是唯一的，這表示同一地區內可能有多個具有相同名稱的網域。因此，如果您想要存取未經驗證的帳戶所擁有的網域，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。

存取您經過驗證的帳戶所擁有的網域：

在您驗證的帳戶中存取網域時，您只需要指定網域名稱即可。下列範例會列出您帳戶所擁有之 *my_domain ### my_ repo* 儲存庫中的套件。

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

存取您未經驗證的帳戶所擁有的網域：

存取您未經驗證的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*#######-repo* 儲存庫中的套件，這些套件是您未經驗證的帳戶所擁有。請注意加入`--domain-owner`參數。

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

支援的 AWS KMS 金鑰類型 CodeArtifact

CodeArtifact 僅支援對稱的 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。若要瞭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)。您必須負責使用 XKS 金鑰的金鑰作業的可用性、耐久性和延遲，這可能會影響到的可用性、耐久性和延遲。 CodeArtifact搭 CodeArtifact配使用 XKS 鍵的一些效果範例：

- 由於要求套件的每個資產及其所有相依性都會受到解密延遲的影響，因此可以隨著 XKS 作業延遲的增加而大幅增加建置延遲。
- 由於所有資產都經過加密 CodeArtifact，因此遺失 XKS 金鑰材料將導致使用 XKS 金鑰與網域相關聯的所有資產遺失。

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，它不會包含任何儲存庫。如需詳細資訊，請參閱 建立 儲存庫。如需使用管理 CodeArtifact 網域的詳細資訊 CloudFormation，請參閱建立 CodeArtifact 資源 AWS CloudFormation。

主題

- 建立網域 (主控台)
- 建立網域 (AWS CLI)
- AWS KMS 金鑰原則範例

建立網域 (主控台)

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在功能窗格中，選擇 [網域]，然後選擇 [建立網域]。
3. 在名稱中，輸入您網域的名稱。
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 受管理的 KMS 金鑰或您管理的 KMS 金鑰。如需中受支援的 KMS 金鑰類型的詳細資訊 CodeArtifact，請參閱支援的 AWS KMS 金鑰類型 CodeArtifact。

- 如果要使用預設值，請選擇 AWS 受管金鑰 AWS 受管金鑰。
- 如果您要使用您管理的 KMS 金鑰，請選擇「客戶管理金鑰」。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 受管金鑰和客戶管理的金鑰。

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`指令。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 受管理的 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。

如需中受支援的 KMS 金鑰類型的詳細資訊 CodeArtifact，請參閱支援的 AWS KMS 金鑰類型 CodeArtifact。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 受管金鑰和客戶管理金鑰。

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

JSON 格式的資料會顯示在輸出中，其中包含有關新網域的詳細資料。

```
{
    "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`參數中加入其 Amazon 資源名稱 (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，其中包含有關新網域的詳細資料。

```
{
    "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

若要使用標籤建立網域，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
```

AWS KMS 金鑰原則範例

在中建立網域時 CodeArtifact，您可以使用 KMS 金鑰來加密網域中的所有資產。您可以選擇 AWS 受管理的 KMS 金鑰或您管理的客戶管理金鑰。如需 KMS 金鑰的詳細資訊，請參閱開[AWS Key Management Service 發人員指南]。

若要使用客戶受管金鑰，您的 KMS 金鑰必須具有授與存取權的金鑰原則 CodeArtifact。金鑰原則是金 AWS KMS 鑰的資源原則，也是控制 KMS 金鑰存取權的主要方式。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。

下列範例金鑰原則陳述式 AWS CodeArtifact 允許建立授權，並代表授權使用者檢視金鑰詳細資料。此政策聲明限制了使用`kms:ViaService`和`kms:CallerAccount`條件鍵代表指定帳戶 ID CodeArtifact 採取行動的權限。它也會將所有 AWS KMS 權限授與 IAM 根使用者，因此在建立金鑰後即可進行管理。

```
{
```

```
    "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 (AWS CLI) 刪除網域。

主題

- 域名刪除的限制
- 刪除網域（主控台）

- 刪除網域 (AWS CLI)

域名刪除的限制

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。

但是，如果 CodeArtifact 無法再存取網域的 KMS 金鑰，即使該網域仍包含儲存庫，您也可以刪除該網域。如果您刪除網域的 KMS 金鑰或撤銷 CodeArtifact 用於存取金鑰的 KMS 授權，就會發生此情況。在此狀態下，您無法存取網域中的儲存庫或儲存在其中的套件。當無法存取網域的 KMS 金鑰時，也 CodeArtifact 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**
>
> 刪除仍包含儲存庫的網域時， CodeArtifact 會在 15 分鐘內非同步刪除儲存庫。刪除域後，存儲庫仍然會顯示在 CodeArtifact 控制台和`list-repositories`命令的輸出中，直到發生自動存儲庫清理為止。

刪除網域（主控台）

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
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指令刪除網域。

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

JSON 格式的資料會顯示在輸出中，其中包含有關已刪除網域的詳細資料。

```
{
```

```
    "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"
    }
}
```

網域原則

CodeArtifact 支持使用基於資源的權限來控制訪問。以資源為基礎的權限可讓您指定誰可以存取資源，以及他們可以對資源執行哪些動作。預設情況下，只有擁有該網域的 AWS 帳戶可以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 主體存取該文件。

如需詳細資訊，請參閱原則和權限和身分型原則和以資源為基礎的原則。

主題

- 啟用對網域的跨帳戶存取
- 網域原則範例
- 網域原則範例 AWS Organizations
- 設定網域原則
- 閱讀網域政策
- 刪除網域原則

啟用對網域的跨帳戶存取

資源策略是 JSON 格式的文本文件。檔案必須指定主參與者 (actor)、一或多個動作，以及效果 (Allow或Deny)。若要在另一個帳戶所擁有的網域中建立存放庫，主參與者必須獲得網域資源的CreateRepository權限。

例如，下列資源策略授與帳號在網域中建立存放庫的123456789012權限。

```
{
```

```
    "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權限。這也將允許帳戶訪問添加標籤到域和其中的所有存儲庫。

因為只會針對其所附加網域的作業評估原則，因此您不需要指定資源。由於隱含資源，所以Resource可以將設定為*。

若要存取其他帳號所擁有之網域中的封裝，必須將網域資源的GetAuthorizationToken權限授與主體。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，以下資源策略授123456789012予帳戶檢索域中任何存儲庫的身份驗證令牌的權限。

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

> **Note**
>
> 想要從存放庫端點擷取套件的主體，除了網域的ReadFromRepository權限之外，還必須獲得存放庫資源的GetAuthorizationToken權限。同樣地，除了要將套件發行至存放庫端點的主體之外，還必須獲得PublishPackageVersion權限GetAuthorizationToken。如需有關ReadFromRepository和PublishPackageVersion權限的詳細資訊，請參閱儲存庫原則。

網域原則範例

當多個帳戶使用一個網域時，應該授與帳戶一組基本的權限，以允許完整使用網域。下列資源策略列出一組允許完整使用網域的權限。

```
{
    "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屬於單一帳戶，且只需要從該帳戶使用，則不需要建立網域原則。

網域原則範例 AWS Organizations

您可以使用aws:PrincipalOrgID條件金鑰來授與組織中所有帳戶的 CodeArtifact 網域存取權，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。

設定網域原則

您可以使用put-domain-permissions-policy指令將原則附加至網域。

```
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，會在評估權限時忽略網域上的資源原則。如此可確保網域擁有者無法將自己鎖定在網域之外，因此他們無法更新資源原則。

> **Note**
>
> 您無法將使用資源策略更新網域上資源策略的權限授與其他 AWS 帳號，因為呼叫時會忽略資源策略 put-domain-permissions-policy。

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 資源名稱 (ARN)、政策文件的完整內容以及修訂識別碼。可以put-domain-permissions-policy使用--policy-revision選項將修訂識別元傳遞給。這可確保文件的已知修訂版本會被覆寫，而不是由其他作者設定的較新版本。

閱讀網域政策

若要讀取原則文件的現有版本，請使用get-domain-permissions-policy指令。若要格式化輸出以提高可讀性，請使用--output和與 Python json.tool 模組--query policy.document一起使用，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"
            }
        }
    ]
}
```

刪除網域原則

使用delete-domain-permissions-policy命令從網域刪除原則。

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

輸出的格式get-domain-permissions-policy與和delete-domain-permissions-policy指令的格式相同。

在中標記網域 CodeArtifact

標籤是與 AWS 資源關聯的索引鍵/值組。您可以在中將標記套用至您的網域 CodeArtifact。如需有關資 CodeArtifact 源標記、使用案例、標籤索引鍵和值條件約束以及支援的資源類型的資訊，請參閱標記資源。

您可以在建立網域時使用 CLI 指定標籤。您可以使用主控台或 CLI 新增或移除標籤，以及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

- 標記網域
- 標記網域 (主控台)

標記網域

您可以使用 CLI 來管理網域標記。

主題

將標記新增至網域 (CLI)

您可以使用主控台或 AWS CLI 標記網域。

若要在建立網域時將標籤新增至網域，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。

在終端機或命令列上，執行tag-resource命令，指定要新增標籤的網域的 Amazon 資源名稱 (ARN)，以及要新增之標籤的金鑰和值。

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

您可以在網域中新增多個標籤。*################ my_domain ######## key1 ########## value1####### key2 ############ value2#*

```
aws codeartifact tag-resource --resource-arn arn:aws:codeartifact:us-
west-2:123456789012:domain/my_domain --tags key=key1,value=value1 key=key2,value=value2
```

如果成功，則此命令沒有輸出。

檢視網域的標記 (CLI)

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

在終端機或命令列上，使用網域的 Amazon 資源名稱 (ARN) 執行list-tags-for-resource命令。

> **Note**
>
> 若要取得網域的 ARN，請執行下describe-domain列命令：
>
> ```
> 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 值的網域的標籤金鑰和標籤值的清單：

```
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"
    }
}
```

編輯網域的標記 (CLI)

請依照下列步驟 AWS CLI 使用編輯網域的標籤。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。

在終端機或命令列上，執行tag-resource命令，指定要更新標籤的網域的 ARN，並指定標籤鍵和標籤值：

> **Note**
>
> 若要取得網域的 ARN，請執行下describe-domain列命令：
>
> ```
> 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，則此命令沒有輸出。

從網域移除標籤 (CLI)

請依照下列步驟 AWS CLI 使用從網域移除標籤。

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

在終端機或命令列上，執行untag-resource命令，指定要移除標籤的網域的 ARN，以及要移除之標籤的標籤金鑰。

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

```
############### 1 ### 2 #### mydomain ##########
```

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

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

標記網域 (主控台)

您可以使用主控台或 CLI 以標記資源。

主題

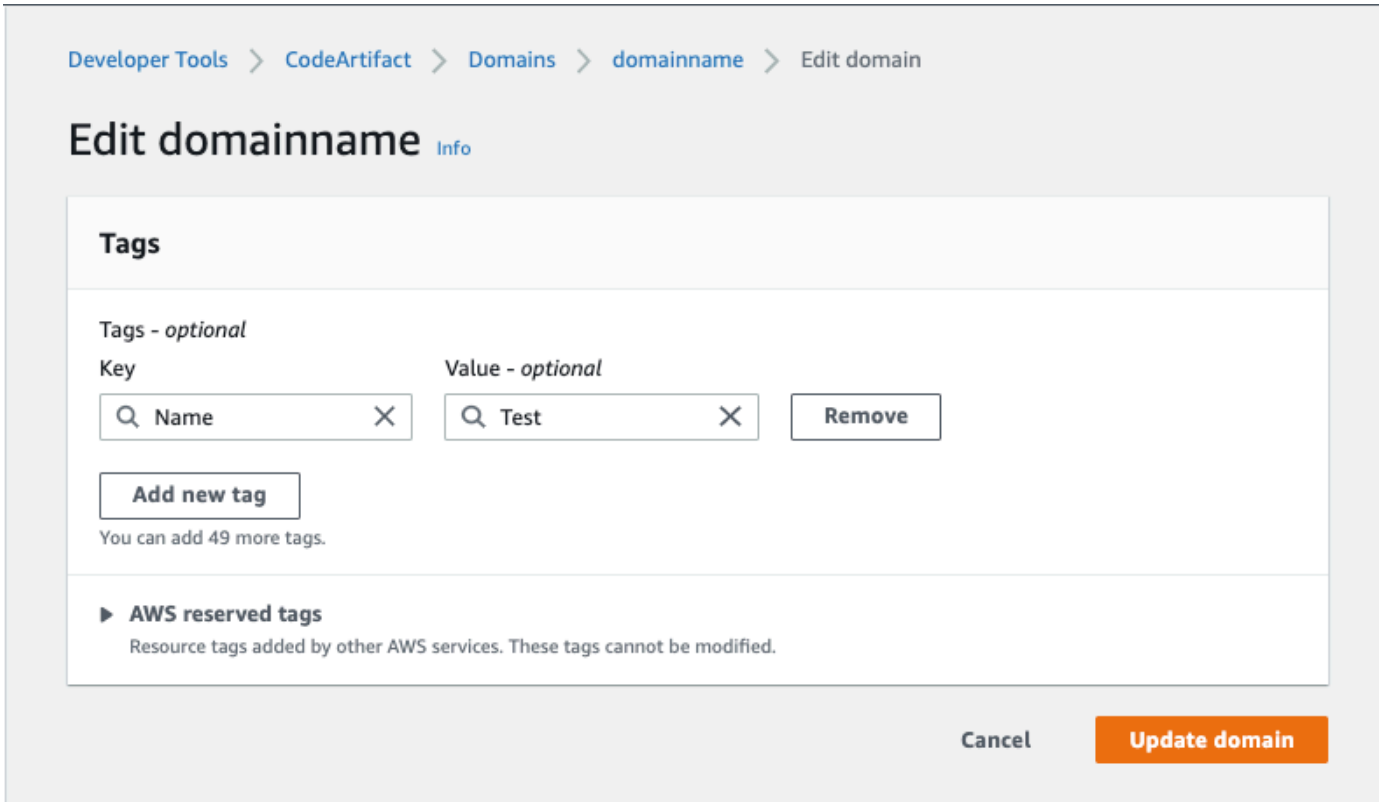
- 將標記新增至網域 (主控台)

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

將標記新增至網域 (主控台)

您可以使用主控台將標籤新增至現有網域。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「網域」頁面上，選擇您要新增標籤的網域。
3. 展開「詳細資料」區段。
4. 在 [網域代碼] 底下，如果網域上沒有標籤，請選擇 [新增網域代碼]，或選擇 [檢視並編輯網域代碼 (如果有的話)]。
5. 選擇 Add new tag (新增標籤)。
6. 在「關鍵字」和「值」欄位中，輸入您要加入的每個標籤的文字。(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]。

檢視網域的標記 (主控台)

您可以使用主控台列出現有網域的標籤。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「網域」頁面上，選擇您要檢視標籤的網域。
3. 展開「詳細資料」區段。
4. 在 [網域代碼] 底下，選擇 [檢視和編輯網域代碼]

> **Note**
>
> 如果沒有新增標籤至此網域，則主控台會顯示「新增網域標記」。

編輯網域的標籤 (主控台)

您可以使用主控台編輯已新增至網域的標籤。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「網域」頁面上，選擇您要更新標籤的網域。
3. 展開「詳細資料」區段。
4. 在 [網域代碼] 底下，選擇 [檢視和編輯網域代碼]

> **Note**
>
> 如果沒有新增標籤至此網域，則主控台會顯示「新增網域標記」。

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. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在「網域」頁面上，選擇您要移除標籤的網域。
3. 展開「詳細資料」區段。
4. 在 [網域代碼] 底下，選擇 [檢視和編輯網域代碼]

> **Note**
>
> 如果沒有新增標籤至此網域，則主控台會顯示「新增網域標記」。

5. 在您要刪除之每個標籤的機碼和值旁邊，選擇「移除」。
6. 選擇 [更新網域]。

使用 npm 的 CodeArtifact

這些主題介紹如何使用 npm (Node.js 套件管理 CodeArtifact 具)。

> **Note**
>
> CodeArtifactnode v4.9.1和更高版本npm v5.0.0和更高版本.

主題

配置和使用 npm CodeArtifact

在中建立儲存庫之後 CodeArtifact，您可以使用 npm 用戶端來安裝和發佈套件。使用存儲庫端點和授權令牌配置 npm 的推薦方法是使用aws codeartifact login命令。您也可以手動配置 npm。

內容

使用登錄命令配置 npm

使用該aws codeartifact login命令獲取憑據以與 npm 一起使用。

> **Note**
>
> 如果您正在訪問您擁有的域中的存儲庫，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 npm 10.x 或更新版本，則必須使用 2.9.5 或更新 AWS CLI 版本才能成功執行命令。`aws codeartifact login`

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

這個命令會對 ~/.npmrc 檔案進行下列變更：

- 從 CodeArtifact 使用 AWS 憑據獲取後添加授權令牌。
- 將 npm 註冊表設置為由`--repository`選項指定的存儲庫。
- 對於 npm 6 及更低版本：添加`"always-auth=true"`以便為每個 npm 命令發送授權令牌。

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權杖。如需有關使用`login`指令建立之授權權杖的詳細資訊，請參閱使用`login`指令建立的權杖。

在不使用登錄命令的情況下配置 npm

您可以通過手動更新 npm 配置來使用 CodeArtifact 存儲庫配置 npm，而無需使用`aws codeartifact login`命令。

在不使用登錄命令的情況下配置 npm

1. 在命令行中，獲取 CodeArtifact 授權令牌並將其存儲在環境變量中。npm 將使用此令牌與您的存儲庫進行身份驗證。 CodeArtifact

> **Note**
>
> 以下指令適用於 macOS 或 Linux 電腦。如需在 Windows 電腦上設定環境變數的資訊，請參閱使用環境變量傳遞身份驗證令牌。

```
CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

2. 通過運行以下命令獲取 CodeArtifact 存儲庫的端點。您的存儲庫端點用於將 npm 指向存儲庫以安裝或發布包。

   - 使用您的######### CodeArtifact 域名稱。
   - 以網域擁有者的 AWS 帳號識別碼取代 *111122223333*。如果您正在訪問您擁有的域中的存儲庫，則不需要包含--domain-owner。如需詳細資訊，請參閱 跨帳戶網域。
   - 用您的 CodeArtifact 存儲庫名稱替換 *my_repo*。

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format npm
```

   以下 URL 是存放庫端點範例。

```
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，您無法連接到存放庫。

3. 使用命npm config set令將註冊表設置為您的 CodeArtifact 存儲庫。將 URL 取代為上一個步驟中的存放庫端點 URL。

```
npm config set
 registry=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/
```

4. 使用該npm config set命令將授權令牌添加到 npm 配置中。

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:_authToken=$CODEARTIFACT_AUTH_TOKEN
```

對於 npm 6 或更低版本：要使 npm 始終將 CodeArtifact auth 令牌傳遞給甚至對於GET請求，請使用npm config set. always-auth

```
npm config set //my_domain-111122223333.d.codeartifact.region.amazonaws.com/
npm/my_repo/:always-auth=true
```

實例 npm 配置文件（**.npmrc**）

以下是遵循上述指示設定 CodeArtifact 登錄端點、新增驗證 Token 並進行設定之後的範例.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
```

運行 npm 命令

設定 npm 用戶端之後，您可以執行 npm 命令。假設軟件包存在於您的存儲庫或其上游存儲庫之一中，則可以使用npm install. 例如，使用下列指令來安裝lodash套件。

```
npm install lodash
```

使用以下命令將新的 npm 包發佈到 CodeArtifact 存儲庫。

```
npm publish
```

如需如何建立 npm 套件的詳細資訊，請參閱 npm 文件網站上的建立 Node.js 模組。如需支援的 npm 命令清單 CodeArtifact，請參閱 npm 命令 Support。

驗證 npm 身份驗證和授權

調用`npm ping`命令是驗證以下內容的一種方法：

- 您已正確設定認證，以便對 CodeArtifact 儲存庫進行驗證。
- 授權配置授予您`ReadFromRepository`權限。

成功叫用的輸出如`npm ping`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
```

該`-d`選項會使 npm 打印其他調試信息，包括存儲庫 URL。此信息可以很容易地確認 npm 配置為使用您期望的存儲庫。

更改回默認的 npm 註冊表

使用配置 npm 將 npm 註冊表 CodeArtifact 設置為指定的 CodeArtifact 存儲庫。連接完成後，您可以運行以下命令將 npm 註冊表設置回其默認註冊表 CodeArtifact。

```
npm config set registry https://registry.npmjs.com/
```

故障排除 npm 8.x 或更高版本的緩慢安裝

npm 8.x 版本及更新版本中存在一個已知問題，如果對套件儲存庫發出請求，且儲存庫會將用戶端重新導向至 Amazon S3，而不是直接串流資產，則 npm 用戶端可能會掛起每個相依性數分鐘。

由於 CodeArtifact 儲存庫的設計一律會將請求重新導向至 Amazon S3，因此有時會發生此問題，導致長時間的 npm 安裝時間造成建置時間。這種行為的實例將自己顯示為顯示幾分鐘的進度條。

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
```

配置和使用紗線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`讀取並使用您的 npm 配置文件（.npmrc）中的信息，而`Yarn 2.X`沒有。的配置`Yarn 2.X`必須在 .yarnrc.yml 檔案中定義。

內容

配置紗線 1.X`aws codeartifact login`命令

對於`Yarn 1.X`，您可以配置紗線CodeArtifact使用`aws codeartifact login`指令。該`login`命令將使用您的〜/.npmrc 文件配置CodeArtifact儲存庫端點資訊和認證。同`Yarn 1.X`,`yarn`指令會使用 ~/.npmrc 檔案中的組態資訊。

若要設定**`Yarn 1.X`**使用登錄命令

1. 如果您尚未這樣做，請配置您的AWS憑證以搭配使用AWS CLI，如中所述入門 CodeArtifact。
2. 若要執行`aws codeartifact login`命令成功，必須安裝 npm。請參閱下載和安裝 Node.js 和故宮在npm 文件以取得安裝說明。
3. 使用`aws codeartifact login`要擷取的指令CodeArtifact憑據並配置您的 ~/.npmrc 文件。
   - 取代*`######`*與您的CodeArtifact網域名稱。
   - 取代*`111122223333`*與AWS網域擁有者的帳號 ID。如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 取代*`####`*與您的CodeArtifact儲存庫名稱。

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

該login指令對您的 ~/.npmrc 檔案進行下列變更：

- 從中獲取後添加授權令牌CodeArtifact使用您的AWS認證。
- 將 npm 註冊表設置為由指定的存儲庫--repository選項。
- 對於 npm 6 及更低版本：添加"always-auth=true"所以授權令牌是為每個 npm 命令發送的。

調用後的默認授權期login是 12 小時，並且login必須調用以定期刷新令牌。如需有關使用login指令，請參閱使用login指令建立的權杖。

4. 對於故宮 7.X 和 8.X，您必須新增always-auth=true到您的〜/.npmrc 文件中以使用紗線。

   - 在文本編輯器中打開〜/.npmrc 文件並添加always-auth=true在一條新行上。

您可以使用yarn config list指令來檢查 Yarn 是否使用正確的組態。執行命令後，請檢查info npm config部分。內容看起來應該類似於下面的代碼片段。

```
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
}
```

配置紗線 2.X 與yarn config set命令

下列程序詳細說明如何設定Yarn 2.X通過更新您的.yarnrc.yml從命令行配置yarn config set指令。

若要更新**yarnrc.yml**從命令行配置

1. 如果您尚未這樣做，請配置您的AWS憑證以搭配使用AWS CLI，如中所述入門 CodeArtifact。

2. 使用`aws codeartifact get-repository-endpoint`命令來獲取您的CodeArtifact儲存庫的端點。

   - 取代*######*與您的CodeArtifact網域名稱。
   - 取代*111122223333*與AWS網域擁有者的帳號 ID。如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 取代*####*與您的CodeArtifact儲存庫名稱。

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format npm
```

3. 更新`npmRegistryServer`使用存儲庫端點的 .yarnrc.yml 文件中的值。

```
yarn config set npmRegistryServer
 "https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"
```

4. 獲取一個CodeArtifact授權令牌並將其存儲在環境變量中。

> **Note**
>
> 以下指令適用於 MacOS 或 Linux 電腦。如需在 Windows 電腦上設定環境變數的相關資訊，請參閱使用環境變量傳遞身份驗證令牌。

   - 取代*######*與您的CodeArtifact網域名稱。
   - 取代*111122223333*與AWS網域擁有者的帳號 ID。如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。
   - 取代*####*與您的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`命令添加您的CodeArtifact身份驗證令牌到您的 .yarnrc.yml 文件。將下列指令中的 URL 取代為步驟 2 中的存放庫端點 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`指令來設定值`npmAlwaysAuth`至`true`。將下列指令中的 URL 取代為步驟 2 中的存放庫端點 URL。

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAlwaysAuth'
 "true"
```

設定完成後，您的 .yarnrc.yml 組態檔案應該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`用於檢查值的命令`npmRegistries`和`npmRegistryServer`。

npm 指令支援

下列各節摘要說明除了不受支援的特定命令之外， CodeArtifact儲存庫所支援的 npm 命令。

內容

與儲存庫互動的支援指令

本節列出 npm 命令，其中 npm 客戶端向已配置的註冊表發出一個或多個請求（例如，使用`npm config set registry`）。這些指令已經過驗證，可在針對 CodeArtifact 儲存庫叫用時正常運作。

Command	描述
臭蟲	嘗試猜測軟件包的錯誤跟踪器 URL 的位置，然後嘗試打開它。
CI	使用乾淨的石板安裝專案。
棄用	棄用套件的版本。
死亡標籤	修改封裝發佈標籤。
文件	嘗試猜測套件文件 URL 的位置，然後嘗試使用`--browser` config 參數開啟它。
醫生	運行一組檢查，以確保您的 npm 安裝具有管理 JavaScript 軟件包所需的內容。
安裝	安裝套件。
install-ci-test	使用乾淨的平板安裝項目並運行測試。別名：`npm cit`。這個命令會立`npm ci`即執行一個`npm test`。
安裝測試	安裝軟件包並運行測試。立即運行`npm install`後跟一個`npm test`.
過時	檢查已設定的登錄，查看是否有任何已安裝的套件目前已過期。
平	ping 配置或給定的 npm 註冊表並驗證身份驗證。
發佈	將套件版本發佈至登錄。

Command	描述
update	猜測套件存放庫 URL 的位置，然後嘗試使用 `--browser` config 參數開啟它。
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

這些命令不需要與存儲庫進行任何直接交互，因此 CodeArtifact 不需要做任何事情來支持它們。

Command	描述
建立	構建一個軟件包。
快取	操作套件快取。
完成	在所有 npm 命令中啟用選項卡完成。
配置	更新使用者和全域`npmrc`檔案的內容。
刪除	搜尋本機套件樹狀結構，並嘗試將相依性往上移動樹狀結構，以便讓多個相依套件更有效地共用相依性，藉此簡化結構。
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封裝資料夾。
探索	瀏覽已安裝的套件。在指定的已安裝套件的目錄中產生一個子 shell。如果指定了一個命令，那麼它在子 shell 中運行，然後立即終止。
help	獲取關於 npm 的幫助。
幫助搜索	搜索 npm 幫助文檔。
初始化	創建一個`package.json` 文件。
鏈接	符號鏈接一個包文件夾。

Command	描述
ls	列出已安裝的套件。
包	從封裝建立壓縮包。
prefix	顯示前置字元。除非另外指定，否則這-g是包含package.json 文件的最接近父目錄。
修剪	移除父套件相依性清單中未列出的套件。
重建	在相符的資料夾上執行npm build命令。
重啟	執行套件的停止、重新啟動和啟動指令碼，以及相關的前置和後置指令碼。
根	將有效的node_modules 資料夾列印為標準輸出。
運行腳本	執行任意套件指令碼。
shrinkwrap	鎖定要發佈的相依性版本。
卸載	解除安裝套件。

不支援的命

CodeArtifact 儲存庫不支援這些 npm 命令。

Command	描述	備註
存取	設定已發佈封裝的存取層級。	CodeArtifact 使用與公共 npmjs 存儲庫不同的權限模型。
添加用戶	新增登錄使用者帳戶	CodeArtifact 使用與公共 npmjs 存儲庫不同的用戶模型。

Command	描述	備註
審計	執行安全性稽核。	CodeArtifact 目前沒有出現安全漏洞數據。
掛鉤	管理 npm 掛鉤，包括添加，刪除，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 使用與公共 npmjs 存儲庫不同的身份驗證模型。如需詳細資訊，請參閱使用 npm 驗證。
登出	登出註冊表。	CodeArtifact 使用與公共 npmjs 存儲庫不同的身份驗證模型。無法從 CodeArtifact 存儲庫註銷，但是身份驗證令牌在其可配置的到期時間後過期。默認令牌持續時間為 12 小時。
所有者	管理套件擁有者。	CodeArtifact 使用與公共 npmjs 存儲庫不同的權限模型。
profile	變更登錄設定檔的設定。	CodeArtifact 使用與公共 npmjs 存儲庫不同的用戶模型。
search	在登錄中搜尋符合搜尋字詞的套件。	CodeArtifact 使用列表包命令支持有限的搜索功能。
明星	標記您最喜歡的軟件包。	CodeArtifact 目前不支持任何類型的收藏夾機制。
明星	檢視標記為我的最愛的套件。	CodeArtifact 目前不支持任何類型的收藏夾機制。

Command	描述	備註
團隊	管理組織團隊和團隊成員。	CodeArtifact 使用與公共 npmjs 存放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識別 (使用者、群組和角色)。
token	管理您的身份驗證令牌。	CodeArtifact 使用不同的模型來獲取身份驗證令牌。如需詳細資訊，請參閱使用 npm 驗證。
取消發佈	從登錄中移除套件。	CodeArtifact 不支持使用 npm 客戶端從存儲庫中刪除軟件包版本。您也可以使用 delete-package-version 命令。
哇美	顯示 npm 使用者名稱。	CodeArtifact 使用與公共 npmjs 存儲庫不同的用戶模型。

npm 標籤處理

npm 註冊表支持標籤，它們是軟件包版本的字符串別名。您可以使用標籤來提供別名而不是版本號。例如，您可能有一個具有多個開發流的項目，並使用不同的標籤（例如`stable`、`beta`、`dev`、`canary`）。如需詳細資訊，請參閱「」DI-標籤在 npm 網站上。

默認情況下，npm 使用`latest`標記來標識軟件包的當前版本。`npm install` *`pkg`*（不含`@`*`version`*或者`@`*`tag`*說明符）安裝最新的標籤。通常情況下，項目僅對穩定版本使用最新標籤。其他標籤用於不穩定版本或售前發佈版本。

使用 npm 客戶端編輯標籤

三個`npm dist-tag`命令 (`add`、`rm`，和`ls`) 在 CodeArtifact 存儲庫中的功能與它們在默認 npm 註冊表。

npm 標籤和複製打包 API

當您使用CopyPackageVersionsAPI 複製 npm 包版本時，所有別名該版本的標籤都將複製到目標存儲庫。當正在複製的版本具有目標中也存在的標記時，複製操作會將目標資料檔案庫中的標記值設置為與源資料檔案庫中的值匹配。

例如，假設存儲庫 S 和存儲庫 D 都包含單個版本的web-helper軟件包的最新標籤，如下表所示。

儲存庫	套件名稱	Package
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被調用來複製web-helper1.0.1 從 S 到 D 操作完成後，latest標籤上web-helper，而不是 1.0.0。

如果複製後需要變更標籤，請使用npm dist-tag命令直接在目標存儲庫中修改標籤。如需的詳細資訊，CopyPackageVersionsAPI，請參在儲存庫之間複製程式。

npm 標籤和上遊存儲庫

當 npm 請求包的標籤並且該軟件包的版本也存在於上遊存儲庫中時，CodeArtifact Fit 會在將標籤返回到客户端之前合併這些標籤。例如，名為 R 的存儲庫具有名為 U 的上遊存儲庫。下表顯示了名為web-helper，這兩個存儲庫都存在。

儲存庫	套件名稱	Package
R	web-helper	最新（1.0.0 版的別名）
U	web-helper	alpha（1.0.1 版的別名）

在這種情況下，當 npm 客户端獲取web-helper軟件包時，它會同時接收最新和alpha標籤。標籤指向的版本不會更改。

當同一個標籤存在於上遊和下遊存儲庫中的同一個包上時，CodeArtifact T 使用上游儲存庫。例如，假設網頁幫助程序已被修改為如下所示。

儲存庫	套件名稱	Package
R	web-helper	最新（1.0.0 版的別名）
U	web-helper	最新（1.0.1 版的別名）

在這種情況下，當 npm 客户端獲取包的標籤網頁協助程式從存儲庫 R 中，最新標籤將對版本進行別名1.0.1因為這就是上遊存儲庫中的內容。這樣可以輕鬆地在上遊存儲庫中使用尚未存在於下遊存儲庫的新軟件包版本，方法是通過運行npm update。

在下遊存儲庫中發佈新版本的軟件包時，在上遊存儲庫中使用標記可能會出現問題。例如，假設包上的最新標籤網頁協助程式在 R 和 U 中都是相同的。

儲存庫	套件名稱	Package
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 版發佈到 R 時，npm 會更新最新標籤設定為 1.0.2。

儲存庫	套件名稱	Package
R	web-helper	最新（1.0.2 版的別名）
U	web-helper	最新（1.0.1 版的別名）

但是，npm 客户端永遠不會看到此標記值，因為最新在 U 為 1.0.1。執行中npm install針對存儲庫 R 發佈 1.0.2 後立即安裝 1.0.1 而不是剛剛發佈的版本。要安裝最近發佈的版本，必須指定確切的軟件包版本，如下所示。

```
npm install web-helper@1.0.2
```

Support 兼容 nPM 的軟件包管理器

這些其他軟件包管理器與 CodeArtifact 兼容，並且可以使用 npm 軟件包格式和 npm 線路協議：

- pnpm 軟件包管理器。最新版本確認可與 CodeArtifact 一起使用，是 3.3.4，已於 2019 年 5 月 18 日發佈。
- 紗線包管理器。最新版本確認可與 CodeArtifact 一起使用，是 1.21.1，已於 2019 年 12 月 11 日發佈。

> **Note**
>
> 我們建議使用 Yarn 2.x 與 CodeArtifact 一起使用。Yarn 1.x 沒有 HTTP 重試，這意味着它更容易受到間歇性服務故障的影響，導致 500 級狀態代碼或錯誤。沒有辦法為 Yarn 1.x 配置不同的重試策略，但這已經添加到 Yarn 2.x 中。您可以使用 Yarn 1.x，但您可能需要在構建腳本中添加更高級別的重試。例如，在循環中運行 yarn 命令，以便在下載軟件包失敗時重試。

使用CodeArtifact與蟒蛇

這些主題說明如何使用pip，Python 軟件包管理器，以及twine，Python 套件發佈公用程式，使用CodeArtifact。

主題

- 配置和使用 pipCodeArtifact
- 配置和使用麻線CodeArtifact
- 套件名稱規範化
- 蟒蛇相容性
- 從上流和外部連接請求 Python 包

配置和使用 pipCodeArtifact

點子是 Python 軟件包的軟件包安裝程序。要使用 pip 從您的CodeArtifact存儲庫中，您必須首先配置 pip 客戶端CodeArtifact儲存庫資訊和認證。

點子只能用於安裝 Python 軟件包。要發布 Python 包，您可以使用纏繞。如需詳細資訊，請參閱配置和使用麻線CodeArtifact。

使用配置點子`login`命令

首先，配置您的AWS憑證以搭配使用AWS CLI，如中所述入門 CodeArtifact。然後，使用CodeArtifact `login`用於擷取認證和設定的命令pip和他們在一起。

> **Note**
>
> 如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱跨帳戶網域。

要配置 pip，請運行以下命令。

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

login從中獲取授權令牌CodeArtifact使用您的AWS認證。該login命令將配置pip與一起使用CodeArtifact透過編輯~/.config/pip/pip.conf以設定index-url至指定的儲存庫--repository選項。

調用後的默認授權期login是 12 個小時，並且login必須調用以定期刷新令牌。如需有關使用login指令，請參閱使用login指令建立的權杖。

無需登錄命令即可配置 pip

如果您無法使用login要設定的指令pip，您可以使用pip config。

1. 使用AWS CLI獲取新的授權令牌。

   > **Note**
   >
   > 如果您正在存取您擁有的網域中的儲存庫，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以設定CodeArtifact登錄網址和認證。下面的命令將只更新當前的環境配置文件。若要更新整個系統的組態檔案，請取代site與global。

   ```
   pip config set site.index-url https://aws:
   $CODEARTIFACT_AUTH_TOKEN@my_domain-
   111122223333.d.codeartifact.region.amazonaws.com/pypi/my_repo/simple/
   ```

> **Important**
>
> 登錄 URL 必須以正斜線 (/) 結尾。否則，您無法連接到存放庫。

點子配置文件示例

以下是一個例子pip.conf設定之後的檔案CodeArtifact登錄網址和認證。

```
[global]
index-url = https://aws:eyJ2ZX...@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/pypi/my_repo/simple/
```

運行點

若要執行pip指令，您必須設定pip與CodeArtifact。如需詳細資訊，請參閱下列文件。

1. 請按照中的步驟進行操作設定使用 AWS CodeArtifact部分來配置AWS帳戶、工具和權限。
2. 配置twine按照中的步驟配置和使用麻線CodeArtifact。

假設軟件包存在於您的存儲庫或其上游存儲庫之一中，則可以使用pip install。例如，使用下列命令來安裝requests包裝。

```
pip install requests
```

使用-i暫時恢復到安裝軟件包的選項https://pypi.org而不是你的CodeArtifact儲存庫。

```
pip install -i https://pypi.org/simple requests
```

配置和使用麻線CodeArtifact

纏繞是 Python 套件的套件發佈公用程式。若要使用麻線將 Python 套件發佈到您的CodeArtifact存儲庫，您必須首先配置麻線CodeArtifact儲存庫資訊和認證。

麻線只能用於發布 Python 包。要安裝 Python 軟件包，您可以使用點子。如需詳細資訊，請參閱配置和使用 pipCodeArtifact。

配置麻線`login`命令

首先，配置您的AWS憑證以搭配使用AWS CLI，如中所述入門 CodeArtifact。然後，使用CodeArtifact login命令獲取憑據並與他們配置麻線。

> **Note**
>
> 如果您正在訪問您擁有的域中的存儲庫，則不需要包含--domain-owner。如需詳細資訊，請參閱跨帳戶網域。

要配置麻線，請運行以下命令。

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

login從中獲取授權令牌CodeArtifact使用您的AWS認證。該login命令配置麻線以與一起使用CodeArtifact透過編輯~/.pypirc以新增指定的儲存庫--repository帶有憑據的選項。

調用後的默認授權期login是 12 個小時，並且login必須調用以定期刷新令牌。如需有關使用login指令，請參閱使用login指令建立的權杖。

配置麻線沒有login命令

如果您無法使用login命令配置麻線，您可以使用~/.pypirc文件或環境變量。若要使用~/.pypirc文件中，將以下條目添加到它。密碼必須是由get-authorization-tokenAPI。

```
[distutils]
index-servers =
 codeartifact
[codeartifact]
repository = https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
pypi/my_repo/
password = auth-token
username = aws
```

若要使用環境變數，請執行下列動作。

> **Note**
>
> 如果您正在存取您擁有的網域中的儲存庫，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`
```

運行麻線

在使用麻線發布 Python 包資產之前，必須先配置CodeArtifact權限和資源。

1. 請按照中的步驟進行操作設定使用 AWS CodeArtifact部分來配置AWS帳戶、工具和權限。
2. 按照以下步驟配置麻線配置麻線login命令或者配置麻線沒有login命令。

配置麻線後，您可以運行twine命令。使用下面的命令來發布 Python 包資產。

```
twine upload --repository codeartifact mypackage-1.0.tgz
```

有關如何構建和打包 Python 應用程序的信息，請參閱產生分發封存在 Python 包裝管理局網站上。

套件名稱規範化

CodeArtifact在存儲它們之前將包名稱標準化，這意味著軟件包名稱CodeArtifact可能與發行套件時提供的名稱不同。

對於 Python 包，在執行規範化時，包名稱是小寫的，所有字符實例.,-，以及_被替換為一個-字元。所以包名pigeon_cli和pigeon.cli被標準化並存儲為pigeon-cli。非標準化名稱可以通過 pip 和麻線使用，但標準化名稱必須用於CodeArtifactCLI 或 API 要求 (例如list-package-versions) 以及在 ARN 中。如需有關 Python 套件名稱規範化的詳細資訊，請參閱佩普 503在 Python 文檔中。

蟒蛇相容性

CodeArtifact不支持 PyPI 的XML-RPC或者JSONAPI。

CodeArtifact支持 PyPI 的Legacy應用程式介面 (除了simpleAPI。雖然CodeArtifact不支援/simple/API 端點，它確實支持/simple/<project>/端點。

有關更多信息，請參閱 Python 打包管理局的以下內容GitHub儲存庫。

- XML-端口 API
- JSON API
- 舊版 API

pip 命令支持

以下各節概述了支持的 pip 命令，CodeArtifact儲存庫，以及不受支援的特定指令。

主題

支援與儲存庫互動的指令

本節列出pip指令，其中pip客戶端向其配置的註冊表發出一個或多個請求。這些命令已被驗證為在調用時正常運行CodeArtifact儲存庫。

命令	描述
安裝	安裝套件。
下載	下載套件。

CodeArtifact沒有實現`pip search`。如果您已設定pip用一個CodeArtifact儲存庫, 執行`pip search`將搜索並顯示軟件包PyPI。

支援的用戶端命

這些命令不需要與存儲庫進行任何直接交互，因此CodeArtifact不需要做任何事情來支持他們。

命令	描述
卸載	解除安裝套件。
凍結	以需求格式輸出已安裝的套件。
名單	列出已安裝的套件。
show	顯示有關已安裝套件的資訊。
檢查	確認安裝的套件具有相容的相依性

命令	描述
配置	管理本機和全域組態。
車輪	根據您的需求構建車輪。
雜湊	計算包存檔的哈希值。
完成	幫助完成命令。
debug	顯示有用於除錯的資訊。
說明	顯示指令的說明。

從上流和外部連接請求 Python 包

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 套件包含少量資產，但有些資產包含 100 多個資產，通常用於支援多種硬體架構和 Python 解譯器。

對於現有的軟件包版本，新資產發佈到 pypi.org 是很常見的。例如，某些專案會在新版 Python 發行時發佈新資產。 CodeArtifact 使用安裝 Python 套件時`pip install`，會更新保留在 CodeArtifact 儲存庫中的套件版本，以反映來自 pypi.org 的最新資產集。

同樣地，如果上游 CodeArtifact 儲存庫中不存在於目前 CodeArtifact 儲存庫中的套件版本可用新資產，則在執行時`pip install`，它們將保留在目前存放庫中。

猛拉包版本

pypi.org 中的某些套件版本會標示為「抽籤」，它會與套件安裝程式（例如 pip）通訊，除非該版本是唯一符合版本說明符的版本（使用或），否則不應安裝該版本。== ===如需詳細資訊，請參閱 PEP_592。

如果中的套件版本原本 CodeArtifact 是從 pypi.org 的外部連線中擷取，則當您從軟體 CodeArtifact 庫安裝套件版本時，請 CodeArtifact 確保套件版本的更新後抽取的中繼資料是從 pypi.org 擷取。

如何知道軟件包版本是否被拉扯

要檢查軟件包版本是否被拉入 CodeArtifact，您可以嘗試使用`pip install` *`packageName`*`===`*`packageVersion`*. 如果套件版本遭到抽取，您將會收到類似下列內容的警告訊息：

```
WARNING: The candidate selected for download or install is a yanked version
```

若要檢查 pypi.org 中是否有套件版本被抽取，您可以瀏覽套件版本的 pypi.org 清單，網址為。`https://pypi.org/project/`*`packageName`*`/`*`packageVersion`*`/`

在私人包裹上設置被抽取狀態

CodeArtifact 不支援為直接發佈至 CodeArtifact 儲存庫的套件設定抽取的中繼資料。

為什麼 CodeArtifact 不獲取軟件包版本的最新抽取元數據或資產？

通常情況下， CodeArtifact 確保當從 CodeArtifact 存儲庫中提取 Python 包版本時，被抽取的元數據 up-to-date 與 pypi.org 上的最新值。此外，軟件包版本中的資產列表也會以 pypi.org 和任何上游存儲庫上的最新集合保持更新。 CodeArtifact 無論您是第一次安裝軟件包版本並將其從 pypi.org CodeArtifact 導入到您的 CodeArtifact 存儲庫中，或者您之前已安裝過該軟件包，都是如此。但是，在某些情況下，軟件包管理器客戶端（例如 pip）不會從 pypi.org 或上游存儲庫中提取最新的被抽取的元數據。相反， CodeArtifact 將返回已存儲在存儲庫中的數據。本節說明可能發生這種情況的三種方式：

上游配置：如果與 pypi.org 的外部連接從存儲庫或使用其上流中刪除 disassociate-external-connection，則將不再從 pypi.org 刷新抽取的元數據。同樣地，如果您移除上游儲存庫，則目前儲存庫中的資產將無法再使用已移除儲存庫的上行串流。如果您使用 CodeArtifact 套件來源控制項來防止提取特定套件的新版本，也是如此 — 設定`upstream=BLOCK`會阻止重新整理被抽取的中繼資料。

P@@ ackage 版本狀態：如果您將軟件包版本的狀態設置為除了`Published`或之外的任何內容`Unlisted`，則不會刷新包版本的拆卸元數據和資產。同樣，如果您正在獲取特定的軟件包版本（例如`torch 2.0.1`），並且上游存儲庫中存在相同的軟件包版本，其狀態為不是`Published`或`Unlisted`，這也將阻止從上游存儲庫到當前存儲庫的抽取元數據和資產傳播。這是因為其他套件版本狀態表示這些版本不會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到 CodeArtifact 儲存庫中，這將防止從其上游儲存庫和 pypi.org 對套件版本進行抽取的中繼資料和資產重新整理。例如，假設您從軟件包版本（例如`torch 2.0.1`，使用 Web 瀏覽器）下載資產`torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`，

然後使用麻線作為`torch 2.0.1`將其發佈到 CodeArtifact 存儲庫。 CodeArtifact 追蹤套件版本是透過直接發佈到您的儲存庫而不是從外部連線到 pypi.org 或上游儲存庫來進入網域。在這種情況下，CodeArtifact 不會使被抽取的元數據與上游存儲庫或 pypi.org 保持同步。如果您發佈`torch 2.0.1`到上游存放庫中，情況也是如此 — 套件版本的存在將會阻止向上游圖形下方延伸的中繼資料和資產到儲存庫的傳播。

CodeArtifact 與 Maven 一起使用

Maven 的倉庫格式是由許多不同的語言，包括 Java，科特林，斯卡拉和 Clojure 使用。它是由許多不同的構建工具，包括 Maven，搖籃，斯卡拉 SBT，阿帕奇常春藤和萊寧根支持。

我們已經測試並確認與以下版本 CodeArtifact 的兼容性：

- 最新的 Maven 版本：3.6.3。
- 最新搖籃版本：6.4.1。5.5.1 也已經過測試。
- 最新版本：1.11.1 也經過測試。

主題

使用CodeArtifact與搖籃

在您擁有之後CodeArtifact環境變量中的 auth 令牌，如中所述使用環境變量傳遞身份驗證令牌，按照這些說明使用 Maven 軟件包，並將新軟件包發佈到CodeArtifact儲存庫。

主題

擷取相依性

從中獲取依賴關係CodeArtifact在搖籃構建中，使用以下過程。

從中獲取依賴關係CodeArtifact在搖籃構建

1. 如果您還沒有，請創建並存儲CodeArtifact按照以下過程在環境變量中的 auth 令牌使用環境變量傳遞身份驗證令牌。
2. 添加一個maven部分到repositories專案中的區段build.gradle文件。

```
maven {
         url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
         credentials {
             username "aws"
             password System.env.CODEARTIFACT_AUTH_TOKEN
         }
}
```

該url在前面的例子中是你的CodeArtifact儲存庫的端點。搖籃使用端點連接到您的存儲庫。在樣本中，my_domain是您的域名，111122223333是域名所有者的 ID，以及my_repo是儲存庫的名稱。您可以使用get-repository-endpoint AWS CLI指令。

例如，使用名為的存放庫####在名為的域內####，指令如下所示：

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format maven
```

該get-repository-endpoint命令將返回存儲庫端點：

```
url 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
```

該credentials上述範例中的物件包括CodeArtifact您在步驟 1 中創建的身份驗證令牌，Gradle用於進行身份驗證CodeArtifact。

3. (選擇性)-若要使用CodeArtifact存儲庫作為項目依賴項的唯一源代碼，刪除中的任何其他部分repositories從build.gradle。如果您有多個存儲庫，Gradle 會按照列出的順序搜索每個存儲庫中的依賴關係。
4. 設定儲存庫之後，您可以將專案相依性新增至dependencies部分與標準搖籃語法。

```
dependencies {
    implementation 'com.google.guava:guava:27.1-jre'
    implementation 'commons-cli:commons-cli:1.4'
    testImplementation 'org.testng:testng:6.14.3'
}
```

獲取插件

默認情況下搖籃將解決公眾的插件搖籃插件門戶。若要從CodeArtifact儲存庫中，請使用下列程序。

若要從CodeArtifact儲存庫

1. 如果您還沒有，請創建並存儲CodeArtifact按照以下過程在環境變量中的 auth 令牌使用環境變量傳遞身份驗證令牌。
2. 添加一個pluginManagement阻止到您的settings.gradle文件。該pluginManagement區塊必須出現在任何其他陳述式之前settings.gradle，請參閱以下代碼片段：

```
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 解析指定存儲庫中的插件。存儲庫必須具有與 Gradle 插件門戶網站的外部連接的上游存儲庫（例如gradle-plugins-store），以便構建常見需要的 Gradle 插件可用。如需詳細資訊，請參閱搖籃文檔。

發佈成品

本節介紹如何將使用搖籃構建的 Java 庫發布到CodeArtifact儲存庫。

首先，添加maven-publish外掛程式plugins該項目的部分build.gradle文件。

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

接下來，添加一個publishing部分到項目build.gradle文件。

```
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
            }
        }
    }
}
```

該maven-publish插件生成一個基於 POM 文件groupId,artifactId，以及version在中指定publishing部分。

在這些更改之後build.gradle完成後，運行以下命令來構建項目並將其上傳到存儲庫。

```
./gradlew publish
```

使用list-package-versions以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"
        }
    ]
}
```

有關更多信息，請參閱 Gradle 網站上的以下主題：

- 建立爪哇圖書館
- 將專案發佈為模組

在智能 J 理念中運行搖籃構建

您可以在 IntelliJ IDEA 中運行一個搖籃構建，從中提取依賴關係CodeArtifact。若要使用驗證 CodeArtifact，您必須為搖籃提供CodeArtifact授權令牌。有三種方法可以提供身份驗證令牌。

- 方法 1：將身份驗證令牌存儲在`gradle.properties`。如果您能夠覆寫或新增內容，請使用此方法`gradle.properties`文件。
- 方法 2：將身份驗證令牌存儲在單獨的文件中。如果您不想修改您的，請使用此方法`gradle.properties`文件。
- 方法 3：通過運行為每次運行生成新的身份驗證令牌`aws`作為內聯腳本`build.gradle`。如果您希望 Gradle 腳本在每次運行時獲取新令牌，請使用此方法。令牌不會存儲在文件系統上。

Token stored in gradle.properties

方法 1：將身份驗證令牌存儲在**gradle.properties**

> **Note**
>
> 此範例顯示gradle.properties檔案位於GRADLE_USER_HOME。

1. 更新您的build.gradle具有以下代碼片段的文件：

```
repositories {
    maven {
            url
 'https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/my_repo/'
            credentials {
                username "aws"
                password "$codeartifactToken"
            }
    }
}
```

2. 若要擷取外掛程式CodeArtifact，添加一個pluginManagement阻止到您的settings.gradle文件。該pluginManagement區塊必須出現在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. 獲取一個CodeArtifact身份驗證令牌：

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. 將身份驗證令牌寫入gradle.properties檔案:

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > ~/.gradle/gradle.properties
```

Token stored in separate file

方法 2：將身份驗證令牌存儲在單獨的文件中

1. 更新您的build.gradle具有以下代碼片段的文件：

```
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. 若要擷取外掛程式CodeArtifact，添加一個pluginManagement阻止到您的settings.gradle文件。該pluginManagement區塊必須出現在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. 獲取一個CodeArtifact身份驗證令牌：

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text --profile profile-name`
```

4. 將身份驗證令牌寫入在您的文件中指定的文件中build.gradle檔案:

```
echo "codeartifactToken=$CODEARTIFACT_AUTH_TOKEN" > file
```

Token generated for each run in build.gradle

方法 3：通過運行為每次運行生成新的身份驗證令牌**aws**作為內聯腳本**build.gradle**

1. 更新您的build.gradle具有以下代碼片段的文件：

```
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. 若要擷取外掛程式CodeArtifact，添加一個pluginManagement阻止到您的settings.gradle文件。該pluginManagement區塊必須出現在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
            }
        }
    }
}
```

CodeArtifact 與 mvn 一起使用

您可以使用該mvn命令來執行 Maven 構建。本節說明如何設定mvn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 Maven 依賴項下載到一個 CodeArtifact 存儲庫
- 阿帕奇 Maven 項目信息

擷取相依性

mvn要配置從 CodeArtifact 存儲庫獲取依賴關係，您必須編輯 Maven 配置文件settings.xml，並可選擇編輯項目的 POM。

1. 如果您還沒有，請按照設置存儲庫的 CodeArtifact 身份驗證中所述在環境變量中創建身份驗證令牌並使用環境變量傳遞身份驗證令牌將其 CodeArtifact 存儲在環境變量中。
2. 在settings.xml（通常位於~/.m2/settings.xml）中，添加一個帶有CODEARTIFACT_AUTH_TOKEN環境變量引用的<servers>部分，以便 Maven 在 HTTP 請求中傳遞令牌。

   ```
   <settings>
   ...
       <servers>
           <server>
               <id>codeartifact</id>
               <username>aws</username>
               <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
           </server>
       </servers>
   ...
   </settings>
   ```

3. 在元素中新增 CodeArtifact 儲存庫的 URL 端<repository>點。您可以在項目的 POM 文件中settings.xml執行此操作。

   您可以使用get-repository-endpointAWS CLI命令擷取存放庫的端點。

   例如，如果在名為 *my_domain ######### my_ repo* 的儲存庫，指令如下所示：

   ```
   aws codeartifact get-repository-endpoint --domain my_domain --repository my_repo --
   format maven
   ```

   該get-repository-endpoint命令將返回存儲庫端點：

   ```
   url 'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   maven/my_repo/'
   ```

   將存放庫端點新增到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，以僅用 CodeArtifact 於該專案。

```
<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，但在<server>和元<repository>素中必須相同。這可讓指定的認證包含在要求中 CodeArtifact。

進行這些配置更改後，您可以構建項目。

```
mvn compile
```

Maven 記錄它下載到控制台的所有依賴關係的完整 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

要發布一個 Maven 工件`mvn`到一個 CodeArtifact 存儲庫，你還必須編輯`~/.m2/settings.xml`和項目 POM。

1. 如果您還沒有，請按照設置存儲庫的 CodeArtifact 身份驗證中所述在環境變量中創建身份驗證令牌並使用環境變量傳遞身份驗證令牌將其 CodeArtifact 存儲在環境變量中。
2. 添加一個`<servers>`部分，並引`settings.xml`用`CODEARTIFACT_AUTH_TOKEN`環境變量，以便 Maven 在 HTTP 請求中傳遞令牌。

   ```
   <settings>
   ...
       <servers>
           <server>
               <id>codeartifact</id>
               <username>aws</username>
   ```

```
            <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

3. 添加一個<distributionManagement>部分到您的項目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到指定的存放庫。

```
mvn deploy
```

用list-package-versions於檢查封裝是否已成功發佈。

```
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 CodeArtifact 用`mvn deploy:deploy-file`. 這對於想要發佈成品且只有 JAR 檔案且無法存取封裝原始程式碼或 POM 檔案的使用者而言，這會很有幫助。

該`mvn deploy:deploy-file`命令將根據在命令行中傳遞的信息生成 POM 文件。

發佈第三方 Maven 成品

1. 如果您還沒有，請按照設置存儲庫的 CodeArtifact 身份驗證中所述在環境變量中創建身份驗證令牌並使用環境變量傳遞身份驗證令牌將其 CodeArtifact 存儲在環境變量中。
2. 建立包含下列內容的`~/.m2/settings.xml`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**
>
> 上面的例子發布commons-cli 1.4。修改 groupId、文件 ID、版本和檔案引數以發行不同的 JAR。

這些說明基於從 Apache Maven 文檔中將第三方 JAR 部署到遠程存儲庫的指南中的示例。

限制 Maven 依賴項下載到一個 CodeArtifact 存儲庫

如果一個包不能從配置的存儲庫中獲取，默認情況下，該mvn命令從 Maven 中央獲取它。添加元mirrors素以settings.xml使mvn始終使用您的 CodeArtifact 存儲庫。

```
<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>
```

如果您加入mirrors元素，您的settings.xml或中也必須有pluginRepository元素pom.xml。下面的例子從 CodeArtifact 存儲庫獲取應用程序依賴關係和 Maven 插件。

```
<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>
```

下面的例子從 CodeArtifact 存儲庫獲取應用程序的依賴關係，並從 Maven 中央獲取 Maven 插件。

```
<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>
```

阿帕奇 Maven 項目信息

如需 Maven 的詳細資訊，請參閱 Apache Maven 專案網站上的下列主題：

- 設置多個存儲庫
- 設定參考
- 配送管理

- 設定檔

CodeArtifact 與部門一起使用

您可以使deps.edn用clj來管理 Clojure 專案的相依性。本節說明如何設定deps.edn使用 CodeArtifact 儲存庫。

主題

- 擷取相依性
- 發 Artifacts

擷取相依性

Clojure要配置從 CodeArtifact 存儲庫獲取依賴關係，您必須編輯 Maven 配置文件，settings.xml.

1. 在中settings.xml，新增一個<servers>區段，其中包含CODEARTIFACT_AUTH_TOKEN環境變數的參考，以便 Clojure 在 HTTP 要求中傳遞權杖。

   > **Note**
   >
   > 如果您希望 settings.xml 文件位於~/.m2/settings.xml. 如果在其他位置，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，請使用clj -Spom.

3. 在您的deps.edn配置文件中，添加一個與 Maven 服務器 ID 匹配的存儲庫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保證將首先檢查 Mavenclojars 庫的central和存儲庫。之後，deps.edn將檢查中列出的其他存儲庫。
> - 為了防止直接從 Clojar 和 Maven 中央下載，central並且clojars需要設置為nil。

確保您在環境變量中具有 CodeArtifact Auth 令牌（請參閱使用環境變量傳遞身份驗證令牌）。在這些變更之後建立套件時，中的相依性deps.edn將會從中擷取 CodeArtifact。

發 Artifacts

1. 更新您的 Maven 設置並deps.edn包含 CodeArtifact 為 Maven 識別的服務器（請參閱擷取相依性）。您可以使用諸如部署之類的工具將成品上傳至 CodeArtifact。
2. 在您的中build.clj，新增deploy任務，將必要的成品上傳至先前的安裝程式codeartifact存放庫。

```
(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. 執行下列命令來發佈成品：clj -T:build deploy

如需有關修改預設存放庫的詳細資訊，請參閱在 Clojure Deps 和 CLI 參考基本原理中修改預設存放庫。

使用 curl 進行發佈

本節說明如何使用 HTTP 客戶端`curl`將 Maven 工件發佈到 CodeArtifact 儲存庫。使用發佈成品`curl`在您的環境中沒有或想要安裝 Maven 客戶端時非常有用。

發佈一個 Maven 工件**`curl`**

1. FETCH CodeArtifact 授權令牌，請按照使用環境變量傳遞身份驗證令牌並返回到這些步驟。
2. 使用以下內容`curl`命令將 JAR 發佈到 CodeArtifact ：儲存庫

   在每個`curl`命令，取代下列預留位置：

   - Replace*###*與您的 CodeArtifact 網域名稱。
   - Replace*111122223333*使用您的擁有者的 ID CodeArtifact 網域。
   - Replace*us-west-2*與您的 CodeArtifact 網域。
   - Replace*####*與您的 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，並使用@character. 將值放在引號中時，@必須包含在引號內。

3. 使用以下內容`curl`命令將 POM 發佈到 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 神器將在您的 CodeArtifact 存儲庫的狀態為Unfinished。為了能夠使用該軟件包，它必須位於Published狀態。您可將軟件包從Unfinished至Published通過上傳maven-metadata.xml文件添加到您的軟件包中，或者調用更新包裝狀態 API以更改狀態。

   a. 選項 1：使用以下內容curl命令添加maven-metadata.xml文件添加到您的包中：

```
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
```

   下列為maven-metadata.xml文件:

```
<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：將包裹狀態更新為Published使用UpdatePackageVersionsStatusAPI。

```
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
```

如果您只有工件的 JAR 文件，則可以將消耗軟件包版本發佈到 CodeArtifact 儲存庫使用mvn。如果您無權訪問工件的源代碼或 POM，這將非常有用。如需詳細資訊，請參閱 發佈第三方成品。

使用 Maven aven aven aven aven aven aven

當 Maven 工件發布到AWS CodeArtifact 存儲庫時，與包中的每個資產或文件關聯的校驗和用於驗證上傳。資產的例子是罐子，POM 和戰爭文件。對於每個資產，Maven 工件包含多個校驗和文件，這些文件使用資產名稱和附加擴展名，例如md5或sha1。例如，名為的檔案的總和檢查碼檔案my-maven-package.jar可能是my-maven-package.jar.md5和my-maven-package.jar.sha1。

> **Note**
>
> Maven 使用的術語artifact。在本指南中，Maven 軟件包與 Maven 工件相同。如需詳細資訊，請參閱AWS CodeArtifact封裝置。

總和檢查碼儲存

CodeArtifact 不會將 Maven 校驗和存儲為資產。這表示總和檢查碼不會顯示為 ListPackageVersionAssets API 輸出中的個別資產。相反地，所有受支援的總和檢查碼類型中的每個資產 CodeArtifact都可以使用計算的總和檢查碼。例如，調 ListPackageVersionAssets 用 Maven 包版本的響應的一部分commons-lang:commons-lang 2.1是：

```
{
    "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
    }
}
```

即使校驗和不存儲為資產，Maven 客戶端仍然可以在預期的位置發布和下載校驗和。例如，如果`commons-lang:commons-lang 2.1`位於名為的存放庫中`maven-repo`，則 JAR 檔案 SHA-256 總和檢查碼的 URL 路徑為：

```
/maven/maven-repo/commons-lang/commons-lang/2.1/commons-lang-2.1.jar.sha256
```

如果您要將現有的 Maven 套件 (例如先前存放在 Amazon S3 的套件) 上傳到 CodeArtifact 使用一般 HTTP 用戶端`curl`，則不需要上傳總和檢查碼。 CodeArtifact 將自動生成它們。如果您要驗證資產是否已正確上傳，可以使用 ListPackageVersionAssets API 作業將回應中的總和檢查碼與每個資產的原始總和檢查碼值進行比較。

發佈期間的總和檢查碼不符

除了資產和校驗和，Maven 工件還包含一個`maven-metadata.xml`文件。Maven 包的正常發布順序是首先上傳的所有資產和校驗和，然後是`maven-metadata.xml`。例如，先前`commons-lang 2.1`描述的 Maven 套件版本的發佈順序 (假設用戶端設定為發行 SHA-256 總和檢查碼檔案) 將是：

```
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 (例如 JAR 檔案) 的總和檢查碼檔案時，如果上傳的總和檢查碼值與計算的總和檢查碼值之間有不相符，則上傳要求將會失敗，並顯示 400 (錯誤請求) 回應 CodeArtifact。如果對應的資產不存在，請求將失敗並顯示 404（未找到）響應。若要避免此錯誤，您必須先上傳資產，然後上傳總和檢查碼。

上傳`maven-metadata.xml`時， CodeArtifact 通常會將 Maven 包版本的狀態從更改`Unfinished`為`Published`。如果偵測到任何資產的總和檢查碼不符， CodeArtifact 將傳回 400 (錯誤請求) 以回應`maven-metadata.xml`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`檔案尚未上載，則無法下載已上傳之套件版本的任何資產。這是因為套件版本的狀態並未設定為`Published`且會保留`Unfinished`。

CodeArtifact 允許將更多資產添加到 Maven 軟件包版本，即使`maven-metadata.xml`已上傳並且包版本狀態已設置為`Published`。在此狀態下，上傳不符合的總和檢查碼檔案的要求也會失敗，並出現 400 (錯誤要求) 回應。但是，由於套件版本狀態已設定為`Published`，因此您可以從套件下載任何資產，包括檢查碼檔案上傳失敗的資產。下載總和檢查碼檔案上傳失敗之資產的總和檢查碼時，用戶端收到的總和檢查碼值將是 CodeArtifact 根據上傳的資產資料計算的總和檢查碼值。

CodeArtifact 總和檢查碼比較區分大小寫，計算的總和檢查碼 CodeArtifact 會以小寫格式化。因此，如果校驗和`909FA780F76DA393E992A3D2D495F468`上傳，它將失敗並出現校驗和不匹配，因為它 CodeArtifact 不會將其視為等於`909fa780f76da393e992a3d2d495f468`。

從校驗和不匹配中恢復

如果總和檢查碼上傳因為總和檢查碼不符而失敗，請嘗試下列其中一種方法來復原：

- 運行再次發布 Maven 工件的命令。如果網絡問題損壞了校驗和文件，這可能會起作用。如果這樣可以解決網路問題，就會比對總和檢查碼，且下載成功。
- 刪除套件版本，然後重新發佈。如需詳細資訊，請DeletePackageVersions參閱 AWS CodeArtifact API 參考中的。

使用 Maven 快照

Maven 快照是指最新生產分支代碼的 Maven 包的特殊版本。它是最終發布版本之前的開發版本。您可以通過附加到包版本的後綴`SNAPSHOT`來識別 Maven 軟件包的快照版本。例如，版本的快照`1.1`為`1.1-SNAPSHOT`。如需詳細資訊，請參閱什麼是快照版本？ 在阿帕奇 Maven 項目網站上。

AWSCodeArtifact支持發布和消費 Maven 快照。使用以時間為基礎的版本號碼的唯一快照是唯一受支援的快照。CodeArtifact不支持由 Maven 2 客戶端生成的非唯一快照。您可以將支援的 Maven 快照發佈到任何CodeArtifact儲存庫。

主題

快照發佈於CodeArtifact

AWSCodeArtifact支援用戶端 (例如，在發佈快照時使用的要求模式)。mvn因此，您可以按照構建工具或包管理器的文檔進行操作，而無需詳細了解 Maven 快照的發布方式。如果您正在執行更複雜的操作，本節將詳細描述如何CodeArtifact處理快照。

當 Maven 快照發布到CodeArtifact存儲庫時，其先前的版本將保留在稱為構建的新版本中。每次發布 Maven 快照時，都會創建一個新的構建版本。快照的所有先前版本都會在其組建版本中進行維護。發行 Maven 快照集時，其封裝版本狀態會設定為，Published且包含先前版本的組建狀態會設定為Unlisted。這種行為僅適用於包版本-SNAPSHOT作為後綴的 Maven 軟件包版本。

例如，com.mycompany.myapp:pkg-1所謂的 maven 包的快照版本上傳到名為的CodeArtifact存儲庫my-maven-repo。快照版本為1.0-SNAPSHOT。到目前為止，尚未發布任何版本。com.mycompany.myapp:pkg-1首先，初始組建的資產會以下列路徑發佈：

```
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是由發佈快照組建的用戶端所產生。

上傳 .pom 和 .jar 檔案之後，儲存庫中唯com.mycompany.myapp:pkg-1一存在的版本是1.0-20210728.194552-1。即使在前面路徑中指定的版本是，也會發生這種情況1.0-SNAPSHOT。此時的套件版本狀態為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"
}
```

接下來，用戶端會上傳套maven-metadata.xml件版本的檔案：

```
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封裝版本並將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。雖然存放庫com.mycompany.myapp:pkg-1中有兩個版本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"
]
```

執行與之前所示的相同`list-package-version-assets`指令，`--package-version`參數變更為會`1.0-20210728.194552-1`產生相同的輸出。

當其他組建新增至存放庫時，會為每個新組建建建立新的`1.0-SNAPSHOTUnlisted`套件版本。版本`1.0-SNAPSHOT`的資產每次都會更新，以便版本始終引用該版本的最新版本。透過上載新組建的`maven-metadata.xml`檔案來啟動`1.0-SNAPSHOT`使用最新資產更新。

使用快照版本

如果您要求快照，則會傳回具有狀態`Published`的版本。這一定是 Maven 的最新版本。您也可以使用 URL 路徑中的組建版本號碼 (例如`1.0-20210728.194552-1`) 來要求快照集的特定組建 (例如，`1.0-SNAPSHOT`)。要查看 Maven 快照的構建版本，請使用 ListPackageVersionsAPI 指南中的 CodeArtifact API 並將狀態參數設置為`Unlisted`。

刪除快照版本

要刪除 Maven 快照的所有構建版本，請使用 DeletePackageVersionsAPI，指定要刪除的版本。

使用 curl 進行快照發佈

如果您有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，您可能需要將它們重新發佈到AWSCodeArtifact。由於 Maven 快照的CodeArtifact支持方式（請參閱快照發佈於CodeArtifact），使用通用 HTTP 客戶端（例如`curl`）發布快照比發布 Maven 發布版本更複雜，如中所述使用 curl 進行發佈。請注意，如果您使用 Maven 客戶端（例如`mvn`或）構建和部署快照版本，則本節不相關`gradle`。您需要遵循該客戶端的文檔。

發佈快照版本涉及發佈快照版本的一或多個組建。在中CodeArtifact，如果有 n 個快照版本的組建，則會有 n 個以上 1 個CodeArtifact版本：n 個版本全部狀態為`Unlisted`，而一個狀態為的快照版本 (最新發佈的組建)`Published`。快照版本 (也就是版本字串包含「-SNAPSHOT」的版本) 包含與最新發佈組建完全相同的資產集。使用建立此結構的最簡單的最簡單的方`curl`式，如下所示：

1. 使用發佈所有組建的所有資產`curl`。

2. 使用發佈上次組建的maven-metadata.xml檔案 (也就是具有最新日期時間戳記的組建)curl。這將在版本字符串中創建一個帶有「-SNAPSHOT」的版本，並使用正確的資產集。
3. 使用 UpdatePackageVersionsStatusAPI 將所有非最新組建版本的狀態設定為Unlisted。

使用下列curl命令發佈套件快照版本1.0-SNAPSHOT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
```

使用這些示例時：

- 使用您的###稱取代我的CodeArtifact網域名稱。
- 請使用您CodeArtifact網域擁有者的身分AWS 帳戶識別碼取代 *111122223333*。
- 將 *us-west-2* 替換為您AWS 區域的CodeArtifact域所在的位置。
- 使用您的CodeArtifact儲存庫名稱取代#的儲存庫。

> **Important**
>
> 您必須在--data-binary參數值前面加上@字元。將值放在引號中時，@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 JAR 和. 之外，可能還有 JAR 檔案和來源 JAR 檔案pom.xml。您不需要針對套件版本資產發佈總和檢查碼檔案，因為會CodeArtifact自動為每個上傳的資產產生總和檢查碼。若要驗證資產是否正確上傳，請使用list-package-version-assets指令

擷取產生的總和檢查碼，並將這些資產與原始總和檢查碼進行比較。如需有關CodeArtifact處理 Maven 檢查總和的詳細資訊，請參閱 使用 Maven aven aven aven aven aven aven aven。

使用以下 curl 命令發布最新構建版本的maven-metadata.xml文件：

```
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中的一個資產。此外，該<timestamp>值必須存在，並且必須與資產文件名中的時間戳匹配。例如，對於先前發佈的20210729.171330-2組建，的內容maven-metadata.xml將是：

```
<?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>
```

發行之後maven-metadata.xml，最後一個步驟是將所有其他組建版本 (也就是，除了最新組建之外的所有組建版本) 設定為的套件版本狀態為Unlisted。例如，如果該1.0-SNAPSHOT版本有兩個版本（第一個構建為）20210728.194552-1，則將該構建設置為的命令Unlisted是：

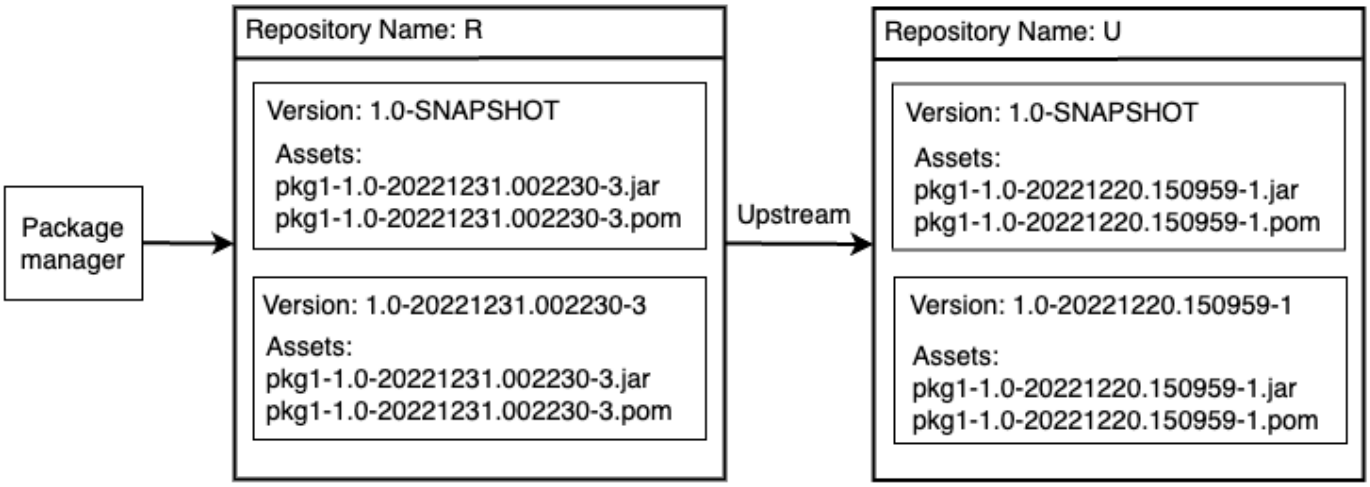
```
aws codeartifact update-package-versions-status --domain my-domain --domain-owner
 111122223333 \
   --repository my-maven-repo --format maven --namespace com.mycompany.app --package
 pkg-1 \
   --versions 1.0-20210728.194552-1 --target-status Unlisted
```

快照和外部連接

Maven 快照不能通過外部連接從 Maven 公共存儲庫中獲取。 AWSCodeArtifact僅支持導入 Maven 發布版本。

快照和上游儲存庫

通常，與上游存儲庫一起使用時，Maven 快照的工作方式與 Maven 發布版本相同。例如，假設一個AWSCodeArtifact域中有兩個存儲庫U，R而且，其中U是R. 在此情況下，您可以自由地將指定套件的快照組建 (例如1.0-SNAPSHOT的com.mycompany.app:pkg-1) 發佈給R和U。不過，當從R (下游存放庫) 使用快照建置時，需要瞭解一些重要的行為。



1. 如果1.0-SNAPSHOT存在於中R，則只R能使用配置為從1.0-SNAPSHOT中獲取軟件包的軟件包管理器來檢索 in 的資產R。您無法擷取到1.0-SNAPSHOT中U的資產R。這是因為中的快照版本U已被中的版本遮蔽R。此行為與 Maven 發行版本和其他套件格式的行為相同。在圖中，GET的/maven/R/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/pkg-1-1.0-20221231.002230-3.jar將返回一個 200（正常）HTTP 響應代碼，但是GET的/maven/R/com/mycompany/myapp/

pkg-1/1.0-SNAPSHOT/pkg-1-1.0-20221220.150959-1.jar將返回 404（未找到）HTTP 響應代碼。

2. 如果1.0-SNAPSHOT存在於中U但不存在R，您可以1.0-SNAPSHOT從中提取資產R。這將導1.0-SNAPSHOT致保留在中R，與發行版本相同。
3. 保留1.0-SNAPSHOT在中之後R，您可以發佈1.0-SNAPSHOT中的其他組建U。但是，R由於第（1）點描述的行為，這些將無法從中訪問。這表示使用快照版本的標準基本原理，也就是透過特定快照版本使用相依性的最新組建，在上游關係中無法如預期般運作。即使新的組建1.0-SNAPSHOT已發佈至U，取用者仍無法存取的1.0-SNAPSHOT最新組建R。若要解決此問題，請定期刪除1.0-SNAPSHOT中的版本，R或將用戶端設定為1.0-SNAPSHOT從中提取版本U。
4. Unlisted快照組建版本可從下游存放庫存取。在圖表中，aGET 的/maven/R/com/mycompany/myapp/pkg-1/1.0-20221220.150959-1/pkg-1-1.0-20221220.150959-1.jar將傳回 200 (OK) 回應碼。即使這要求上游存放庫中存在的資產，因為版本是使用構建版本字符串（1.0-20221220.150959-1）來解決的，因此可以通過下游存儲庫獲取該資產。這也GET會導致版本1.0-20221220.150959-1保留在中R，套件版本狀態為Unlisted。

從上流和外部連接請求 Maven 軟件包

匯入標準資產名稱

從公用儲存庫 (例如 Maven Central) 匯入 Maven 套件版本時，AWS 會 CodeArtifact 嘗試匯入該套件版本中的所有資產。如中所述請求具有上游存儲庫的軟件包版本，匯入發生在以下情況：

- 客戶端請求從 CodeArtifact 存儲庫中的 Maven 資產。
- 套件版本尚未存在於儲存庫或其上行串流中。
- 有一個可訪問的外部連接到公共 Maven 存儲庫。

即使客戶端可能只請求了一個資產，但會 CodeArtifact 嘗試導入它可以為該軟件包版本找到的所有資產。如何 CodeArtifact 發現哪些資產可用於 Maven 包版本取決於特定的公共存儲庫。一些公共 Maven 存儲庫支持請求資產列表，但有些則不支持。對於不提供列出資產的方式的存放庫， CodeArtifact 會產生一組可能存在的資產名稱。例如，當請求 Maven 包版本junit 4.13.2的任何資產時，CodeArtifact 將嘗試導入以下資產：

- 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 客戶端請求與上述模式之一不匹配的資產時，請 CodeArtifact 檢查該資產是否存在於公共存儲庫中。如果資產存在，它將被導入並添加到現有的軟件包版本記錄（如果存在）。例如，Maven 軟件包版本com.android.tools.build:aapt2 7.3.1-8691043包含以下資產：

- 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

當客戶端請求 POM 文件時，如果 CodeArtifact 無法列出軟件包版本的資產，POM 將是唯一導入的資產。這是因為沒有其他資產符合標準資產名稱樣式。不過，當用戶端要求其中一個 JAR 資產時，該資產將會匯入並新增至儲存在中的現有套件版本 CodeArtifact。最下游存放庫 (用戶端發出請求的存放庫) 和附加了外部連線的存放庫中的套件版本將會更新，以包含新資產，如中從上游儲存庫保留 Package所述。

一般而言，一旦套件版本保留在儲 CodeArtifact 存庫中，它就不會受到上游儲存庫中的變更影響。如需詳細資訊，請參閱從上游儲存庫保留 Package。但是，具有前面描述的非標準名稱的 Maven 資產的行為是此規則的例外。儘管如果沒有客戶端請求其他資產，下游軟件包版本將不會更改，但在此情況下，保留的軟件包版本在最初保留後會被修改，因此不是不可變的。這種行為是必要的，因為具有非標準名稱的 Maven 資產將無法通過訪問 CodeArtifact。如果它們被添加到公共存儲庫的 Maven 包版本後，該行為也會啟用軟件包版本保留在 CodeArtifact 存儲庫中。

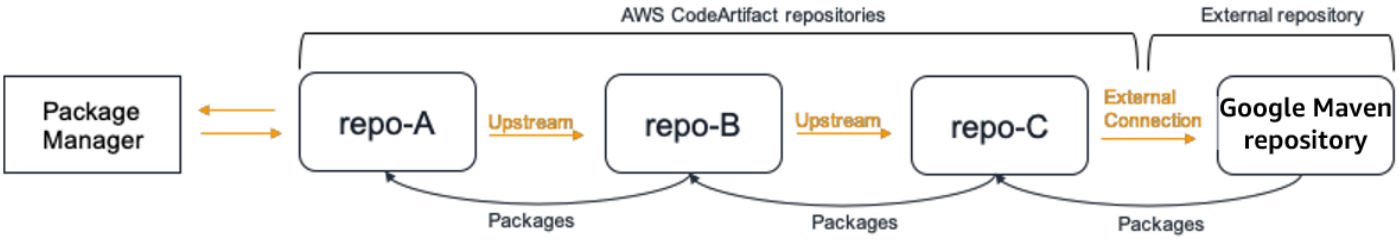
檢查資產來源

將新資產添加到以前保留的 Maven 包版本時， CodeArtifact 確認保留的包版本的來源與新資產的來源相同。這樣可以防止創建「混合」包版本，其中不同的資產來自不同的公共存儲庫。如果沒有此檢查，如果 Maven 軟件包版本發佈到多個公共存儲庫，並且這些存儲庫是存儲庫上游圖的一部分，則可能會發生資產混合。 CodeArtifact

在上游存儲庫中導入新資產和軟件包版本狀態

上游儲存庫中套件版本的套件版本狀態可防 CodeArtifact 止在下游存放庫中保留這些版本。

例如，假設一個域有三個存儲庫：repo-Arepo-Brepo-C，repo-A和，其repo-B中repo-Crepo-B是.



Maven Package 7.3.1 的套com.android.tools.build:aapt2件版本存在於中，repo-B且狀態為Published。它不存在於中repo-A。如果客戶端請求此軟件包版本的資產repo-A，則響應將是 200（OK），Maven 包版本7.3.1將保留在中repo-A。但是，如果軟件包版7.3.1本的狀態repo-B為Archived或Disposed，則響應將為 404（未找到），因為這兩種狀態中的軟件包版本的資產無法下載。

請注意，將套件來源控制項設定upstream=BLOCK為 com.android.tools.build:aapt2 in repo-A repo-B、和repo-C將防止從該套件的所有版本擷取新資產repo-A，而不論套件版本狀態為何。

Maven 的故障

以下信息可以幫助您解決 Maven 和 CodeArtifact.

禁用 parallel 看跌以修復錯誤 429：請求太多

從版本 3.9.0 開始，Maven 會 parallel 上傳套件工件 (一次最多 5 個檔案)。這可能會導 CodeArtifact 致偶爾回應錯誤碼 429 (要求過多) 回應。如果遇到此錯誤，則可以禁用 parallel put 來修復它。

若要停用 parallel 置入，請在檔案false中的設定settings.xml檔中將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 Artifact 品解析程式組態選項。

使用CodeArtifact取代為NuGet

這些主題說明如何使用和發佈NuGet軟件包使用CodeArtifact。

> **Note**
>
> AWS CodeArtifact僅支持NuGet.exe 4.8 版和更高的。

主題

在 Visual Studio 中使用代碼工作

您可以使用 CodeArtifact 憑據提供程序直接在 Visual Studio 中使用 CodeArtifact 包。憑據提供程序簡化了 Visual Studio 中 CodeArtifact Facit 存儲庫的設置和身份驗證，並且可以在AWS Toolkit for Visual Studio。

> **Note**
>
> 所以此AWS Toolkit for Visual Studio不適用於 Mac 的視覺工作室。

要配置 NuGet 並使用 CLI 工具，請參閱CodeArtifact與數字網或網點網 CLI 一起使用。

主題

使用 CodeArtifact 憑據提供程序配置可視化工作室

CodeArtifact 憑據提供程序簡化了 CodeArtifact 和 Visual Studio 之間的設置和繼續身份驗證。CodeArtifact 身分登入資料的有效期最長為 12 小時。為避免在 Visual Studio 中工作時必須手動刷新令牌，憑據提供程序在當前令牌過期之前定期獲取新令牌。

> **Important**
>
> 要使用憑據提供程序，請確保任何現有AWSCodeArtifact 憑據將從`nuget.config`文件，可能已手動添加或通過運行`aws codeartifact login`以配置 NuGet。

在視覺工作室中使用 CodeArtifactAWS Toolkit for Visual Studio

1. 安裝AWS Toolkit for Visual Studio請使用下列步驟。使用這些步驟，該工具包與視覺工作室 2017 和 2019 兼容。AWSCodeArtifact 不支持視覺工作室 2015 年及更早版本。
   1. Toolkit for Visual Studio 2017 和視覺工作室 2019 的工具包分發在Visual Studio Marketplace。您還可以在 Visual Studio 內使用工具≫擴展和更新(Visual Studio 2017) 或擴充≫管理擴充(Visual Studio 2019)。
   2. 安裝工具包後，通過選擇AWS探險者來自檢視選單。
2. 使用 Toolkit in Visual Studio 配置AWS憑據，請遵循提供者AWS登入資料中的AWS Toolkit for Visual Studio使用者指南。
3. （選擇性）設定AWS配置文件，您希望與 CodeArtifact 一起使用。如果未設置，CodeArtifact 將使用默認配置文件。要設置配置文件，請轉到工具 > NuGet 包管理器 > 選擇 CodeArtifactAWS設定檔。
4. 在 Visual Studio 中將您的 CodeArtifact 存儲庫添加為軟件包源。
   1. 導覽至您的儲存庫AWS探險者窗口中，右鍵單擊並選擇`Copy NuGet Source Endpoint`。
   2. 使用工具命令並滾動到NuGet 套件管理工具。
   3. 選取套件來源節點。
   4. 選擇+，編輯名稱，然後將步驟 3a 中複製的存儲庫 URL 終端節點粘貼到來源框，然後選擇更新。
   5. 選中新添加的軟件包源的複選框以啟用它。

> **Note**
>
> 我們建議將外部連接添加到整體組織添加到您的 CodeArtifact 存儲庫，並禁用組織在 Visual Studio 中。當使用外部連接時，所有從整體組織將存儲在您的 CodeArtifact 存儲庫中。如果整體組織變為不可用時，您的應用程序依賴關係仍然可用於 CI 構建和本地開發。如需關於外部連線的詳細資訊，請參將 CodeArtifact 存儲庫 Connect 到公共存儲庫。

5. 重新啟動 Visual Studio，讓變更生效。

配置完成後，Visual Studio 可以使用 CodeArtifact 存儲庫、其任何上遊存儲庫中的軟件包，或從整體組織（如果已添加外部連接）。如需在 Visual Studio 中瀏覽和安裝 NuGet 套件的詳細資訊，請參使用 NuGet 軟件包管理器在可視工作室中安裝和管理軟件包中的NuGet 文檔。

使用可視工作室軟件包管理器控制台

Visual Studio 軟件包管理器控制台將不使用 CodeArtifact 憑據提供程序的 Visual Studio 版本。要使用它，您必須配置命令行憑據提供程序。如需詳細資訊，請參閱「CodeArtifact與數字網或網點網 CLI 一起使用」。

CodeArtifact與數字網或網點網 CLI 一起使用

您可以使用 CLI 工具，例`nuget`如從CodeArtifact.`dotnet` 本文件提供設定 CLI 工具以及使用它們來發佈或使用套件的相關資訊。

主題

- 設定數字或網點網路 CLI
- 使用NuGet套件CodeArtifact
- 將NuGet套件發佈至CodeArtifact
- CodeArtifactNuGet憑證提供者參考
- CodeArtifactNuGet認證提供者版本

設定數字或網點網路 CLI

您可以使用或手動使用CodeArtifactNuGet認證提供者來設定 nuget 或 dotnet CLI。AWS CLI強烈建議您NuGet使用認證提供者進行設定，以簡化設定和持續驗證。

方法 1：使用CodeArtifactNuGet認證提供者進行設定

CodeArtifactNuGet認證提供者可簡化CodeArtifact使用NuGet CLI 工具的驗證和設定。CodeArtifact身分驗證憑證的有效期最長為 12 小時。為了避免在使用 nuget 或 dotnet CLI 時必須手動刷新令牌，憑據提供程序會在當前令牌到期之前定期獲取新令牌。

> **Important**
>
> 若要使用認證提供者，請確定已從您的`nuget.config`檔案中清除任何可能已手動新增的現有AWSCodeArtifact認證，或透過執行NuGet先前設定`aws codeartifact login`來進行設定。

安裝和設定CodeArtifactNuGet認證提供者

dotnet

1. 下載最新版的 AWS。 CodeArtifact。 NuGet。 CredentialProvider使用以下`dotnet`命令來自 NuGet .org 的工具。

   ```
   dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
   ```

2. 使用命`codeartifact-creds install`令將憑證提供者複製到NuGet plugins 資料夾。

   ```
   dotnet codeartifact-creds install
   ```

3. (選擇性)：設AWS定您要與認證提供者搭配使用的設定檔。如果未設定，認證提供者將使用預設設定檔。如需AWS CLI設定檔的詳細資訊，請參閱具名的設定檔。

   ```
   dotnet codeartifact-creds configure set profile profile_name
   ```

nuget

執行下列步驟以使用NuGet CLI 從 Amazon S3 儲存貯體安裝CodeArtifactNuGet登入資料提供者並進行設定。認證提供者將使用預設設定AWS CLI檔，如需有關設定檔的詳細資訊，請參閱具名的設定檔。

1. 從 Amazon S3 儲存貯體下載最新版本的CodeArtifactNuGet登入資料提供者 (codeartifact-nuget-credentialprovider.zip)。

   若要檢視和下載舊版，請參閱CodeArtifactNuGet認證提供者版本。

2. 解壓縮檔案。
3. 複製 AWS。 CodeArtifact。 NuGetCredentialProvider文件夾從網絡文件夾`%user_profile%/.nuget/plugins/netfx/`到視窗或 Linux 或 MacOS`~/.nuget/plugins/netfx` 上。
4. 複製 AWS。 CodeArtifact。 NuGetCredentialProvider文件夾從網絡核心文件夾到`%user_profile%/.nuget/plugins/netcore/`視窗或 Linux 或 MacOS`~/.nuget/plugins/netcore` 上。

建立存放庫並設定認證提供者之後，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 使用NuGet套件CodeArtifact 及 將NuGet套件發佈至CodeArtifact。

方法 2：使用登錄命令配置 nuget 或 dotnet

中的`codeartifact login`命令將存儲庫端點和授權令牌AWS CLI添加到您的NuGet配置文件中，以使 nuget 或 dotnet 連接到您的CodeArtifact存儲庫。這將修改位於視窗和`~/.config/NuGet/NuGet.Config`或`~/.nuget/NuGet/NuGet.Config` Mac /Linux`%appdata%\NuGet\NuGet.Config` 的用戶級NuGet配置。如需有關NuGet組態的詳細資訊，請參閱一般NuGet組態。

使用**`login`**命令配置 nuget 或網點網

1. 設定您的AWS認證以搭配使用AWS CLI，如中所述入門 CodeArtifact。
2. 確定已正確安裝及設定NuGet CLI 工具 (`nuget`或`dotnet`)。如需指示，請參閱 Nuget 或 dotnet 文件。
3. 使用命CodeArtifact`login`令擷取要搭配使用的認證NuGet。

> **Note**
>
> 若您要存取您擁有的網域中的存放庫，則無須納入`--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，因此擷取的認證會以純文字形式儲存在您的設定檔中。

```
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
```

登入指令將會：

- 從CodeArtifact使用您的AWS憑據中獲取授權令牌。
- 使用NuGet套件來源的新項目來更新您的使用者層級NuGet組態。指向您的CodeArtifact存儲庫端點的源將被調用*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權杖。如需有關使用`login`指令建立之授權權杖的詳細資訊，請參閱使用`login`指令建立的權杖。

建立存放庫並設定驗證後，您可以使用`nuget`、`dotnet`、或`msbuild` CLI 用戶端來安裝和發佈套件。如需詳細資訊，請參閱 使用NuGet套件CodeArtifact 及 將NuGet套件發佈至CodeArtifact。

方法 3：在沒有登錄命令的情況下配置 nuget 或 dotnet

對於手動配置，您必須將存儲庫端點和授權令牌添加到NuGet配置文件中，以使 nuget 或 dotnet 連接到您的CodeArtifact存儲庫。

手動配置 nuget 或 dotnet 以連接到您的CodeArtifact存儲庫。

1. 使用`get-repository-endpoint`AWS CLI命令確定您的CodeArtifact存儲庫端點。

```
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. 使用命令從軟件包管理器獲取授權`get-authorization-token`AWS CLI令牌以連接到存儲庫。

```
aws codeartifact get-authorization-token --domain my_domain
```

輸出範例：

```
{
   "authorizationToken": "eyJ2I...viOw",
   "expiration": 1601616533.0
}
```

3. 透過附加至步驟 3 所傳回的 URL`/v3/index.json` 來建立完整的`get-repository-endpoint`存放庫端點 URL。
4. 將 nuget 或 dotnet 配置為使用步驟 1 中的存儲庫端點和步驟 2 中的授權令牌。

> **Note**
>
> 來源網址必須以結尾，nuget 或 dotnet 才能成功連線到CodeArtifact儲存庫。`/v3/index.json`

dotnet

Linux 和 MacOS 使用者：由於非 Windows 平台不支援加密，因此您必須將`--store-password-in-clear-text`旗標新增至下列命令。請注意，這會將您的密碼作為純文本存儲在您的配置文件中。

```
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**
>
> 若要更新既有的來源，請使用`dotnet nuget update source`指令。

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.
```

使用NuGet套件CodeArtifact

NuGet配置完成後CodeArtifact，您可以使用儲存在儲存庫或其上游儲存庫之一中的NuGet套裝程式。CodeArtifact

若要使`nuget`用或使用儲CodeArtifact存庫或其上游儲存庫中的套件版本`dotnet`，請執行下列指令，將 *PackageName ##*為您要使用的套件名稱，並*packageSourceName*使用NuGet組態檔中儲CodeArtifact存庫的來源名稱取代 PackageName。如果您使用此`login`命令來配NuGet置您的組態，則來源名稱為*####/repo_name*。

> **Note**
>
> 當要求套件時，NuGet用戶端會快取該套件的哪些版本存在。由於這種行為，先前在所需版本可用之前請求的套件，安裝可能會失敗。若要避免此失敗並成功安裝現有的套件，您可以在使用或進行安裝之前清除NuGet快取`dotnet nuget locals all --clear`，`nuget locals all --clear`或者透過提供的選項`nuget`或`-NoCache`選項來避免在`install`和`restore`指令期間使用快取`dotnet`。`--no-cache`

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

若要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，請參閱使用 nuget.exe CLI 管理套件或使用 dotnet CLI 安裝和管理套件。

使用NuGet .org 中的NuGet套件

您可以透過CodeArtifact儲存庫使用 NuGet.org 的外部連線來使用 NuGet.org 的NuGet套件。從 NuGet.org 使用的套件會擷取並儲存在您的儲存CodeArtifact庫中。若需新增外部連接的詳細資訊，請參閱將 CodeArtifact 存儲庫 Connect 到公共存儲庫。

將NuGet套件發佈至CodeArtifact

配置NuGet完成後CodeArtifact，您可以使用`nuget`或`dotnet`將套件版本發佈到CodeArtifact儲存庫。

若要將套件版本推送至CodeArtifact儲存庫，請執行下列指令，並在NuGet組態檔中使用.nupkg檔案的完整路徑和CodeArtifact儲存庫的來源名稱。如果您使用login指令來設定NuGet組態，則來源名稱為domain_name/repo_name。

> **Note**
>
> 如果您沒有要發佈的NuGet封裝，您可以建立封裝。如需詳細資訊，請參閱 Microsoft 文件中的 P ackage 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
```

CodeArtifactNuGet憑證提供者參考

CodeArtifactNuGet憑證提供者可讓您輕鬆設定和驗證您NuGet的CodeArtifact儲存庫。

CodeArtifactNuGet認證提供者命令

本節包含CodeArtifactNuGet認證提供者的命令清單。這些命令必須納入前綴，dotnet codeartifact-creds如下列範。

```
dotnet codeartifact-creds command
```

- `configure set profile profile`：設定認證提供者以使用提供的AWS設定檔。
- `configure unset profile`：移除設定的設定檔 (如果已設定)。
- `install`：將認證提供者複製到資plugins料夾。
- `install --profile profile`：將認證提供者複製到plugins資料夾，並將其設定為使用提供的AWS設定檔。
- `uninstall`：解除安裝認證提供者。這不會移除組態檔案的變更。

- `uninstall --delete-configuration`：解除安裝認證提供者，並移除組態檔的所有變更。

CodeArtifactNuGet認證提供者記錄

若要啟用CodeArtifactNuGet認證提供者的記錄，您必須在環境中設定記錄檔。認證提供者記錄檔包含有用的偵錯資訊，例如：

- 用於AWS建立連線的設定檔
- 任何驗證錯誤
- 如果提供的端點不是CodeArtifact URL

設定CodeArtifactNuGet認證提供者記錄檔

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

設定記錄檔之後，任何`codeartifact-creds`指令都會將其記錄輸出附加至該檔案的內容。

CodeArtifactNuGet認證提供者版本

下表包含CodeArtifactNuGet憑證提供者的版本歷程記錄和下載連結。

版本	改變	出版日期	下載連結 (S3)
下 0.1 0.1 (最新版的)	增加了對 net5，net6 和 SSO 配置文件的支持	03/05/2022	下載版 1.0.1 0.1 0.1 0.1 0.1
1.0.0	初始CodeArtifactNuGet認證提供者版本	11/20/2020	下載版 1.0.1 0.1 0.1 0.1 0.1

NuGet軟件包名稱、版本和資產名稱規範化

CodeArtifact在存儲軟件包和資源名稱和軟件包版本之前對它們進行規範化，這意味着CodeArtifact可能與發佈包或資產時提供的不同。

套件名稱規範化：CodeArtifact標準化NuGet軟件包名稱，方法是將所有字母轉換為小寫。

套件版本規範化：CodeArtifact標準化NuGet軟件包版本使用與NuGet。下列資訊來自標準化版本編號來自NuGet文件中)。

- 從版本號中刪除前導零：
  - `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`
- 版本號的第四部分中的零將被省略：
  - `1.0.0.0`被視為`1.0.0`
  - `1.0.01.0`被視為`1.0.1`
- SemVer2.0.0 生成元數據被刪除：
  - `1.0.7+r3456`被視為`1.0.7`

軟件包資產名稱規範化：CodeArtifact建構NuGet包資源名稱來自規範化的軟件包名稱和軟件包版本。

非規範化軟件包名稱和版本名稱可以與 API 和 CLI 請求一起使用，因為CodeArtifact對這些請求的軟件包名稱和版本輸入執行規範化。例如，輸入`--package Newtonsoft.JSON`和`--version 12.0.03.0`將被規範化，並返回一個包的標準化包名稱為`newtonsoft.json`和版本`12.0.3`。

您必須在 API 和 CLI 請求中使用規範化的軟件包資產名稱作為CodeArtifact不會在`--asset`輸入。

必須在 ARN 中使用規範化名稱和版本。

要查找包的標準化名稱，請使用`aws codeartifact list-packages`命令。如需詳細資訊，請參閱列出套件名稱。

要查找包的非標準化名稱，請使用`aws codeartifact describe-package-version`命令。包的非標準化名稱返回在`displayName`欄位。如需詳細資訊，請參閱檢視和更新套件版本詳細資料和相依性。

NuGet 相容性

本指南包含有關代碼工件與不同 NuGet 工具和版本的兼容性的信息。

主題

NuGet 常規相容性

AWSCodeArtifact 支持 NuGet 4.8 及更高版本。

AWSCodeArtifact 僅支持 NuGet HTTP 協議的 V3。這意味着不支持某些依賴於協議 V2 的 CLI 命令。如需詳細資訊，請參閱nuget.exe 命令支持一節。

AWSCodeArtifact 不支持 PowerShellGet 2.x。

NuGet 命令列支持

AWSCodeArtifact 支持 NuGet (`nuget.exe`) 和 .NET Core (`dotnet`) CLI 工具。

nuget.exe 命令支持

因為 CodeArtifact 僅支持 NuGet HTTP 協議的 V3，因此對 CodeArtifact 資源使用以下命令將不起作用：

- `list`：`nuget list`命令會顯示來自給定源的軟件包清單。若要獲取 CodeArtifact Fore 存儲庫中的軟件包清單，您可以使用列出套件名稱命令AWSCLI。

使用 CodeArtifact 斯威夫特

這些主題說明如何搭配使用 Swift Package 件管理員 CodeArtifact 來安裝和發佈 Swift 套件。

> **Note**
>
> CodeArtifact 支持斯威夫特 5.8 及更高版本以及 Xcode 14.3 及更高版本。
> CodeArtifact 推薦斯威夫特 5.9 及更高版本以及 Xcode 15 及更高版本。

主題

使用配置 Swift Package 管理器 CodeArtifact

若要使用 Swift Package 件管理員將套件發佈至或使用套件 AWS CodeArtifact，您必須先設定認證才能存取您的 CodeArtifact 儲存庫。使用 CodeArtifact 憑據和存儲庫端點配置 Swift Package 管理器 CLI 的推薦方法是使用`aws codeartifact login`命令。您也可以手動配置 Swift Package 管理器。

使用登錄命令配置斯威夫特

使用命`aws codeartifact login`令來配置 Swift Package 管理器 CodeArtifact。

> **Note**
>
> 要使用登錄命令，斯威夫特 5.8 或更高版本是必需的，並建議使用 Swift 5.9 或更高版本。

該`aws codeartifact login`命令將執行以下操作：

1. 從中獲取身份驗證令牌 CodeArtifact 並將其存儲在您的環境中。認證的儲存方式取決於環境的作業系統：

   a. macOS：項目是在「macOS 鑰匙圈」應用程式中建立的。

b. Linux 和視窗：在~/.netrc檔案中建立一個項目。

在所有作業系統中，如果有認證項目存在，此命令會以新的 Token 取代該項目。

2. 獲取您的 CodeArtifact 存儲庫端點 URL 並將其添加到您的 Swift 配置文件中。此指令會將儲存庫端點 URL 新增至位於的專案層級組態檔案/path/to/project/.swiftpm/configuration/registries.json。

> **Note**
>
> 該aws codeartifact login命swift package-registry令調用必須從包含該Package.swift文件的目錄中運行的命令。正因為如此，aws codeartifact login命令必須從 Swift 項目中運行。

使用登錄命令配置斯威夫特

1. 導航到包含項目Package.swift文件的 Swift 項目目錄。
2. 執行下列 aws codeartifact login 命令。

   如果您正在訪問您擁有的域中的存儲庫，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此選項會將應用程式設定為只使用 CodeArtifact 儲存庫中的套件 (如果套件位於指定的命名空間中)。CodeArtifact 命名空間是範圍的代名詞，用於將代碼組織到邏輯組中，並防止在代碼庫包含多個庫時可能發生名稱衝突。

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權杖。如需有關使用login指令建立之授權權杖的詳細資訊，請參閱使用login指令建立的權杖。

配置斯威夫特沒有登錄命令

雖然建議您使用aws codeartifact login命令配置 Swift，但您也可以通過手動更新 Swift Package 管理器配置來配置沒有登錄命令的 Swift Package 管理器。

在下列程序中，您將使用執 AWS CLI 行下列作業：

1. 從中獲取身份驗證令牌 CodeArtifact 並將其存儲在您的環境中。認證的儲存方式取決於環境的作業系統：

   a. macOS：項目是在「macOS 鑰匙圈」應用程式中建立的。

   b. Linux 和視窗：在`~/.netrc`檔案中建立一個項目。

2. 獲取存 CodeArtifact 儲庫端點 URL。

3. 在`~/.swiftpm/configuration/registries.json`組態檔案中，新增包含存放庫端點 URL 和驗證類型的項目。

在沒有登錄命令的情況下配置 Swift

1. 在命令行中，使用以下命令獲取 CodeArtifact 授權令牌並將其存儲在環境變量中。

   - 使用您的*#########* CodeArtifact 域名稱。
   - 請以網域擁有者的 AWS 帳號識別碼取代 *111122223333*。如果您正在訪問您擁有的域中的存儲庫，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

   - 視窗 (使用預設命令介面):

     ```
     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
     ```

   - 視窗 PowerShell:

     ```
     $env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
     domain my_domain --domain-owner 111122223333 --query authorizationToken --
     output text
     ```

2. 通過運行以下命令獲取 CodeArtifact 存儲庫的端點。您的存儲庫端點用於將 Swift Package 管理器指向您的存儲庫以使用或發布軟件包。

   - 使用您的*########* CodeArtifact 域名稱。
   - 請以網域擁有者的 AWS 帳號識別碼取代 *111122223333*。如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱 跨帳戶網域。
   - 用您的 CodeArtifact 存儲庫名稱替換 *my_repo*。

   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

   - 視窗 (使用預設命令介面):

     ```
     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
     ```

   - 視窗 PowerShell:

     ```
     $env:CODEARTIFACT_REPO = aws codeartifact get-repository-endpoint --
     domain my_domain --domain-owner 111122223333 --repository my_repo --format
      swift --query repositoryEndpoint --output text
     ```

   以下 URL 是存放庫端點範例。

   ```
   https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   swift/my_repo/
   ```

   > **Important**
   >
   > 當用於配置 Swift Package 管理器時，您必須附加`login`到存儲庫 URL 端點的末尾。這是在此過程的命令中為您完成的。

3. 將這兩個值存儲在環境變量中，使用`swift package-registry login`命令將它們傳遞給斯威夫特，如下所示：

   macOS and Linux

   ```
   swift package-registry login ${CODEARTIFACT_REPO}login --token
    ${CODEARTIFACT_AUTH_TOKEN}
   ```

   Windows

   - 視窗 (使用預設命令介面):

     ```
     swift package-registry login %CODEARTIFACT_REPO%login --token
      %CODEARTIFACT_AUTH_TOKEN%
     ```

   - 視窗 PowerShell:

     ```
     swift package-registry login $Env:CODEARTIFACT_REPO+"login" --token
      $Env:CODEARTIFACT_AUTH_TOKEN
     ```

4. 接下來，更新您的應用程序使用的軟件包註冊表，以便從 CodeArtifact 存儲庫中提取任何依賴項。此命令必須在您嘗試解決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

   - 視窗 (使用預設命令介面):

     ```
     $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
     ```

   - 視窗 PowerShell:

     ```
     $ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
     ```

`--scope`此選項會將應用程式設定為只使用 CodeArtifact 儲存庫中的套件 (如果套件位於指定範圍內)。範圍是CodeArtifact 命名空間的同義詞，可用來將程式碼組織成邏輯群組，以及防止在程式碼庫包含多個程式庫時可能發生名稱衝突。

5. 您可以在專案目錄中執行下列指令，藉此檢視專`.swiftpm/configuration/registries.json`案層級檔案的內容，以確認設定已正確設定：

```
$ 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
}
```

現在您已經使用 CodeArtifact 儲存庫設定了 Swift Package 件管理員，您可以使用它來發佈和使用 Swift 套件，並從中取用。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

使用斯威夫特軟件包 CodeArtifact

使用下列程序來使用 AWS CodeArtifact 儲存庫中的 Swift 套件。

若要使用 CodeArtifact 儲存庫中的 Swift 套件

1. 如果您還沒有，請依照中的步驟設使用配置 Swift Package 管理器 CodeArtifact定「Swift Package 管理員」，以使用適當的認證來使用 CodeArtifact 儲存庫。

> **Note**
>
> 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已經過去了 12 個小時，則需要創建一個新的。

2. 編輯應用程式專Package.swift案資料夾中的檔案，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，請新增一個區段。

   b. 在Package.swift檔案的dependencies區段中，透過新增套件識別碼來新增您要使用的套件。包標識符由範圍和包名稱由一個句點分隔。如需範例，請參閱稍後的步驟之後的程式碼片段。

      > **Tip**
      >
      > 若要尋找套件識別碼，您可以使用主 CodeArtifact 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，請新增一個區段。

   d. 在此targets區段中，新增需要使用相依性的目標。

      下面的代碼片段是一個示例代碼片段dependencies，顯示Package.swift文件中配置的targets部分：

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
    ],
...
```

3. 現在，所有內容都已配置完畢，請使用以下命令從中下載軟件包依賴關係 CodeArtifact。

```
swift package resolve
```

CodeArtifact 在 Xcode 中使用斯威夫特軟件包

使用以下過程從 Xcode 中的 CodeArtifact 存儲庫中使用 Swift 軟件包。

從 Xcode 中的 CodeArtifact 存儲庫中使用 Swift 軟件包

1. 如果您還沒有，請依照中的步驟設使用配置 Swift Package 管理器 CodeArtifact定「Swift Package 管理員」，以使用適當的認證來使用 CodeArtifact 儲存庫。

   > **Note**
   >
   > 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已經過去了 12 個小時，則需要創建一個新的。

2. 在 Xcode 中將軟件包作為依賴項添加到項目中。

   a. 選擇「檔案 > 新增套件」。

   b. 使用搜尋列搜尋您的套件。您的搜索必須在表格中`package_scope.package_name`。

   c. 找到後，選擇軟件包並選擇添加包裹。

   d. 驗證 Package 件後，請選擇要新增為相依性的套件產品，然後選擇「新增套件」。

如果您在 Xcode 中使用 CodeArtifact 存儲庫時遇到問題，請參迅速的故障閱常見問題和可能的修復。

發布斯威夫特軟件包 CodeArtifact

CodeArtifact 推薦 Swift 5.9 或更高版本，並使用該`swift package-registry publish`命令發布 Swift 軟件包。如果您使用的是較早的版本，則必須使用 Curl 命令將 Swift 套件發佈到 CodeArtifact.

使用`swift package-registry publish`指令發佈 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 件管理員」的 CodeArtifact 儲存庫。

1. 如果您還沒有，請依照中的步驟設使用配置 Swift Package 管理器 CodeArtifact定「Swift Package 管理員」，以使用適當的認證來使用 CodeArtifact 儲存庫。

> **Note**
>
> 生成的授權令牌有效期為 12 小時。如果自創建以來已過 12 小時，則需要創建一個新的。

2. 導航到包含包含`Package.swift`文件的 Swift 項目目錄。
3. 執行下列`swift package-registry publish`命令以發佈套件。此指令會建立套件來源封存檔，並將其發佈至您的 CodeArtifact 存放庫。

```
swift package-registry publish packageScope.packageName packageVersion
```

例如：

```
swift package-registry publish myScope.myPackage 1.0.0
```

4. 您可以簽入主控台或使用`aws codeartifact list-packages`指令，來確認套件已發佈且存在於存放庫中，如下所示：

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

您可以使用`aws codeartifact list-package-versions`命令列出單個版本的軟件包，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

雖然建議您使用 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 套件發佈到 CodeArtifact.

請使用下列程序，將 Swift 套件發佈至含有 Curl 的 AWS CodeArtifact 儲存庫。

1. 如果尚未執行，請依照中的步驟建立`CODEARTIFACT_AUTH_TOKEN`並更新和`CODEARTIFACT_REPO`環境變數使用配置 Swift Package 管理器 CodeArtifact。

> **Note**
>
> 身分驗證字符的有效期為 12 小時。如果自建立以來已過 12 小時，您將需要使用新認證重新整理CODEARTIFACT_AUTH_TOKEN環境變數。

2. 首先，如果你沒有創建一個 Swift 包，你可以通過運行以下命令來做到這一點：

```
mkdir testDir && cd testDir
swift package init
git init .
swift package archive-source
```

3. 使用以下 Curl 命令將您的 Swift 包發布到 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. 您可以簽入主控台或使用aws codeartifact list-packages指令，來確認套件已發佈且存在於存放庫中，如下所示：

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

   您可以使用aws codeartifact list-package-versions命令列出單個版本的軟件包，如下所示：

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

從中獲取 Swift 軟件包 GitHub 並重新發布到 CodeArtifact

請使用下列程序來擷取 Swift 套件， GitHub 並將其重新發佈至 CodeArtifact 儲存庫。

要從中獲取 Swift 包 GitHub 並將其重新發布到 CodeArtifact

1. 如果您還沒有，請依照中的步驟設使用配置 Swift Package 管理器 CodeArtifact定「Swift Package 管理員」，以使用適當的認證來使用 CodeArtifact 儲存庫。

   > **Note**
   >
   > 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已經過去了 12 個小時，則需要創建一個新的。

2. 克隆你想獲取並通過使用下面的`git clone`命令重新發布斯威夫特包的 git 存儲庫。如需複製 GitHub 儲存庫的詳細資訊，請參閱在 GitHub 文件中複製儲存庫。

   ```
   git clone repoURL
   ```

3. 導航到您剛剛克隆的存儲庫：

   ```
   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`指令建立套件，並將其發佈到您設定的 CodeArtifact 儲存庫。

      ```
      swift package-registry publish packageScope.packageName versionNumber
      ```

      例如：

      ```
      swift package-registry publish myScope.myPackage 1.0.0
      ```

   b. 如果您使用的是早於 5.9 的 Swift 版本，則必須使用該`swift archive-source`命令來創建包，然後使用 Curl 命令發布它。

      i. 如果您尚未設定`CODEARTIFACT_AUTH_TOKEN`和`CODEARTIFACT_REPO`環境變數，或已經超過 12 小時，請依照中的步驟執行配置斯威夫特沒有登錄命令。

      ii. 通過使用`swift package archive-source`命令創建斯威夫特包：

```
swift package archive-source
```

如果成功，您將Created *package_name*.zip在命令行中看到。

iii. 使用下面的 Curl 命令來發布斯威夫特包 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. 您可以簽入主控台或使用aws codeartifact list-packages指令，來確認套件已發佈且存在於存放庫中，如下所示：

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

您可以使用aws codeartifact list-package-versions命令列出單個版本的軟件包，如下所示：

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Swift 包名稱和命名空間規範化

CodeArtifact 在儲存套件名稱和命名空間之前將其標準化，這意味著中的名稱 CodeArtifact 可能與發佈套件時提供的名稱不同。

P@@ ackage 名稱和命名空間規範 CodeArtifact 化：通過將所有字母轉換為小寫來標準化 Swift 包名稱和命名空間。

Package 版本規範化： CodeArtifact 不會標準化 Swift 軟件包版本。請注意， CodeArtifact 僅支援語意版本化 2.0 版本模式，如需有關語意化版本控制的詳細資訊，請參閱語意化版本 2.0.0。

非標準化套件名稱和命名空間可與 API 和 CLI 要求搭配使用，因為 CodeArtifact 會對這些要求的輸入執行標準化。例如，`--package myPackage`和的輸入`--namespace myScope`會被標準化，並傳回一個包含標準化套件名稱`mypackage`和命名空間的套件。`myscope`

您必須在 ARN 中使用標準化名稱，例如 IAM 政策。

若要尋找套件的標準化名稱，請使用指`aws codeartifact list-packages`令。如需詳細資訊，請參閱 列出套件名稱。

迅速的故障

下列資訊可協助您疑難排解 Swift 和 CodeArtifact.

即使在配置 Swift Package 管理器後，我也在 Xcode 中收到 401 錯誤

問題：當您嘗試從 CodeArtifact 存儲庫中添加一個軟件包作為 Xcode 中的 Swift 項目的依賴項時，即使在按照將 Swift 連接到的說明之後，也會收到 401 未經授權的錯誤 CodeArtifact。

可能的修復：這可能是由 macOS 鑰匙串應用程序存儲您的 CodeArtifact 憑據的問題引起的。若要修正此問題，我們建議您開啟 Keychain 應用程式並刪除所有 CodeArtifact 項目，然後依照中的指示，再次使用您的 CodeArtifact 儲存庫配置 Swift Package 管理員使用配置 Swift Package 管理器 CodeArtifact。

由於鑰匙串提示輸入密碼，Xcode 掛在 CI 機器上

問題：當您嘗試將 Swift 包作 CodeArtifact 為持續集成（CI）服務器上的 Xcode 構建的一部分（例如使用 GitHub Actions）提取 Swift 包時，身份驗證 CodeArtifact 可能會掛起並最終失敗，並顯示類似於以下內容的錯誤消息：

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

可能的修復：這是由於憑據未保存到 CI 機器上的鑰匙串中，並且 Xcode 僅支持保存在鑰匙串中的憑據引起的。若要解決此問題，我們建議您使用下列步驟手動建立鑰匙圈項目：

1. 準備鑰匙串。

```
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. 獲取身 CodeArtifact 份驗證令牌和存儲庫端點。

```
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. 手動創建鑰匙串條目。

```
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}"
```

如需有關此錯誤及解決方案的詳細資訊，請參閱 https://github.com/apple/ swift-package-manager /問題/ 7236。

使用 CodeArtifact 紅寶石

這些主題說明如何使用 RubyGems 和捆綁器工具 CodeArtifact 來安裝和發佈 Ruby 寶石。

> **Note**
>
> CodeArtifact 建議紅寶石 3.3 或更高版本，並且不適用於 Ruby 2.6 或更早版本。

主題

配置和使用 RubyGems 以及捆綁器 CodeArtifact

在中建立儲存庫之後 CodeArtifact，您可以使用 RubyGems (`gem`) 和 Bundler (`bundle`) 來安裝和發佈寶石。本主題說明如何設定套件管理員來驗證和使用 CodeArtifact 存放庫。

配置 RubyGems （**`gem`**）和捆綁器（**`bundle`**） CodeArtifact

要使用 RubyGems (`gem`) 或 Bundler (`bundle`) 向寶石發布或使用寶石 AWS CodeArtifact，首先需要使用 CodeArtifact 存儲庫信息配置它們，包括憑據才能訪問它們。請遵循下列其中一個程序中的步驟，使用 CodeArtifact 存放庫端點資訊`gem`和認證來設定和 `bundle` CLI 工具。

使用控制台說明配置 RubyGems 和捆綁器

您可以使用控制台中的配置說明將 Ruby 包管理器連接到存 CodeArtifact 儲庫。控制台指示提供自定義命令，您可以運行這些命令來設置您的軟件包管理器，而無需查找和填寫您的 CodeArtifact 信息。

1. 請在以下位置開啟 AWS CodeArtifact 主控台。 https://console.aws.amazon.com/codesuite/codeartifact/home
2. 在瀏覽窗格中，選擇 [存放庫]，然後選擇要用於安裝或推送 Ruby gem 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. 按照生成的說明配置軟件包管理器客戶端以從中安裝 Ruby gem 或將 Ruby gem 發佈到存儲庫。

手動配置 RubyGems 和捆綁器

如果您無法或不想使用主控台中的組態指示，您可以使用下列指示手動連線到 Ruby 套件管理員至儲 CodeArtifact 存庫。

1. 在命令行中，使用以下命令獲取 CodeArtifact 授權令牌並將其存儲在環境變量中。

   - 使用您的*#########* CodeArtifact 域名稱。
   - 以網域擁有者的 AWS 帳號識別碼取代 *111122223333*。如果您正在訪問您擁有的域中的存儲庫，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

   - 視窗 (使用預設命令介面):

     ```
     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
     ```

   - 視窗 PowerShell:

     ```
     $env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
     domain my_domain --domain-owner 111122223333 --query authorizationToken --
     output text
     ```

2. 要將 Ruby gem 發布到存儲庫，請使用以下命令獲取存儲 CodeArtifact 庫的端點並將其存儲在`RUBYGEMS_HOST`環境變量中。gemCLI 會使用此環境變數來判斷 gem 的發佈位置。

   > **Note**
   >
   > 或者，您可以在使用`gem push`指令時為儲存庫端點提供`--host`選項，而不是使用`RUBYGEMS_HOST`環境變數。

- 使用您的*#########* CodeArtifact 域名稱。
- 以網域擁有者的 AWS 帳號識別碼取代 *111122223333*。如果您正在訪問您擁有的域中的存儲庫，則不需要包含`--domain-owner`。如需詳細資訊，請參閱 跨帳戶網域。
- 用您的 CodeArtifact 存儲庫名稱替換 *my_repo*。

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

以下命令檢索存儲庫端點，修剪尾隨/，然後將它們存儲在環境變量中。

- 視窗 (使用預設命令介面):

```
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%
```

- 視窗 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("/")
```

以下 URL 是存放庫端點範例：

```
https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/
```

3. 要將 Ruby gem 發布到您的存儲庫，您必須 RubyGems 通過編輯`~/.gem/credentials`文件以 CodeArtifact 包含身份驗證令牌來對其進行身份驗證。如果`~/.gem/`目錄或`~/.gem/credentials`文件不存在，請創建一個目錄和文件。

macOS and Linux

```
echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/
credentials
```

Windows

- 視窗 (使用預設命令介面):

```
echo :codeartifact_api_key: Bearer %CODEARTIFACT_AUTH_TOKEN% >> %USERPROFILE
%/.gem/credentials
```

- 視窗 PowerShell:

```
echo ":codeartifact_api_key: Bearer $env:CODEARTIFACT_AUTH_TOKEN" | Add-
Content ~/.gem/credentials
```

4. gem要使用從存儲庫安裝 Ruby gem，您必須將存儲庫端點信息和身份驗證令牌添加到.gemrc文件中。您可以將其添加到全局文件（~/.gemrc）或項目.gemrc文件中。您必須添加到的 CodeArtifact 信息.gemrc是存儲庫端點和身份驗證令牌的組合。它的格式如下：

```
https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/
```

- 對於驗證 Token，您可以使用在先前步驟中設定的CODEARTIFACT_AUTH_TOKEN環境變數。
- 要獲取存儲庫端點，您可以讀取之前設置的RUBYGEMS_HOST環境變量的值，或者您可以使用以下get-repository-endpoint命令，根據需要替換值：

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint
 --output text
```

取得端點後，請使用文字編輯器aws:${CODEARTIFACT_AUTH_TOKEN}@在適當的位置加入。創建存儲庫端點和 auth 令牌字符串後，請使用以下echo命令將其添加到.gemrc文件的:sources:部分中：

> **Warning**
>
> CodeArtifact 不支援使用`gem sources -add`指令將儲存庫新增為來源。您必須將源直接添加到文件中。

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

- 視窗 (使用預設命令介面):

```
echo ":sources:
    - https://aws:%CODEARTIFACT_AUTH_TOKEN
%@my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/"
 > "%USERPROFILE%\.gemrc"
```

- 視窗 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. 要使用 Bundler，您必須通過運行以下命令來配置 Bundler 與存儲庫端點 URL 和身份驗證令牌：`bundle config`

   macOS and Linux

   ```
   bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
   ```

   Windows

   - 視窗 (使用預設命令介面):

```
bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
```

- 視窗 PowerShell:

```
bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
```

現在您已經使用 CodeArtifact 存儲庫配置了 RubyGems （ gembundle ）和 Bundler （ ），您可以使用它們來發布和使用 Ruby 寶石。

從安裝紅寶石寶石 CodeArtifact

使用下列程序，透過gem或 bundle CLI 工具從 CodeArtifact 儲存庫安裝 Ruby gem。

安裝紅寶石寶石 **gem**

您可以使用 RubyGems （ gem ）CLI 從 CodeArtifact 存儲庫快速安裝特定版本的 Ruby gem。

要從 CodeArtifact 存儲庫安裝 Ruby 寶石 **gem**

1. 如果尚未執行，請依照中的步驟設配置 RubyGems （ gem ）和捆綁器（ bundle ） CodeArtifact定 gem CLI，使其具有適當認證使用 CodeArtifact 存放庫。

   > **Note**
   >
   > 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已過 12 小時，則需要創建一個新的。

2. 使用以下命令從安裝 Ruby 寶石 CodeArtifact：

   ```
   gem install my_ruby_gem --version 1.0.0
   ```

安裝紅寶石寶石 **bundle**

您可以使用捆綁器（ bundle ）CLI 來安裝Gemfile在.

要從 CodeArtifact 存儲庫安裝 Ruby 寶石 **`bundle`**

1. 如果尚未執行，請依照中的步驟設[配置 RubyGems （ gem ）和捆綁器（ bundle ） CodeArtifact](#)定 `bundle` CLI，使其具有適當認證使用 CodeArtifact 存放庫。

   > **Note**
   >
   > 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已過 12 小時，則需要創建一個新的。

2. 將您的 CodeArtifact 存儲庫端點 URL 添加到您的`Gemfile`作為`source`以從 CodeArtifact 存儲庫及其上流安裝配置的 Ruby gem。

   ```
   source "https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
   ruby/my_repo/"

   gem 'my_ruby_gem'
   ```

3. 使用下面的命令來安裝 Ruby 寶石中指定的`Gemfile`：

   ```
   bundle install
   ```

發布紅寶石寶石 CodeArtifact

請使用下列程序，使用 `gem` CLI 將 Ruby 寶石發佈至 CodeArtifact 儲存庫。

1. 如果尚未執行，請依照中的步驟設[配置 RubyGems （ gem ）和捆綁器（ bundle ） CodeArtifact](#)定 `gem` CLI，使其具有適當認證使用 CodeArtifact 存放庫。

   > **Note**
   >
   > 生成的授權令牌有效期為 12 小時。如果自創建令牌以來已過 12 小時，則需要創建一個新的。

2. 使用以下命令將 Ruby 寶石發布到存 CodeArtifact 儲庫。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 指令支援

CodeArtifact 支援`gem install`和`gem push`指令。 CodeArtifact 不支援下列`gem`指令：

- `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`

CodeArtifact 搭配泛型套件使用

這些主題向您展示如何使用 AWS CodeArtifact.

主題

一般套件概觀

使用generic封裝格式，您可以上傳任何類型的檔案，以在 CodeArtifact 存放庫中建立套件。通用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。這對於存儲和版本化任意構建成品（例如應用程序安裝程序，機器學習模型，配置文件等）非常有用。

通用套件包含套件名稱、命名空間、版本，以及一或多個資產 (或檔案)。一般套件可以與單一 CodeArtifact 儲存庫中其他格式的套件一起存在。

您可以使用 AWS CLI 或 SDK 來處理一般套件。如需與通用套件搭配使用的 AWS CLI 命令的完整清單，請參閱一般套件支援的指令。

通用包約束

- 它們永遠不會從上游存儲庫中提取。它們只能從發佈目標的存放庫中取得。
- 它們無法宣告從中傳回ListPackageVersionDependencies或顯示在中的相依性 AWS Management Console 。
- 他們可以存儲自述文件和許可證文件，但它們不會被解釋 CodeArtifact。這些檔案中的資訊不會從GetPackageVersionReadme或傳回 DescribePackageVersion，也不會出現在 AWS Management Console.
- 就像中的所有套件一樣 CodeArtifact，資產大小和每個套件的資產數量都有限制。如需中限制和配額的詳細資訊 CodeArtifact，請參閱配額 AWS CodeArtifact。
- 它們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，允許使用這些 Unicode 字元類別：小寫字母 (Ll)、修飾符字母 (Lm)、其他字母 (Lo)、標題大寫字母 (Lt)、大寫字母 (Lu)、字母編號 (Nl) 和十進位數字 (Nd)。

- 允許使用以下特殊字元：~!@^&()-_+[]{};,.
- 資產無法命名.或 ..
- 空格是唯一允許的空白字符。資產名稱不能以空格字元開頭或結尾，也不能包含連續的空格。

一般套件支援的指令

您可以使用 AWS CLI 或 SDK 來處理一般套件。下列 CodeArtifact 命令適用於通用套件：

- copy-package-versions(請參閱在儲存庫間複製套件)
- 刪除套件 (請參閱) 刪除套件 (AWS CLI)
- delete-package-versions(請參閱刪除套件版本 (AWS CLI))
- 描述包
- describe-package-version(請參閱檢視和更新套件版本詳細資料和相依性)
- 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(請參閱列出軟件包版本)
- 列表包（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(請參閱編輯套件原點控制項)

> **Note**
>
> 您可以使用publish原始控制項設定來允許或封鎖在存放庫中發佈一般套件名稱。但是，此upstream設定不適用於一般套件，因為無法從上游存放庫擷取這些套件。

- 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指令。您可以使用list-package-version-asset命令列出通用軟件包的資產，然後使用get-package-version-asset. 下列主題包含使用這些指令發行一般套件或下載一般套件資產的指 step-by-step 示。

發佈一般套件

通用套件包含套件名稱、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`，名稱空間`my-ns`、版本`1.0.0`，以及包含一個名為的資產`asset.tar.gz`。

先決條件：

- 設定和設定 AWS Command Line Interface 使用 CodeArtifact (請參閱設定使用 AWS CodeArtifact)
- 擁有 CodeArtifact 網域和儲存庫 (請參閱AWS CLI 使用入門)

若要發佈一般套件

1. 使用下列指令為您要上傳至套件版本的每個檔案產生 SHA256 雜湊，並將該值置於環境變數中。此值用作完整性檢查，以驗證檔案內容在最初傳送之後是否未變更。

   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，您可以針對每個要上傳的資產呼叫一`publish-package-version`次。除了上傳最終資產時`publish-package-version`，請包含每次呼叫的`--unfinished`引數。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至該套件。

或者，每--unfinished次呼叫都包含publish-package-version，然後Published使用update-package-versions-status指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

若要列出一般套件中包含的資源，請使用`list-package-version-assets`指令。如需詳細資訊，請參閱 列出套件版本資產。

下面的例子列出了包的版本`1.0.0`的資產`my-package`。

若要列出套件版本資產

- 調用`list-package-version-assets`以列出通用包中包含的資產。

  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源，請使用get-package-version-asset指令。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my-package件1.0.0的版本下載到目前的工作目錄中，並且名為asset.tar.gz。

下載套件版本資產

- 調get-package-version-asset用從通用包下載資產。

  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"
}
```

使用CodeArtifact與CodeBuild

這些主題說明如何在CodeArtifact儲存庫AWS CodeBuild構建項目。

主題

在中使用 npm 套件CodeBuild

以下步驟已通過列出的操作系統進行了測試碼頭圖片提供CodeBuild。

使用 IAM 角色設定許可

使用 npm 套件時需要這些步驟CodeArtifact在CodeBuild。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)。在「」角色」頁面上，編輯您使用的角色CodeBuild構建項目。此角色必須具有下列權限。

```
{
  "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**
>
> 如果你也想使用CodeBuild若要發佈套件，請新增**codeartifact:PublishPackageVersion**權限。

如需資訊，請參閱〈修改角色在IAM 使用者指南。

登入並使用 npm

若要從中使用 npm 套件CodeBuild，執行`login`來自的命令`pre-build`項目的部分`buildspec.yaml`進行配置npm從中獲取軟件包CodeArtifact。如需詳細資訊，請參閱使用 npm 進行驗證。

之後`login`已成功運行，您可以運行`npm`來自的指令`build`部分安裝 npm 軟件包。

Linux

> **Note**
>
> 只需要升級AWS CLI與`pip3 install awscli --upgrade --user`如果您使用的是較舊的CodeBuild圖像。如果您使用的是最新的映像版本，則可以刪除該行。

```
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
```

在中使用套件CodeBuild

以下步驟已通過列出的操作系統進行了測試碼頭圖片提供CodeBuild。

使用 IAM 角色設定許可

使用 Python 套件時，需要執行這些步驟CodeArtifact在CodeBuild。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)。在「」角色」頁面上，編輯您使用的角色CodeBuild構建項目。此角色必須具有下列權限。

```
{
  "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**
>
> 如果你也想使用CodeBuild若要發佈套件，請新增**`codeartifact:PublishPackageVersion`**權限。

如需資訊，請參閱〈修改角色在IAM 使用者指南。

登錄並使用 pip 或麻線

若要使用 Python 套件CodeBuild，執行`login`來自的命令`pre-build`項目的部分`buildspec.yaml`要設定的檔案`pip`從中獲取軟件包CodeArtifact。如需詳細資訊，請參閱使用CodeArtifact與蟒蛇。

之後`login`已成功運行，您可以運行`pip`來自的指令`build`部分以安裝或發布 Python 軟件包。

Linux

> **Note**
>
> 只需要升級AWS CLI與`pip3 install awscli --upgrade --user`如果您使用的是較舊的CodeBuild圖像。如果您使用的是最新的映像版本，則可以刪除該行。

若要使用安裝 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

若要使用安裝 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
```

在中使用 Maven 軟件包CodeBuild

使用 IAM 角色設定許可

使用 Maven 軟件包時需要這些步驟CodeArtifact在CodeBuild。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)。在「」角色」頁面上，編輯您使用的角色CodeBuild構建項目。此角色必須具有下列權限。

```
{
  "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**
>
> 如果你也想使用CodeBuild若要發佈套件，請新增**codeartifact:PublishPackageVersion**和**codeartifact:PutPackageMetadata**權限。

如需資訊，請參閱〈修改角色在IAM 使用者指南。

使用搖籃或 MVN

若要搭配使用 Maven 套件gradle或者mvn，存儲CodeArtifact環境變量中的 auth 令牌，如中所述在環境變量中傳遞身份驗證令牌。以下是範例。

> **Note**
>
> 只需要升級AWS CLI與pip3 install awscli --upgrade --user如果您使用的是較舊的CodeBuild圖像。如果您使用的是最新的映像版本，則可以刪除該行。

```
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`
```

要使用搖籃：

如果您參考CODEARTIFACT_AUTH_TOKEN搖籃中的變量build.gradle檔案，如中所述使用CodeArtifact與搖籃，您可以從中調用您的 Gradle 構建buildspec.yaml build部分。

```
build:
  commands:
    - gradle build
```

要使用 mvn：

你必須配置你的 Maven 配置文件（`settings.xml`和`pom.xml`) 按照中的說明進行操作使用CodeArtifact與 MVN。

使用NuGet中的套件CodeBuild

以下步驟已通過列出的操作系統進行了測試碼頭圖片提供CodeBuild。

主題

- 使用 IAM 角色設定許可
- 消耗NuGet套件
- 使用建置NuGet套件
- 發佈NuGet套件

使用 IAM 角色設定許可

使用時需要這些步驟NuGet套件來自CodeArtifact在CodeBuild。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)。在「」角色」頁面上，編輯您使用的角色CodeBuild構建項目。此角色必須具有下列權限。

```
{
  "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**
>
> 如果你也想使用CodeBuild若要發佈套件，請新增**`codeartifact:PublishPackageVersion`**權限。

如需資訊，請參閱〈修改角色在IAM 使用者指南。

消耗NuGet套件

要消費NuGet套件來自CodeBuild，在您的項目中包含以下內容`buildspec.yaml`文件。

1. 在`install`區段中，安裝CodeArtifact憑證提供者，用於配置命令行工具，例如`msbuild`和`dotnet`建置套件並將其發佈至CodeArtifact。
2. 在`pre-build`」區段中，新增您的CodeArtifact存儲庫到您的NuGet配置。

請參閱以下內容`buildspec.yaml`例子。如需詳細資訊，請參閱使用CodeArtifact取代為NuGet。

安裝認證提供者並新增存放庫來源之後，您就可以執行NuGetCLI 工具命令來自`build`要消耗的部分NuGet套件。

Linux

要消費NuGet套件使用`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

要消費NuGet套件使用`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
```

使用建置NuGet套件

若要建置NuGet套件來自CodeBuild，在您的項目中包含以下內容`buildspec.yaml`文件。

1. 在install區段中，安裝CodeArtifact憑證提供者，用於配置命令行工具，例如msbuild和dotnet建置套件並將其發佈至CodeArtifact。
2. 在pre-build」區段中，新增您的CodeArtifact存儲庫到您的NuGet配置。

請參閱以下內容buildspec.yaml例子。如需詳細資訊，請參閱使用CodeArtifact取代為NuGet。

安裝認證提供者並新增存放庫來源之後，您就可以執行NuGet像這樣的 CLI 工具命令dotnet build從build部分。

Linux

若要建置NuGet套件使用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
```

若要建置NuGet套件使用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

若要建置NuGet套件使用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
```

若要建置NuGet套件使用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

若要發佈NuGet套件來自CodeBuild，在您的項目中包含以下內容`buildspec.yaml`文件。

1. 在`install`區段中，安裝CodeArtifact憑證提供者，用於配置命令行工具，例如`msbuild`和`dotnet`建置套件並將其發佈至CodeArtifact。
2. 在`pre-build`」區段中，新增您的CodeArtifact存儲庫到您的NuGet配置。

請參閱以下內容`buildspec.yaml`例子。如需詳細資訊，請參閱使用CodeArtifact取代為NuGet。

安裝認證提供者並新增存放庫來源之後，您就可以執行NuGetCLI 工具命令來自`build`部分並發布您的NuGet套件。

Linux

若要發佈NuGet套件使用`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

若要發佈NuGet套件使用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

您可以啟用本地緩存CodeBuild減少需要從中獲取的依賴關係的數量CodeArtifact對於每個構建。如需資訊，請參閱〈在中構建緩存AWS CodeBuild在AWS CodeBuild使用者指南。啟用自定義本地緩存後，將緩存目錄添加到項目的buildspec.yaml文件。

例如，如果您正在使用mvn，使用以下內容。

```
cache:
  paths:
    - '/root/.m2/**/*'
```

對於其他工具，請使用此表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

監控是維持其他 AWS 解決方案的可靠性、可用性和效能的 CodeArtifact 重要組成部分。 AWS 提供下列監控工具來監視 CodeArtifact、在發生錯誤時回報，並在適當時自動採取行動：

- 您可以使用 Amazon EventBridge 自動化 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。來自 AWS 服務的事件會以近乎即時 EventBridge 的方式傳送到。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 事件格式與範例.
- 您可以使用 Amazon CloudWatch 指標按操作查看 CodeArtifact 使用情況。 CloudWatch 量度包括向其發出的所有要求 CodeArtifact，而請求則會依帳戶顯示。您可以瀏覽至「使用量/依 AWS 資源 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 使用者指南中的使 CloudWatch 用 Amazon 指 CloudWatch 標。

主題

- 監控 CodeArtifact 事件
- 使用事件開始 CodePipeline執行
- 使用事件執行 Lambda 函數

監控 CodeArtifact 事件

CodeArtifact 與 Amazon 整合 EventBridge，這是一項可自動化和回應事件 (包括 CodeArtifact 儲存庫中的變更) 的服務。您可以為事件建立規則，並設定事件符合規則時會發生什麼情況。 EventBridge 以前稱為「 CloudWatch 事件」。

事件可以觸發下列動作：

- 調用一個 AWS Lambda 函數。
- 啟動 AWS Step Functions 狀態機。
- 通知 Amazon SNS 主題或 Amazon SQS 隊列。
- 在中啟動管線 AWS CodePipeline。

CodeArtifact 建立、修改或刪除封裝版本時建立事件。以下是 CodeArtifact 事件的範例：

- 發佈新的套件版本 (例如，執行`npm publish`)。

- 將新資源添加到現有的軟件包版本（例如，通過將新的 JAR 文件推送到現有的 Maven 包中）。
- 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 在儲存庫間複製套件。
- 刪除套件版本使用`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。
- 從上游存放庫擷取下游存放庫時，將其保留在下游存放庫中的套件版本。如需詳細資訊，請參閱 使用中的上游存儲庫 CodeArtifact。
- 將套件版本從外部存放庫導入存放 CodeArtifact 庫。如需詳細資訊，請參閱 將 CodeArtifact 存儲庫 Connect 到公共存儲庫。

事件會傳遞至擁有網域的帳戶和管理存放庫的帳戶。例如，假設該帳戶`111111111111`擁有網域`my_domain`。帳戶在調用中`222222222222`創建一`my_domain`個存儲庫`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存放庫，則只會傳遞單一事件。

下列主題說明 CodeArtifact 事件格式。它們會示範如何設定 CodeArtifact事件，以及如何搭配其他 AWS 服務使用事件。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的 Amazon 入門。

CodeArtifact 事件格式與範例

以下是事件欄位和說明，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。目前只有一個版本，`0`.
id	事件的唯一識別碼。
詳細資訊類型	事件的類型。這決定了`detail`對象中的字段。`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，這將是`aws.codeartifact` 。
帳戶	接收事件之帳戶的帳戶 ID。 AWS
time	觸發事件的確切時間。
region	觸發事件的區域。
resources	包含已變更套件 ARN 的清單。清單包含一個項目。如需封裝 ARN 格式的資訊，請參閱授與封裝的寫入存取權。
domainName	包含包含套件之存放庫的網域。
網域擁有者	網域擁有者的 AWS 帳戶 ID。
儲存庫名稱	包含套件的存放庫。
儲存庫管理員	儲存庫管理員的 AWS 帳號 ID。
套件格式	觸發事件的封裝格式。
包裝空間	觸發事件之封裝的命名空間。
封裝名稱	觸發事件的封裝名稱。
套件版本	觸發事件的套件版本。
packageVersionState	觸發事件時的封裝版本狀態。可能值為 `Unfinished` 、`Published` 、`Unlisted`、`Archived` 和 `Disposed`。
packageVersionRevision	此值可唯一識別觸發事件時封裝版本的資產和中繼資料狀態。如果套件版本遭到修改 (例如，將另一個 JAR 檔案新增至 Maven 套件)，則會`packageVersionRevision` 變更。

事件欄位	描述
更改。資產添加	新增至觸發事件之封裝的資產數目。一個資產的例子是一個 Maven 的 JAR 文件或一個 Python 輪。
變更。已移除資產	從封裝中移除的資產數目，觸發事件。
變更。資產更改	在封裝中修改的資產數目，以觸發事件。
變更。中繼資料保護	Boolean 值，`true`如果事件包含修改的套件層級中繼資料，則設定為。例如，事件可能會修改 Maven `pom.xml` 檔案。
變更。狀態已變更	布林值，在事件被修改時設定為 (例如，如果從`packageVersionStatus` 變更`Unfinished` 為`Published`)。`true` `packageVersionStatus`
操作類型	描述封裝版本變更的高階類型。可能的值為 `Created`、`Updated` 和 `Deleted`。
序列編號	整數；指定封裝的事件編號。一個包上的每個事件都會增加`sequenceNumber` 所以事件可以按順序排列。事件可以使用任何整數`sequenceNumber` 來遞增。 **Note** EventBridge 事件可能接收不順序。`sequenceNumber` 可用於確定其實際訂單。
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叫用相同的目標。

CodeArtifact 事件範例

以下是發佈 npm 套 CodeArtifact 件時可能會觸發的事件範例。

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

使用事件開始 CodePipeline執行

本範例示範如何設定 Amazon EventBridge 規則，以便在發佈、修改或刪除 CodeArtifact 儲存庫中的套件版本時開始 AWS CodePipeline 執行。

主題

設定 EventBridge權限

您必須新增權限， EventBridge 才能叫用 CodePipeline 您建立的規則。若要使用 AWS Command Line Interface (AWS CLI) 新增這些權限，請遵循使用指南中的為 CodeCommit 來源建立 CloudWatch 事件規則 (CLI) 中AWS CodePipeline 的步驟 1。

建立規 EventBridge則

若要建立規則，請將指`put-rule`令與`--name`和`--event-pattern`參數搭配使用。事件模式指定與每個事件的內容匹配的值。如果模式與事件匹配，則觸發目標。例如，下列模式會比對`my_domain`網域中`myrepo`儲存庫中的 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"]}}'
```

建立規 EventBridge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。對於`RoleArn`參數，請指定本主題稍早建立之角色的 Amazon 資源名稱 (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'
```

使用事件執行 Lambda 函數

此範例說明如何設定在發行、修改或刪除 CodeArtifact 存放庫中的套件版本時啟動 AWS Lambda 函數的 EventBridge 規則。

如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南 EventBridge中的教學課程：排程 AWS Lambda 函數使用。

主題

- 建立規 EventBridge 則
- 建立規 EventBridge 則目標
- 設定 EventBridge權限

建立規 EventBridge 則

若要建立啟動 Lambda 函數的規則，請搭配--name和--event-pattern選項使用put-rule指令。下列模式會指定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"]}}'
```

建立規 EventBridge 則目標

下列命令會在事件符合規則時，將目標新增至執行 Lambda 函數的規則。對於arn參數，請指定 Lambda 函數的 Amazon 資源名稱 (ARN)。

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

設定 EventBridge權限

使用此add-permission命令授與規則叫用 Lambda 函數的權限。對於--source-arn參數，請指定您先前在此範例中建立之規則的 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

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。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：

- 雲端本身的安全 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計劃的一部分。若要了解適用於 CodeArtifact 的合規計畫，請參閱合規計畫的 AWS 服務範圍。
- 雲端內部的安全：您的責任取決於所使用的 AWS 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。

本文件有助於您了解如何在使用 CodeArtifact 時套用共同責任模型。下列主題說明如何將 CodeArtifact 設定為達到您的安全及合規目標。您也將了解如何使用其他 AWS 服務來協助您監控並保護 CodeArtifact 資源。

主題

- AWS CodeArtifact 中的資料保護
- 監控 CodeArtifact
- AWS CodeArtifact 的合規驗證
- AWS CodeArtifact 身份驗證和令牌
- 韌性在AWS CodeArtifact
- 基礎結構安全AWS CodeArtifact
- 依賴替換攻擊
- 的 Identity and Access Management AWS CodeArtifact

AWS CodeArtifact 中的資料保護

AWS共同責任模型適用於 AWS 中的資料保護 CodeArtifact。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。您也必須負責您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，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，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。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 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 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。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，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) 140-2 概觀。

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 Name (名稱) 欄位。這包括當您使用主控台、API CodeArtifact 或 AWS SDK 時AWS 服務使用或其他使用時。AWS CLI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

加密是安 CodeArtifact 全性的重要組成部分。某些加密 (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進行設定。

- 靜態資料加密-所有儲存在中的資產 CodeArtifact都會使用 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。每個網域都使用一個 KMS 金鑰來加密其所有資產。依預設，會使用AWS受管理的 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管理 KMS 金鑰。如需詳細資訊，請參閱AWS Key Management Service使用指南中的建立金AWS鑰和金鑰管理服務概念。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，請參閱使用中的網域 CodeArtifact。
- 傳輸中的資料加密-客戶之間以及 CodeArtifact CodeArtifact 及其下游相依性之間的所有通訊，均使用 TLS 加密保護。

流量隱私權

您可以透過設定為使用介面虛擬私有雲端 (VPC) 端點 CodeArtifact 來改善 CodeArtifact 網域及其所包含資產的安全性。若要這麼做，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，請參閱透過 PrivateLink以下AWS PrivateLink方式存取 AWS 服務。

監控 CodeArtifact

監控是維護可靠性、可用性與效能的重要環節。AWSCodeArtifact 和AWS解決方案。您應該全面收集監控資料，AWS解決方案，以便在發生多點故障的情況下，更輕鬆地偵錯。AWS提供以下資訊，以監控 CodeArtifact 資源並回應潛在事件：

主題

- 使用記錄 CodeArtifact API 呼叫AWS CloudTrail

使用記錄 CodeArtifact API 呼叫AWS CloudTrail

CodeArtifact 與整合AWS CloudTrail，該服務提供記錄使用者、角色或AWS服務在 CodeArtifact 中。CloudTrail 會將 CodeArtifact 的所有 API 呼叫捕獲為事件，包括來自包管理器客戶端的呼叫。

如果您建立追蹤，就可以將 CloudTrail 事件持續提供給 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 Event history (事件歷史記錄) 檢視最新事件。您可以使用由 CloudTrail 收集的資訊來判斷對 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

當您建立帳戶時，系統即會在 AWS 帳戶中啟用 CloudTrail。此外，CodeArtifact 發生活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他AWS中的服務事件事件歷史記錄。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

若要持續記錄AWS帳戶（包括 CodeArtifact 的事件），請創建線索。追蹤能讓 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。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的

Amazon S3 儲存貯體。您也可以設定其他AWS服務，以進一步分析 CloudTrail 日誌中所收集的事件資料，並採取相應動作。如需詳細資訊，請參閱下列主題：

- 建立 AWS 帳户的追蹤
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知

當啟用 CloudTrail 日誌記錄時，在AWS帳户時，對 CodeArtifact 發出的 API 呼叫會在 CloudTrail 日誌檔案中追蹤，與其他AWS服務記錄。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，呼叫至`ListRepositories`（在AWS CLI、`aws codeartifact list-repositories`),`CreateRepository`(`aws codeartifact create-repository`），以及`ListPackages`(`aws codeartifact list-packages`）動作會在 CloudTrail 日誌檔案中產生項目，以及包管理器客户端命令。軟件包管理器客户端命令通常向服務器發出多個 HTTP 請求。每個請求都會生成一個單獨的 CloudTrail 日誌事件。

跨帳户交付 CloudTrail 日誌

最多三個獨立的帳户會收到用於單個 API 調用的 CloudTrail 日誌：

- 發出請求的帳户，例如，調用`GetAuthorizationToken`。
- 存儲庫管理員帳户 — 例如，管理`ListPackages`被召喚.
- 網域所有者的帳户，例如，擁有包含已呼叫 API 的資料庫的網域的帳户。

對於像`ListRepositoriesInDomain`是針對域而不是特定存儲庫的操作，則只有調用帳户和域所有者的帳户才會收到 CloudTrail 日誌。對於像`ListRepositories`，則只有調用者的帳户接收 CloudTrail 日誌。

瞭解 CodeArtifact 日誌檔案項目

CloudTrail 日誌檔案可包含一個或多個日誌項目。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：一個日誌條目，用於調用 GetAuthorizationToken API

- 範例：用於獲取 npm 軟件包版本的日誌條目

範例：一個日誌條目，用於調用 GetAuthorizationToken API

創建的日誌條目GetAuthorizationToken會將網域名稱包含在requestParameters欄位。

```
{
  "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：用於獲取 npm 軟件包版本的日誌條目

所有軟件包管理器客户端發出的請求，包括 **npm** 客户端，記錄其他數據，其中包括域名、存儲庫名稱和軟件包名稱`requestParameters`欄位。URL 路徑和 HTTP 方法記錄在`additionalEventData`欄位。

```
{
   "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

要瞭解 AWS 服務 是否在特定法規遵循方案範圍內，請參閱法規遵循方案範圍內的 AWS 服務，並選擇您感興趣的法規遵循方案。如需一般資訊，請參閱 AWS 法規遵循方案。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱 AWS Artifact 中的下載報告。

您使用 AWS 服務 時的法規遵循責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理法規遵循事宜：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心的基準環境的架構考量和步驟。
- Amazon Web Services 的 HIPAA 安全與法規遵循架構：本白皮書說明公司可如何運用 AWS 來建立符合 HIPAA 規定的應用程式。

> **Note**
>
> 並非全部的 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源：這組手冊和指南可能適用於您的產業和位置。
- AWS 客戶合規指南：透過合規的角度瞭解共同的責任模式。這份指南橫跨多個架構 (包含國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準組織 (ISO))，總結保護 AWS 服務的最佳實務並將指導方針對應至安全控制。

- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估您的資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 此 AWS 服務 可供您全面檢視 AWS 中的安全狀態。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 controls reference。
- AWS Audit Manager – 此 AWS 服務 可協助您持續稽核 AWS 使用情況，以簡化管理風險與法規與業界標準的法規遵循方式。

AWS CodeArtifact 身份驗證和令牌

CodeArtifact 要求使用者透過服務進行驗證，才能發佈或使用套件版本。您必須使用憑據創建授權令牌來對 CodeArtifact 服務進行身份驗 AWS 證。為了創建授權令牌，您必須具有正確的權限。如需建立授權權杖所需的權限，請參閱中的`GetAuthorizationToken`項目AWS CodeArtifact 權限參考。如需有關 CodeArtifact 權限的更多一般資訊，請參閱如何與 IAM AWS CodeArtifact 搭配使用。

要從中獲取授權令牌 CodeArtifact，您必須調用 GetAuthorizationToken API。使用 AWS CLI，您可以使`GetAuthorizationToken`用`login`或`get-authorization-token`命令來呼叫。

> **Note**
>
> 根使用者無法呼叫`GetAuthorizationToken`。

- `aws codeartifact login`：這個命令可讓您輕鬆設定一般套件管理員，以便 CodeArtifact 在單一步驟中使用。調用`login`獲取令牌，`GetAuthorizationToken`並使用令牌和正確 CodeArtifact 的存儲庫端點配置您的包管理器。支持包管理器如下：
  - dotnet
  - NPM
  - 金塊
  - pip
  - 迅速
  - 纏繞
- `aws codeartifact get-authorization-token`: 對於不受支援的套件管理員`login`，您可以`get-authorization-token`直接呼叫，然後根據需要使用 Token 來設定套件管理員，例如，將它新增至組態檔或將其儲存為環境變數。

CodeArtifact 授權令牌在 12 小時的默認時間內有效。令牌可以在 15 分鐘到 12 小時之間配置生命週期。當生命週期到期時，您必須獲取另一個令牌。令牌生命週期在login或get-authorization-token被調用之後開始。

如果get-authorization-token在假設角色時呼叫login或，您可以將值設定為，將權杖的生命週期設定為等於角色工作階段持續時間中的剩餘時間0。--duration-seconds否則，權杖存留期與角色的工作階段持續時間上限無關。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以擷取 CodeArtifact 授權權杖。在這種情況下，即使這比 15 分鐘的會話持續時間長，令牌在整個 12 小時內有效。如需控制工作階段持續時間的相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用login指令建立的權杖

該aws codeartifact login命令將獲取令牌，GetAuthorizationToken並使用令牌和正確的 CodeArtifact 存儲庫端點配置您的包管理器。

下表說明命login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dotnetnpm、nuget、pip、swift和twir
--domain	是	存放庫所屬的網域名稱。
--domain-owner	否	網域擁有者的識別碼。如果存取您未經驗證的 AWS 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	要進行驗證的存放庫名稱。
--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	將命名空間與儲存庫工具相關聯。
--dry-run	否	僅列印將執行的指令，以連接您的工具與儲存庫，而不對您的組態進行任何變更。

參數	必要	描述
* login 在假設角色時調用時，值為 0 也是有效的。login使用調用會--duration-seconds 0創建一個令牌，其生命週期等於假定角色的會話持續時間中的剩餘時間。		

下面的示例演示了如何使用命令獲取授權login令牌。

```
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 使用指login令的特定指引，請參閱配置和使用 npm CodeArtifact。如需 Python，請參閱使用CodeArtifact與蟒蛇。

使用 GetAuthorizationToken API 建立的權杖

您可以調get-authorization-token用從中獲取授權令牌 CodeArtifact。

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text
```

您可以使用--duration-seconds參數更改令牌的有效時間。最小值為 900，最大值為 43200。下列範例會建立持續 1 小時 (3600 秒) 的權杖。

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 3600
```

如果get-authorization-token在假設角色的情況下調用，則令牌生命週期與角色的最大會話持續時間無關。您可以將權杖設定為在假定角色的工作階段持續時間到期時過期，方法是設定--duration-seconds為 0。

```
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：

- 有關令牌和環境變量的指導，請參閱使用環境變量傳遞身份驗證令牌。
- 如需 Python 使用者，請參閱無需登錄命令即可配置 pip或配置和使用麻線CodeArtifact。
- 對於 Maven 使用者，請參閱使用CodeArtifact與搖籃或 CodeArtifact 與 mvn 一起使用。
- 如需 npm 使用者，請參閱在不使用登錄命令的情況下配置 npm。

使用環境變量傳遞身份驗證令牌

AWS CodeArtifact 使用 `GetAuthorizationToken` API 提供的授權令牌來驗證和授權構建工具（例如 Maven 和 Gradle）的請求。有關這些身份驗證令牌的更多信息，請參閱使用 `GetAuthorizationToken` API 建立的權杖。

您可以將這些身份驗證令牌存儲在可以由構建工具讀取的環境變量中，以獲取從 CodeArtifact 存儲庫中獲取軟件包或將軟件包發佈到其中所需的令牌。

出於安全原因，這種方法比將令牌存儲在可能被其他用戶或進程讀取的文件中，或者意外簽入源代碼控制中的文件中更好。

1. 如中所述設定您的 AWS 認證安裝或升級，然後設定 AWS CLI。
2. 設定 `CODEARTIFACT_AUTH_TOKEN` 環境變數：

> **Note**
>
> 在某些情況下，您不需要包含引`--domain-owner`數。如需詳細資訊，請參閱 跨帳戶網域。

- macOS 系統或

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text`
```

- 視窗 (使用預設命令介面):

```
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
```

- 視窗 PowerShell:

```
$env:CODEARTIFACT_AUTH_TOKEN = aws codeartifact get-authorization-token --
domain my_domain --domain-owner 111122223333 --query authorizationToken --output
 text
```

撤銷 CodeArtifact 授權令牌

當經過身份驗證的用戶創建令牌以訪問 CodeArtifact 資源時，該令牌將持續到其可自定義的訪問期結束為止。預設存取期限為 12 小時。在某些情況下，您可能希望在訪問期限過期之前撤銷對令牌的訪問權限。您可以按照以下說明撤銷對 CodeArtifact 資源的存取權。

如果您使用臨時安全登入資料 (例如假設角色或聯合身分使用者存取權) 建立存取權杖，則可以透過更新 IAM 政策以拒絕存取權來撤銷存取權。如需詳細資訊，請參閱 IAM 使用者指南中的停用臨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。

韌性在AWS CodeArtifact

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。AWSCodeArtifact在多個可用區域中運作，並將人工因素資料和中繼資料存放在 Amazon S3 和 Amazon DynamoDB 中。您的加密資料會以冗餘方式儲存在多個設施和每個設施中的多個裝置，因此具有高可用性和高度耐用性。

如需有關 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

基礎結構安全AWS CodeArtifact

作為託管服務，AWS CodeArtifact受到保護AWS全球網絡安全。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可使用 AWS 發佈的 API 呼叫，透過網路存取 CodeArtifact。用戶端必須支援下列項目：

- 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，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

依賴替換攻擊

Package 管理器簡化了打包和共享可重複使用代碼的過程。這些套件可能是由組織開發供在其應用程式中使用的私有套件，也可能是公開的，通常是開放原始碼套件，這些套件是在組織外部開發並由公開套件儲存庫散佈的。請求軟件包時，開發人員依賴他們的軟件包管理器來獲取其依賴項的新版本。相依性替換攻擊 (也稱為相依性混淆攻擊) 會利用套件管理員通常無法區分合法版本的套件與惡意版本的事實。

依賴替換攻擊屬於被稱為軟件供應鏈攻擊的黑客的一個子集。軟件供應鏈攻擊是一種利用軟件供應鏈中任何地方漏洞的攻擊。

依賴性替換攻擊可以針對使用內部開發的軟件包和從公共存儲庫中獲取的軟件包的任何人。攻擊者識別內部軟件包名稱，然後策略性地將具有相同名稱的惡意代碼放置在公共軟件包存儲庫中。一般而言，惡意程式碼會以較高版本號碼的套件發佈。Package 件管理員會從這些公開摘要擷取惡意程式碼，因為他們認為惡意套件是套件的最新版本。這會導致所需的軟件包和惡意軟件包之間出現「混淆」或「替換」，從而導致代碼遭到入侵。

為了防止依賴替換攻擊， AWS CodeArtifact 提供包源控件。Package 件原始控制項是控制套件新增至儲存庫的方式的設定。這些控制項可用來確保套件版本不能直接發佈到您的儲存庫，也不能從公開來源擷取，保護您免受相依性替代攻擊。您可以在套件群組上設定原始控制項，在個別套件和多個套件上設定 Origin 控制項。如需有關套件來源控制項以及如何變更它們的詳細資訊，請參閱編輯套件原點控制項和Package 群組原點控制項。

的 Identity and Access Management AWS CodeArtifact

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 CodeArtifact 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在進行的工作 CodeArtifact。

服務使用者 — 如果您使用 CodeArtifact 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 CodeArtifact 功能來完成工作時，您可能需要其他權限。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。如果無法存取中的圖徵 CodeArtifact，請參閱疑難排解 AWS CodeArtifact 身分和存取。

服務管理員 — 如果您負責公司的 CodeArtifact 資源，您可能擁有完整的存取權 CodeArtifact。決定您的服務使用者應該存取哪些 CodeArtifact 功能和資源是您的工作。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。若要進一步瞭解貴公司如何搭配使用 IAM CodeArtifact，請參閱如何與 IAM AWS CodeArtifact 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策來管理存取權限的詳細資訊 CodeArtifact。若要檢視可在 IAM 中使用的 CodeArtifact 基於身分的政策範例，請參閱。以身分識別為基礎的原則範例 AWS CodeArtifact

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴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識別的範例。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 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

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) 來增加帳戶的安全性。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 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 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

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

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，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用

程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，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 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，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，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要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 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。

- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。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。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。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 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。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 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。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暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，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哪些內容。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。具有該原則的使用者可以從 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。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

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。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，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。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (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 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 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原則類型。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。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。

- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。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 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

如何與 IAM AWS CodeArtifact 搭配使用

在您使用 IAM 管理存取權限之前 CodeArtifact，請先了解哪些 IAM 功能可搭配使用 CodeArtifact。

您可以搭配使用的 IAM 功能 AWS CodeArtifact

IAM 功能	CodeArtifact 支持
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	否
ACL	否
ABAC(政策中的標籤)	部分

IAM 功能	CodeArtifact 支持
臨時憑證	是
主體許可	是
服務角色	否
服務連結角色	否

若要深入瞭解如何以 CodeArtifact 及其他 AWS 服務如何使用大多數 IAM 功能，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

以身分識別為基礎的原則 CodeArtifact

支援身分型政策	是

身分型政策是可以連接到身分 (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。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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。

以身分識別為基礎的原則範例 CodeArtifact

若要檢視以 CodeArtifact 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS CodeArtifact

以資源為基礎的政策 CodeArtifact

支援以資源基礎的政策	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。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作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。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。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 角色與資源型政策有何差異。

的政策動作 CodeArtifact

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哪些內容。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。

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。原則動作通常與關聯的 AWS API 作業具有相同的名稱。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 AWS CodeArtifact中所定義的動作。

中的策略動作在動作之前 CodeArtifact 使用下列前置詞：

```
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*"
```

若要檢視以 CodeArtifact 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS CodeArtifact

的政策資源 CodeArtifact

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哪些內容。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 ARN 的清單，請參閱服務授權參考 AWS CodeArtifact中所定義的資源。若要瞭解可以使用哪些動作指定每個資源的 ARN，請參閱由 AWS CodeArtifact定義的動作。若要查看在策略中指定 CodeArtifact 資源 ARN 的範例，請參閱AWS CodeArtifact 資源與營運。

的政策條件索引鍵 CodeArtifact

支援服務特定政策條件金鑰	否

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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，請使用邏輯OR運算來 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 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

> **Note**
>
> AWS CodeArtifact 不支援下列 AWS 全域條件內容索引鍵：
>
> - Referer
> - UserAgent

若要查看 CodeArtifact 條件索引鍵清單，請參閱服務授權參考 AWS CodeArtifact中的條件金鑰。若要瞭解您可以使用條件索引鍵的動作和資源，請參閱定義的動作 AWS CodeArtifact。

若要檢視以 CodeArtifact 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS CodeArtifact

ACL 在 CodeArtifact

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。

阿巴克與 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。您可以將標籤附加到 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。

若要根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件金鑰，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》中的什麼是 ABAC?。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)。

如需有關標記 CodeArtifact資源的詳細資訊，包括以身分識別為基礎的政策範例，以根據該資源上的標籤限制對資源的存取，請參閱。使用標籤來控制對 CodeArtifact 資源的存取

使用臨時登入資料 CodeArtifact

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。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》中的切換至角色 (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。

的跨服務主體權限 CodeArtifact

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，您會被視為主體。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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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 動作需要呼叫主體在其他服務中具有權限：

1. `GetAuthorizationToken`需要`sts:GetServiceBearerToken`與`codeartifact:GetAuthorizationToken`.
2. `CreateDomain`提供非預設加密金鑰時，需要`kms:DescribeKey`和`kms:CreateGrant`在 KMS 金鑰上同時使用`codeartifact:CreateDomain`。

如需中動作之必要權限和資源的詳細資訊 CodeArtifact，請參閱AWS CodeArtifact 權限參考。

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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的權限可能會中斷 CodeArtifact 功能。只有在 CodeArtifact 提供指引時才編輯服務角色。

服務連結角色 CodeArtifact

支援服務連結角色。	否

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。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。

以身分識別為基礎的原則範例 AWS CodeArtifact

依預設，使用者和角色沒有建立或修改 CodeArtifact資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。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。

若要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。

如需有關由所定義之動作和資源類型的詳細資訊 CodeArtifact，包括每個資源類型的 ARN 格式，請參閱服務授權參考 AWS CodeArtifact中的動作、資源和條件索引鍵。

主題

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 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 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您的使用案例的 AWS 客戶管理政策，進一步降低使用權限。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。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，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，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 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，您必須擁有最少一組權限。這些權限必須允許您列出和檢視有關 AWS 帳戶. CodeArtifact 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。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。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。

若要確保使用者和角色仍可使用 CodeArtifact 主控台，請同時將`AWSCodeArtifactAdminAccess`或受`AWSCodeArtifactReadOnlyAccess` AWS 管理的原則附加至實體。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 政策來解決許多常見使用案例 AWS。這些 AWS 受管理的政策會為常見使用案例授與必要的權限，因此您可以避免調查需要哪些權限。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 受管理的策略 (您可以附加至帳戶中的使用者) 是特定的 AWS CodeArtifact。

- `AWSCodeArtifactAdminAccess`— 提供完整存取權限， CodeArtifact 包括管理 CodeArtifact 網域的權限。

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

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

- `AWSCodeArtifactReadOnlyAccess`— 提供對的唯讀存取 CodeArtifact。

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

```
}
```

若要建立和管理 CodeArtifact 服務角色，您還必須附加名為的 AWS 受管理策略`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權限

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。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "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 使用者或角色列出並描述任何類型的 CodeArtifact 資源，包括網域、存放庫、套件和資產。此原則也包含`codeArtifact:ReadFromRepository`權限，可讓主體從 CodeArtifact 存放庫擷取套裝程式。它不允許創建新的域或存儲庫，也不允許發布新的軟件包。

呼叫 `GetAuthorizationToken` API 需要`codeartifact:GetAuthorizationToken`和`sts:GetServiceBearerToken`權限。

```
{
   "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

以下顯示權限原則範例，該原則允許使用者僅在該`us-east-2`區域中列出以該名稱開頭之任何網域的帳戶`123456789012`的網域`my`。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codeartifact:ListDomains",
         "Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/my*"
      }
   ]
}
```

允許使用者取得有關特定儲存庫的資訊

以下顯示權限原則的範例，可讓使用者取得以結尾之儲存庫的相關資訊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權限。

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

限制授權令牌時間

用戶必須使用授權令牌進行身份驗證，才能發布或使用包版本。 CodeArtifact 授權令牌僅在其配置的生命週期內有效。令牌的默認生命週期為 12 小時。如需授權權杖的詳細資訊，請參閱AWS CodeArtifact 身份驗證和令牌。

獲取令牌時，用戶可以配置令牌的生命週期。授權令牌生命週期的有效值為900（15 分鐘）到43200（12 小時）之間的任何數字。0的值0會建立持續時間等於使用者角色暫時登入資料的權杖。

系統管理員可以使用附加至使用者或群組的權限原則中的sts:DurationSeconds條件金鑰，來限制授權權杖存留期的有效值。如果用戶嘗試創建具有有效值以外的生命週期的授權令牌，則令牌創建將失敗。

以下示例策略限制了 CodeArtifact 用戶創建的授權令牌的可能持續時間。

示例策略：將令牌存留期限制為正好 12 小時（43200 秒）

使用此政策，用戶將僅能夠創建有效期為 12 小時的授權令牌。

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

範例原則：將權杖存留期限制在 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，用戶將能夠創建 15 分鐘到 1 小時之間有效的令牌。使用者也可以透過指定0為來建立持續其角色暫時登入資料持續時間的權杖--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"
```

```
                }
            }
        }
    ]
}
```

使用標籤來控制對 CodeArtifact 資源的存取

IAM 使用者政策陳述式中的條件，屬於您用來指定CodeArtifact動作所需的資源存取許可的語法。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 CodeArtifact 資源標籤功能的詳細資訊，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，資源可以擁有標籤，也有一些動作會包含標籤。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存放庫資源已有的標籤，對其執行動作。
- 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，標籤必須位於動作所在的資源上。例如，若要拒絕具有標籤的`DescribeRepository`權限，標籤必須位於每個存放庫，而不是網域。請參AWS CodeArtifact 權限參考閱，以取得其中的動作CodeArtifact及其運作的資源清單。

標籤型存取取取取取制

以下範例顯示了如何指定 CodeArtifact 使用者政策中的標記條件。

Example 1：根據請求中的標籤限制動作

AWSCodeArtifactAdminAccess 受管使用者政策允許使用者在任何資源上執行 CodeArtifact 動作，許可不受限制。

下列政策會限制此能力，並拒絕未授權的使用者許可，禁止其建立存取存取存取存取存放放放放放庫。為了達到此種效果，如果該請求未指定名為的標記，costcenter含有1或2的其中一值，其會拒絕CreateRepository動作。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 受管使用者政策允許使用者在任何資源上執行 CodeArtifact 動作，許可不受限制。

下列政策會限制此能力，並拒絕未授權的使用者許可，禁止其在指定的網域中的存取制動作。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，禁止在中執行動作，並取得有關存取有關的存取得資訊CodeArtifact。

為了達到此種效果，如果存取名為的標記，含有名為的標記，含有名為Key1的標記，含有名為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。此受管使用者政策可提供使用者不受限制的許可，以在任何資源上執行 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

下列政策會授予使用者在中的指定網域中建立存取存放庫的許可CodeArtifact。

為此，如果請求中的創建資源 API 指定了以值命名Key1的標籤，則允許CreateRepository和TagResource操作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。此受管使用者政策可提供使用者不受限制的許可，以在任何資源上執行 CodeArtifact 動作。

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

```
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

AWS CodeArtifact 權限參考

AWS CodeArtifact 資源與營運

在中 AWS CodeArtifact，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。存儲庫也是資源，並具有與之相關聯的 ARN。如需詳細資訊，請參閱中的 Amazon 資源名稱 (ARN)。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*
Package 群組	arn:aws:codeartifact: *region-ID* :*account-ID* :package-group/ *my_domain* /*encoded_package_group_pattern*
Package 含命名空間	arn:aws:codeartifact: *region-ID* :*account-ID* :package/ *my_domain* /*my_repo*/*package-format* /*namespace* /*my_package*
沒有命名空間的 Package	arn:aws:codeartifact: *region-ID* :*account-ID* :package/ *my_domain* /*my_repo*/*package-format* //*my_package*
所有 CodeArtifact 資源	arn:aws:codeartifact:*

資源類型	ARN 格式
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。

您可以使用它的 ARN 在陳述式中指定特定## (*MyDomain*)，如下所示。

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

您可以使用它的 ARN 在您的聲明中指定一個特定的存儲庫（*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裝程式和儲存庫。

```
"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 ARN 中的相同字元。但是，在資源模式和規則中 CodeArtifact 使用完全匹配。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 操作和權限

當您設定存取控制和撰寫可附加至 IAM 身分 (身分型政策) 的許可政策時，您可以使用下表做為參考。

您可以在 AWS CodeArtifact 原則中使用 AWS寬條件金鑰來表示條件。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:*指定所有 CodeArtifact 動作，並codeartifact:Describe*指定以該字開頭的所有 CodeArtifact 動作Describe。

Package 群組 ARN

> **Note**
>
> 本節說明套件群組 ARN 和模式編碼如何提供資訊。建議您從控制台複製 ARN，或使用 DescribePackageGroup API 獲取 ARN，而不是編碼模式和構建 ARN。

IAM 政策使用萬用字元*, 來比對多個 IAM 動作或多個資源。Package 群組模式也會使用*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，封裝群組 ARN 格式如下：

```
arn:aws:codeartifact:region:account-ID:package-
group/my_domain/encoded_package_group_pattern
```

其中編碼的包組模式是包組模式，某些特殊字符替換為其百分比編碼值。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 (/*) 的根套件群組的 ARN 會是：

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

請注意，不包括在列表中的字符不能被編碼，並且 ARN 是區分大小寫的，因此*必須編碼為%2a和不。%2A

疑難排解 AWS CodeArtifact 身分和存取

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 CodeArtifact 常見問題。

主題

我沒有執行操作的授權 CodeArtifact

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 系統管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 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：

- 若要瞭解是否 CodeArtifact 支援這些功能，請參閱如何與 IAM AWS CodeArtifact 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。

- 若要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)。
- 若要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 角色與資源型政策的差異。

使用 Amazon VPC 端點

您可以設定 CodeArtifact 為使用介面虛擬私有雲 (VPC) 端點來改善 VPC 的安全性。

VPC 端點使用這項服務 AWS PrivateLink，可讓您透過私有 IP 位址存取 CodeArtifact API。 AWS PrivateLink 限制 VPC 和 CodeArtifact AWS 網路之間的所有網路流量。使用介面 VPC 端點時，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 區域要求。確保您在計劃向其發出 API 呼叫的相同 AWS 區域中建立端點 CodeArtifact。
- 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

為以下項目建立 VPC 端點 CodeArtifact

若要為其建立虛擬私有雲端 (VPC) 端點 CodeArtifact，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 使用者指南中的介面 VPC 端點 (AWS PrivateLink)。

需要兩個 VPC 端點，以便所有要求 CodeArtifact 都位於 AWS 網路中。第一個端點用於呼叫 CodeArtifact API (例如`GetAuthorizationToken`和`CreateRepository`)。

```
com.amazonaws.region.codeartifact.api
```

第二個端點用於使用包管理器和構建工具（例如 npm 和 Gradle）訪問 CodeArtifact 存儲庫。

```
com.amazonaws.region.codeartifact.repositories
```

以下命令創建一個端點來訪問 CodeArtifact 存儲庫。

```
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 --no-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`選項建立私人 DNS 主機名稱。但是，由於`codeartifact.api`和`codeartifact.repositories`端點目前不支援多個私人 DNS 主機名稱，因此請使用的`--no-private-dns-enabled``codeartifact.api`選項。如果您無法或不想在建立`codeartifact.repositories`端點時建立私人 DNS 主機名稱，則必須遵循額外的設定步驟， CodeArtifact 從 VPC 使用套件管理員。如需更多資訊，請參閱使用沒有私有 DNS 的`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，才能將端點搭配 CodeArtifact使用。如需 Amazon VPC 中安全群組的詳細資訊，請參閱安全群組。

如果您在連線時遇到問題 CodeArtifact，可以使用 VPC 可 Reachability Analyzer 工具來偵錯問題。如需詳細資訊，請參閱什麼是 VPC Reach ability Analyzer？

建立 Amazon S3 閘道端點

CodeArtifact 使用亞馬遜簡單儲存服務 (Amazon S3) 來存放套件資產。若要從中提取套件 CodeArtifact，您必須為 Amazon S3 建立閘道端點。當您的建置或部署程序從下載套件時 CodeArtifact，必須存取 CodeArtifact 以取得套件中繼資料，Amazon S3 才能下載套件資產 (例如 Maven `.jar` 檔案)。

> **Note**
>
> 使用 Python 或斯威夫特套件格式時，不需要使用 Amazon S3 端點。

若要為其建立 Amazon S3 閘道端點 CodeArtifact，請使用 Amazon EC2 `create-vpc-endpoint` AWS CLI 命令。建立端點時，您必須為 VPC 選取路由表。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
```

Amazon S3 儲存貯體的最低許可 AWS CodeArtifact

Amazon S3 閘道端點會使用 IAM 政策文件來限制服務的存取權限。若只要允許最低 Amazon S3 儲存貯體許可 CodeArtifact，請限制存取 Amazon S3 儲存貯體，該儲存貯體在您為端點建立 IAM 政策文件時 CodeArtifact 使用此儲存貯體。

下表說明您應在政策中參考以允許在每個區域存取的 Amazon S3 儲存貯體。 CodeArtifact

區域	Amazon S3 桶 ARN
us-east-1	骨架:符號:: 3:: 資產
us-east-2	骨架:符號:S3:: 資產管理系統-美國東部 2
us-west-2	阿姆斯特丹:: 三:: 資產-美國西部
eu-west-1	骨架:符號:S3:: 資產
eu-west-2	系統：符號：3：：資產-歐盟西部 2
eu-west-3	系統：AWS：3：：資產 -762466490029-歐盟西部
eu-north-1	航天:符號:S3:: 資產
eu-south-1	航天:符號:3:: 資產

區域	Amazon S3 桶 ARN
eu-central-1	骨架:符號:S3:: 資產-歐盟中央 -1
ap-northeast-1	系統：AWS：3： ：資產 -660291247815-應用程序東北 -1
ap-southeast-1	企業管理系統:: 三:: 資產管理
ap-southeast-2	骨架:符號:S3:: 資產 -860415559748-安培東南部
ap-south-1	航天:符號:3:: 資產 -6811374769-阿里南 -1

您可以使用此`aws codeartifact describe-domain`命令來擷取 CodeArtifact 網域使用的 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對`us-east-1`區域中 CodeArtifact 操作所需之 Amazon S3 儲存貯體的存取權。對於其他地區，請根據上表，使用您所在地區的正確權限 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/*"]
    }
  ]
}
```

CodeArtifact 從虛擬私人雲端使用

若要使用 AWS CLI 或 SDK 使用 VPC 端點呼叫 CodeArtifact API，您必須覆寫所 CodeArtifact使用的預設端點。按照中的說明將設定 AWS CLI 為使用codeartifact.api端點取得 VPC 端點主機名稱，並使用其設定 CLI。

如果您無法或不想在您建立的 com.amazonaws.*region*.codeartifact.repositories VPC 端點上啟用私有 DNS為以下項目建立 VPC 端點 CodeArtifact，則必須為存放庫端點使用不同的組態，才能 CodeArtifact 從 VPC 使用。 CodeArtifact 如果com.amazonaws.*region*.codeartifact.repositories端點未啟使用沒有私有 DNS 的codeartifact.repositories端點用私人 DNS，請依照中的指示進行設定。

將設定 AWS CLI 為使用codeartifact.api端點

請遵循下列指示，以 com.amazonaws.*region*.codeartifact.api VPC 端點使用的主機名稱覆寫預設 CodeArtifact 主機名稱。

1. 執行下列命令以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.api \
     --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
   ```

   輸出看起來如下。

   ```
   [
   ```

```
  [
    "vpce-0743fe535b883ffff-76ddffff.api.codeartifact.us-
west-2.vpce.amazonaws.com",
    "vpce-0743fe535b883ffff-76edffff-us-west-2a.api.codeartifact.us-
west-2.vpce.amazonaws.com"
  ]
]
```

在此範例中，您可以使用任一主機名稱覆寫`com.amazonaws.`*`region`*`.codeartifact.api`端點。

2. 如果使用 CodeArtifact AWS CLI，請使用命`codeartifact login`令將端點傳遞至`--endpoint-url`參數，以 Amazon VPC 端點覆寫預設 CodeArtifact 主機名稱。請參閱以下範例。

> **Warning**
>
> 該`login`命令不支持 Maven 或搖籃。若要配置這些套件管理員，請參閱 CodeArtifact 與 Maven 一起使用。

```
aws codeartifact login --tool npm --domain mydomain --domain-owner 111122223333 --
repository myrepo --endpoint-url VPC_endpoint
```

將 *VPC_Endpoint* 取代為您的 Amazon 虛擬私人雲端端點，前綴為. `https://` 請參閱下列範例端點。

```
https://vpce-0743fe535b883ffff-76ddffff.api.codeartifact.us-
west-2.vpce.amazonaws.com
```

如果您使用 SDK，請參閱 SDK 文件以瞭解如何覆寫主機名稱。如何執行此操作會因您使用的語言而有所不同。

使用沒有私有 DNS 的`codeartifact.repositories`端點

如果您無法或不想在您建立的 `com.amazonaws.`*`region`*`.codeartifact.repositories` VPC 端點上啟用私有 DNS為以下項目建立 VPC 端點 CodeArtifact，則必須遵循這些指示，使用正確的 CodeArtifact URL 設定套件管理員。

1. 執行下列命令以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 存放庫名稱。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。

- *format*：以有效的 CodeArtifact 套件格式取代，例如，npm或pypi。
- *domain_name*：取代為包含託管套裝程式之 CodeArtifact 儲存區 CodeArtifact 域的網域。
- *#####*：以 CodeArtifact 網域擁有者的 ID 取代，例如。111122223333
- *repo_name*：取代為代管您套裝程式的 CodeArtifact 儲存區域。

以下 URL 是 npm 存儲庫端點的示例。

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. 將您的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指令的情況下設定套件管理員。如需每種套件格式的組態指示，請參閱下列文件。

- 故宮：在不使用登錄命令的情況下配置 npm
- nuget：在沒有登錄命令的情況下配置 nuget 或 dotnet
- 點子：無需登錄命令即可配置 pip
- 麻線：配置和使用麻線CodeArtifact

- 搖籃：使用CodeArtifact與搖籃
- MVN： CodeArtifact 與 mvn 一起使用

建立 CodeArtifact 的 VPC 端點政策

若要為建立 VPC 端點策略 CodeArtifact，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。

下列範例原則會指定帳戶 123456789012 中的主體可呼叫 `GetAuthorizationToken` API 並從存放庫擷取套件。 CodeArtifact

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
         "AWS": "arn:aws:iam::123456789012:root"
       }
    }
  ]
}
```

建立 CodeArtifact 資源 AWS CloudFormation

CodeArtifact 與整合的服務可協助您建立資源模型並設定資 AWS 源 AWS CloudFormation，以減少建立和管理資源和基礎架構的時間。您可以建立一個範本來描述所 AWS CloudFormation 需的所有 AWS 資源，並為您處理佈建和設定這些資源。

使用時 AWS CloudFormation，您可以重複使用範本，以一致且重複地設定 CodeArtifact 資源。只需描述您的資源一次，然後在多個帳戶和區域中一遍又一遍 AWS 地佈建相同的資源。

CodeArtifact 和 AWS CloudFormation 範本

若要佈建和設定 CodeArtifact 與相關服務的資源，您必須瞭解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，可以使用 AWS CloudFormation 設計工具來協助您開始 AWS CloudFormation 使用範本。如需詳細資訊，請參閱什麼是 AWS CloudFormation 設計師？ 在《AWS CloudFormation 使用者指南》中。

CodeArtifact 支援在中建立網域、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

防止刪除資 CodeArtifact 源

CodeArtifact 存儲庫包含重要的應用程序依賴關係，如果丟失，可能不容易重新創建。若要在使用管理 CodeArtifact 資源時防止 CodeArtifact 資源遭到意外刪除 CloudFormation，請`Retain`在所有網域`DeletionPolicy`和存放庫上包含值為的和`UpdateRetainPolicy`屬性。如果資源從堆棧模板中刪除，或者整個堆棧意外刪除，這將防止刪除。下列 YAML 程式碼片段會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
```

若要取得有關這些屬性的更多資訊，請參閱《AWS CloudFormation 使用指南》UpdateReplacePolicy中的DeletionPolicy和。

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 指令行介面使用者指南

AWS CodeArtifact 疑難排解

下列資訊可協助您疑難排解與的常見問題 CodeArtifact。

如需疑難排解格式特定問題的相關資訊，請參閱下列主題：

- Maven 的故障
- 迅速的故障

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」主控台的一項功能，但目前 CodeArtifact 不支援通知。沒有 CodeArtifact 包含允許使用者檢視或管理通知的權限的受管理策略。如果您使用 Developer Tools 主控台中的其他服務，而這些服務支援通知，則這些服務的受管理原則會包含檢視和管理這些服務通知所需的權限。

標記資源

標籤是一種自訂屬性標籤，可由您或 AWS 指派給 AWS 資源。每個 AWS 標籤都有兩個部分：

- 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整理 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AWS CodeBuild專案的儲存庫。

如需使用標籤的秘訣和最佳做法，請參閱標記AWS資源的最佳做法白皮書。

您可以在 CodeArtifact 中標記下列資源類型：

- 標記儲存庫 CodeArtifact
- 在中標記網域 CodeArtifact

您可以使用主控台AWS CLI、 CodeArtifact API 或 AWS SDK 來：

- 建立網域或儲存庫時，將標籤新增至網域或儲存庫 *。
- 新增、管理和移除網域或存放庫的標籤。

* 在主控台中建立網域或儲存庫時，無法將標籤新增至網域或儲存庫。

除了使用標籤識別、組織和追蹤資源之外，您還可以在 IAM 政策中使用標籤來協助控制誰可以檢視您的資源並與其互動。如需以標籤為基礎的存取政策範例，請參閱使用標籤來控制對 CodeArtifact 資源的存取。

CodeArtifact 帶有標籤的成本分配

您可以使用標籤在中分配儲存和請求成本 CodeArtifact。

分配資料儲存成本 CodeArtifact

數據存儲成本與域相關聯，因此為了分配 CodeArtifact 存儲成本，您可以使用應用於域的任何標籤。如需將標籤新增至網域的資訊，請參閱在中標記網域 CodeArtifact。

配置請求成本 CodeArtifact

大多數請求使用與存儲庫相關聯，因此為了分配 CodeArtifact 請求成本，您可以使用應用於存儲庫的任何標籤。如需將標籤新增至儲存庫的資訊，請參閱標記儲存庫 CodeArtifact。

某些請求類型與域而不是存儲庫相關聯，因此與請求相關的請求使用情況和成本將分配給域上的標記。判斷要求類型是否與網域或存放庫相關聯的最佳方法是使用「服務授權參考」中由AWS CodeArtifact表格定義的動作。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 domain，則該類型的請求將向該網域收取費用。如果資源類型是存放庫或套件，則該類型的要求會向儲存庫收取費用。某些動作會顯示兩種資源類型，對於這些動作，計費資源取決於要求中傳遞的值。

配額 AWS CodeArtifact

下表說明中的資源配額 CodeArtifact。若要檢視的資源配額以及的服務端點清單 CodeArtifact，請參閱 AWSAmazon Web Services 一般參考.

您可以要求提高下列 CodeArtifact 資源配額的服務配額。如需有關要求增加 Service Quotas 的詳細資訊，請參閱AWS 服務配額。

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。
CopyPackageVersions 每秒要求數	每個受支援的區域：5	是	每秒可撥打的最大通話 CopyPackageVersions 數。
每個儲存庫的直接上傳	每個受支援的區域：10	否	每個儲存庫的直接上游儲存庫數目上限。
每個 AWS 帳戶的網域	每個受支援的區域：10	是	每個 AWS 帳戶可建立的網域數目上限。
GetAuthorizationToken 每秒要求數	每個受支援的區域：40	是	每秒擷取的授權權杖數目上限。
GetPackageVersionAsset 每秒要求數	每個受支援的區域：50	是	每秒可撥打的最大通話 GetPackageVersionAsset 數。
ListPackageVersionAssets 每秒要求數	每個受支援的區域：200	是	每秒可撥打的最大通話 ListPackageVersion Assets 數。

名稱	預設	可調整	描述
ListPackageVersions 每秒要求數	每個受支援的區域：200	是	每秒可撥打的最大通話 ListPackageVersions 數。
ListPackages 每秒要求數	每個受支援的區域：200	是	每秒可撥打的最大通話 ListPackages 數。
PublishPackageVersion 每秒要求數	每個受支援的區域：10	是	每秒可撥打的最大通話 PublishPackageVersion 數。
從單一 AWS 帳戶每秒讀取要求	每個支持地區：800	是	每秒來自一個 AWS 帳戶的讀取請求數目上限。
每個域的存儲庫	每個受支援的區域：1,000	是	每個網域可建立的儲存庫數目上限。
使用單一驗證 Token 的每秒要求數	每個支持的地區：1,200	否	使用單一驗證 Token 的每秒要求數目上限。
每個 IP 地址沒有驗證令牌的請求	每個支持的地區：600	否	來自單一 IP 位址，沒有驗證 Token 的每秒要求數目上限。
搜尋上游儲存庫	每個受支援的區域：25	否	解析套件時搜尋的上游儲存庫數目上限。
從單一 AWS 帳戶每秒寫入要求	每個受支援的區域：100	是	每秒來自一個 AWS 帳戶的寫入要求數目上限。

> **Note**
>
> 一般而言，發出的每個讀取請求都 CodeArtifact 算作一個請求計入配額。但是，對於 Ruby 包格式，對`/api/v1/dependencies`操作的單個讀取請求可以請求有關多個包的數據。

例如，請求可以看起來像`https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1,gem2.gem3`。在此範例中，要求會計為針對配額的三個要求。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，系統只會針對一個要求向您收取費用，但這會計算為服務配額的三個要求。

AWS CodeArtifact 使用者指南文件歷史

下表說明的文件的重要變更 CodeArtifact。

變更	描述	日期
增加了配置和使用 Ruby 的文檔 CodeArtifact	CodeArtifact 現在支持紅寶石寶石。已新增包含設定 Ruby 套件管理程式以使用 CodeArtifact 儲存庫的指引文件。如需詳細資訊，請參閱 使用 CodeArtifact 紅寶石。	2024 年 4 月 30 日
新增使用客戶管理金鑰建立網域的範例 AWS KMS 金鑰政策	新增範例金鑰原則，可用來建立客戶受管 KMS 金鑰，以加密 CodeArtifact 網域中的資產。如需詳細資訊，請參閱 AWS KMS 金鑰原則範例。	2024年4月18日
已新增文件以支援套件群組的啟動。	已新增有關在中管理和使用封裝群組的文件 CodeArtifact。如需詳細資訊，請參閱 使用套件群組 CodeArtifact。	2024年3月21日
在有關 aws codeartifact 登錄命令的文檔中添加了其他有效的軟件包管理器。	已新增`dotnetnuget`、和`swift`至要與`aws codeartifact login` 指令搭配使用的有效套件管理員清單中。如需詳細資訊，請參閱 AWS CodeArtifact 身份驗證和令牌。	2024年2月18日
在 Swift 故障排除文檔中添加了有關 Xcode 懸掛在 CI 機器上的條目	已新增資訊，包括解決方案，這些問題可能會導致 Xcode 因鑰匙串提示輸入密碼而在 CI 機器上掛起的問題。如需詳細資訊，請參閱 由於鑰匙串提示輸	2024年2月6日

	入密碼，Xcode 掛在 CI 機器上。	
已新增關於使用 npm 8.x 或更高版本進行緩慢 npm 套件安裝時間的疑難排解	已新增有關解決緩慢 npm 套件安裝時間的相關資訊 CodeArtifact，這可能會導致建置時間變慢。如需詳細資訊，請參閱 故障排除 npm 8.x 或更高版本的緩慢安裝。	2023 年 12 月 29 日
更新了有關 Python 包資產和元數據行為的信息 CodeArtifact	更新了有關 CodeArtifact 存儲庫如何保留和重新整理 Python 包版本資產和元數據的信息。如需詳細資訊，請參閱 從上流和外部連接請求 Python 包。	2023 年 12 月 14 日
重組有關監控的文件 CodeArtifact	重組監控 CodeArtifact 事件的相關資訊，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
已新增更多有關管理 CodeArtifact 資源的資訊 AWS CloudFormation	已新增有關使用管理 CodeArtifact 資源的文件參考和連結 CloudFormation，包括防止刪除使用管理的 CodeArtifact 資源的章節 CloudFormation。如需詳細資訊，請參閱 防止刪除資 CodeArtifact 源。	2023 年 12 月 7 日
已新增詳細說明 CodeArtifact 對 AWS KMS 外部金鑰存放區 (XKS) 支援的文件	已新增區段，其中包含 KMS 金鑰支援 CodeArtifact的相關資訊，包括搭配 CodeArtifact使用 XKS 金鑰。如需詳細資訊，請參閱 支援的 AWS KMS 金鑰類型 CodeArtifact。	2023 年 10 月 31 日

更新現有的和添加了新的故障排除	添加了 Maven 故障排除主題，並在一般的故障排除主題中包含了 Swift 和 Maven 故障排除文檔的鏈接。如需詳細資訊，請參閱 AWS CodeArtifact 疑難排解。	2023 年 9 月 28 日
更新的文檔，包括斯威夫特 Package 管理器發布命令	斯威夫特 5.9 引入了一個`swift package-registry publish` 命令來創建和發布一個 Swift 包到一個包庫。更新了 Swift 文檔，包括使用該命令的說明。如需詳細資訊，請參閱 使用 CodeArtifact 斯威夫特。	2023 年 9 月 25 日
增加了 CodeArtifact 與斯威夫特配置文檔	CodeArtifact 現在支持斯威夫特包。已新增包含設定 Swift 以使用 CodeArtifact 儲存庫的指引文件。如需詳細資訊，請參閱 使用 CodeArtifact 斯威夫特。	2023 年 9 月 20 日
添加了有關如何 CodeArtifact 處理被抽取的 Python 包版本的指導	已新增文件，其中包含有關如何判斷 Python 套件版本是否被抽取、如何 CodeArtifact 處理被抽取的套件版本，以及常見問題解答的資訊。如需詳細資訊，請參閱 猛拉包版本。	2023 年 8 月 2 日
修復了 Yarn 文檔中錯誤的命令行命令	修復了一個不正確的命令行命令，該命令行命令獲取 CodeArtifact 授權令牌並將其存儲在 Yarn 文檔中的環境變量中。	2023 年 7 月 20 日

對 Python 文檔進行了次要添加和小錯誤修復	在各自的文檔中添加了 pip 和麻線信息，並糾正了使用麻線`codeartifact login` 命令時會發生的情況。如需詳細資訊，請參閱 配置和使用 pipCodeArtifact 及 配置和使用麻線CodeArtifact。	2023 年 7 月 14 日
修正了文檔中不正確的 dotnet 命令 CodeBuild	更正了使用NuGet中的套件 CodeBuild文檔中的`dotnet add package`命令。	2023 年 7 月 13 日
更新 AWS CodeArtifact 和 AWS Identity and Access Management 文檔	在 CodeArtifact 文件中對 IAM 進行了全面修改，以增加其他 AWS 服務文件的清晰度和一致性。請參閱的 Identity and Access Management AWS CodeArtifact。	2023 年 5 月 24 日
添加了有關抽取的 Python 軟件包版本的信息	已新增有關如何 CodeArtifact 保留抽取的 Python 套件版本中繼資料的資訊，如需詳細資訊，請參閱猛拉包版本。	2023 年 4 月 11 日
已新增關於 Clojure 支援的資訊	已新增 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 CodeArtifact 與部門一起使用。	2023 年 3 月 21 日
已新增有關一般封裝發行的資	已新增有關一般套件，以及如何使用 AWS CLI. 如需詳細資訊，請參閱 CodeArtifact 搭配泛型套件使用、發佈和使用一般套件及一般套件支援的指令。	2023 年 3 月 10 日

已新增有關發佈資產大小限制的資訊	已新增區段至 Package 發佈，以說明發佈的資產大小限制。	2022 年 6 月 21 日
重構外部連線文件	移動外部連線說明文件並重新整理，以專注於使用者的最終目標，也就是將其儲 CodeArtifact 存庫連線至公用套件儲存庫。還添加了有關實現該目標的不同方法的更多指導和信息。如需詳細資訊，請參閱 將 CodeArtifact 存儲庫 Connect 到公共存儲庫。	2022 年 5 月 9 日
更新了 Amazon CloudWatch 活動的 CodeArtifact 事件信息	已新增更多資訊至`account`欄位，並新增`repositoryAdministrator` 欄位。如需詳細資訊，請參閱 CodeArtifact 事件格式與範例。	2022 年 3 月 7 日
新增了 CodeArtifact 從沒有私有 DNS 的 VPC 使用的配置說明	如果您無法或不想在 `codeartifact.repositories` VPC 端點上啟用私有 DNS，則必須為存放庫端點使用不同的組態，才能 CodeArtifact 從 VPC 使用。如需詳細資訊，請參閱使用沒有私有 DNS 的`codeartifact.repositories` 端點。	2022 年 2 月 8 日
增加了深入的文檔更新軟件包版本的狀態	將更新套件版本狀態文件擴充到其自己的主題中。已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，以及可能發生的錯誤。如需詳細資訊，請參閱更新套件版本狀態。	2021 年 9 月 1 日

更新了複製包版本文檔，其中包含更深入的權限信息	已新增有關呼叫`aws codeartifact copy-package-versions` 指令以將套件版本從一個儲存庫複製到另一個儲存庫中相同網域中 CodeArtifact的另一個儲存庫時所需的 IAM 和資源型政策許可的詳細資訊。除了更多資訊外，現在還有來源和目標儲存庫所需的以資源為基礎的原則範例。如需詳細資訊，請參閱複製套件所需的 IAM 許可。	2021 年 8 月 25 日
更新了在 IntelliJ IDEA 中運行搖籃構建的文檔	更新了在 IntelliJ IDEA 中運行 Gradle 構建的文檔，其中包含配置搖籃以從中獲取插件的步驟。 CodeArtifact 還添加了一個選項，用於通過內聯調用為每個新運行創建新的 CodeArtifact 授權令牌`aws codeartifact get-authorization-token` 。如需詳細資訊，請參閱在智能 J 理念中運行搖籃構建。	2021 年 8 月 23 日
增加了用於配置和使用紗線的文檔 AWS CodeArtifact	已新增設定及使用 Yarn 1.X 與 Yarn 2.X 來管理 npm 套件的文件。 CodeArtifact如需詳細資訊，請參閱配置和使用紗線CodeArtifact。	2021 年 7 月 30 日

AWS CodeArtifact 現在支持 NuGet 軟件包	CodeArtifact 使用者現在可以發佈和使用 NuGet 套件。已新增設定和使用 Visual Studio 和 NuGet 命令列工具 (例如`nuget` CodeArtifact 儲存庫) `dotnet` 的文件。如需詳細資訊，請參閱使用CodeArtifact取代為NuGet。	2020 年 11 月 19 日
標記資源 AWS CodeArtifact	已新增有關在中標記儲存庫和網域的文件 AWS CodeArtifact。請參閱標記資源。	2020 年 10 月 30 日
CodeArtifact 現在支持 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。建立 CodeArtifact 資源 AWS CloudFormation如需詳細資訊並開始使用，請參閱。	2020 年 10 月 8 日
新增建立與 Amazon VPC CodeArtifact 搭配使用的 Amazon S3 閘道端點的相關資訊	已新增使用 Amazon EC2 AWS CLI 命令建立 Amazon S3 閘道端點的相關資訊。本文件也包含與 Amazon VPC 環境搭配使用之特定許可的相關資訊。 CodeArtifact 請參閱建立 Amazon S3 閘道端點。	2020 年 8 月 12 日
使用捲曲發布 Maven 工件並發布第三方 Maven 工件	已新增使用 curl 進行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。